ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ
КИЇВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ІМЕНІ ТАРАСА ШЕВЧЕНКА
КАФЕДРА ТЕОРІЇ ТА ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ

"ПРОГРАМУВАННЯ: ТЕОРІЯ ТА ПРАКТИКА"

ЗБІРНИК МАТЕРІАЛІВ
ЗА РЕЗУЛЬТАТАМИ ІТ-ПРОЄКТУ
МІЖДИСЦИПЛІНАРНОЇ ІНТЕГРАЦІЇ

2023-2024 навчальний рік

КИЇВ, 2024

FACULTY OF COMPUTER SCIENCE AND CYBERNETICS
TARAS SHEVCHENKO NATIONAL UNIVERSITY OF KYIV
DEPARTMENT OF THEORY AND TECHNOLOGY OF PROGRAMMING

"PROGRAMMING: THEORY AND PRACTICE"

MATERIALS OF THE INTERDISCIPLINARY INTEGRATION IT PROJECT RESULTS

2023-2024 academic year

KYIV, 2024

УДК 004.9
П78

Рекомендовано до друку вченою радою факультету комп'ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка (протокол №17 від 28.06.2024).

Ухвалено науково-методичною комісією факультету комп'ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка (протокол № 15 від 27.06.2024 року).

Рецензенти:
Марченко О.О., доктор фізико-математичних наук, професор кафедри математичної інформатики
Шкільняк О.С., кандидат фізико-математичних наук, доцент кафедри інтелектуальних програмних систем

Загальна редакція:
Омельчук Л.Л., кандидат фізико-математичних наук, доцент кафедри теорії та технології програмування
Ткаченко О.М., кандидат технічних наук, доцент кафедри теорії та технології програмування
Шишацька О.В., кандидат фізико-математичних наук, доцент кафедри теорії та технології програмування
Русіна Н.Г., кандидат педагогічних наук, доцент кафедри теорії та технології програмування

Програмування: теорія та практика. Збірник матеріалів за результатами ІТ-проєкту міждисциплінарної інтеграції / За редакцією Л.Л. Омельчук, О.М. Ткаченка, О.В. Шишацької, Н.Г. Русіної. – Одеса: "Гельветика", 2024. – 201 с.

Збірник містить публікації студентів за результатами міждисциплінарного проєкту інтеграції на прикладі курсів "Основи управління ІТ-проєктами ", "Розробка ПЗ під мобільні платформи", "Soft Skills в інформаційних технологіях", "Інформаційні технології та правовий захист", "Інструментальні середовища та технології програмування" (спеціальність 122 "Комп'ютерні науки").

Матеріали подано в авторській редакції, відповідальність за достовірність фактів, цитат, посилань на джерела та вживання назв документів, власних імен тощо несуть автори публікацій.

УДК 004.9
П78

Recommended for printing by the Academic Council of the Faculty of Computer Science and Cybernetics at Taras Shevchenko National University of Kyiv (Protocol No.17, June 28, 2024).

Approved by the Scientific and Methodological Commission of the Faculty of Computer Science and Cybernetics of Taras Shevchenko National University of Kyiv (Protocol No. 15 June 27, 2024).

Reviewers:
O. Marchenko, Doctor in Computer Science, Professor at the Department of Mathematical Informatics.
O. Shkilniak, Ph.D. in Computer Science, Associate Professor at the Department of Intelligent Software Systems.

General Editing:
L. Omelchuk, Ph.D. Computer Science, Associate Professor at the Department of Theory and Technology of Programming
O. Tkachenko, Ph.D. in Computer Science, Associate Professor at the Department of Theory and Technology of Programming
O. Shyshatska, Ph.D. in Computer Science, Associate Professor at the Department of Theory and Technology of Programming
N.Rusina, Ph.D. in Pedagogics, Associate Professor at the Department of Theory and Technology of Programming

Programming: Theory and Practice. Materials of the interdisciplinary integration IT project results. Edited by L.Omelchuk, O.Tkachenko, O.Shyshatska, N. Rusina. – Odesa: "Helvetyka", 2024. – 201 p.

The collection includes student publications based on the results of the interdisciplinary project of integration on the example of courses "Fundamentals of IT project management", "Software development for mobile platforms", "Soft Skills in Information Technologies", "Information Technologies and Legal Protection", "Development Tools and Programming Technologies" (specialty 122 "Computer Science").

The materials are presented in the author's edition. Authors of the publications are responsible for the accuracy of facts, quotations, references to sources, and the use of document titles, personal names, etc.

ЗМІСТ

CONTENTS

ДОСВІД ВПРОВАДЖЕННЯ ПРОЄКТНОГО ПІДХОДУ ТА КРОСФУНКЦІОНАЛЬНОЇ КОЛЕКТИВНОЇ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ МІЖДИСЦИПЛІНАРНОЇ ІНТЕГРАЦІЇ

Людмила Омельчук, Олексій Ткаченко, Олена Шишацька, Наталія Русіна, Антон Свистунов, Андрій Шишацький

Роботу присвячено опису досвіду впровадження проєктного підходу та колективної розробки програмного забезпечення в кросфункціональних командах на основі міждисциплінарної інтеграції. Продемонстровано застосування такого підходу на прикладі поєднання практичних частин дисциплін освітньо-професійної програми першого рівня вищої освіти "Інформатика" (спеціальність 122 "Комп'ютерні науки"), що реалізується факультетом комп'ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка та освітньо-наукових програм другого рівня вищої освіти "Урбаністика та регіональний розвиток" (спеціальність 106 "Географія") і "Туризм" (спеціальність 242 "Туризм і рекреація"), що реалізуються географічним факультетом Київського національного університету імені Тараса Шевченка.

Ключові слова: проєктний підхід, розробка програмного забезпечення, інтеграційний підхід, кросфункціональна команда.

The work is dedicated to describing the experience of implementing a project approach and collective software development in cross-functional teams based on interdisciplinary integration. The application of this approach is demonstrated by combining the practical parts of disciplines of the B.S. program "Computer Science" (specialty 122 "Computer Sciences"), implemented by the Faculty of Computer Science and Cybernetics of Taras Shevchenko National University of Kyiv, and M.S. programs "Urban Studies and Regional Development" (specialty 106 "Geography") and "Tourism" (specialty 242 "Tourism and Recreation"), which are implemented by the Geographic Faculty of Taras Shevchenko National University of Kyiv.

Key words: project approach, software development, integration approach, cross-functional team.

ВСТУП

В умовах воєнного стану ІТ є однією з небагатьох галузей економіки України, що продовжує розвиватися і потребує нових висококваліфікованих фахівців [1]. Разом з тим, з об'єктивних причин і цю галузь не оминув спад [2], що лише посилює конкурентність на ІТ-ринку праці, а отже, підвищує вимоги і до випускників ІТ-спеціальностей. Це створює нові виклики для тих, хто реалізує відповідні освітні програми.

Основними проблемами при підготовці якісних ІТ-фахівців є невідповідність між очікуваннями роботодавців та наявним рівнем компетентностей випускників, розмитість міждисциплінарних зв'язків та недостатнє цілісне розуміння випускниками життєвого циклу розробки програмного забезпечення.

Формування професійної компетентності фахівців на основі використання міжпредметних зв'язків у навчальному процесі розглядається як важливе завдання сучасної освіти. Слід зазначити, що, незважаючи на численну кількість досліджень (як правило, теоретичних) міжпредметної інтеграції в навчальному процесі закладів вищої освіти, не розроблено конкретні методики впровадження основних положень теорії міжпредметної інтеграції відповідно профілю освіти.

Сучасна освіта передбачає різні шляхи міжпредметної інтеграції, зокрема: створення інтегрованих курсів (адаптація та інтегрування знань декількох наук); створення нових форм занять (заняття з міжпредметними зв'язками, інтегроване заняття тощо); впровадження навчальних проєктів. Останній бачиться найбільш прийнятним та ефективним у навчальних дисциплінах ІТ-галузі.

Перевагами інтегрованого навчання для студентів є чітке розуміння мети кожного предмету в різних контекстах, глибоке розуміння будь-якої теми, завдяки її дослідженню через кілька точок зору, усвідомлення комплексного підходу, через який предмети, навички, ідеї та різні точки зору пов'язані з реальним світом та вдосконалення навичок системного мислення [3-6].

Зрозуміло, що процес інтеграції вимагає виконання певних умов: об'єкти дослідження однакові або досить близькі (дослідження об'єкту з різних сторін, використовуючи навчальний матеріал різних дисциплін); у навчальних предметах використовуються однакові або близькі методи дослідження предметів та явищ (демонстрація способу пізнання дійсності на прикладах з різних предметів); пізнавані об'єкти та явища підпорядковуються загальним закономірностям (узагальнення навчального матеріалу з різних навчальних дисциплін та пізнання більш складної системи) [3-6].

УМОВИ ОРГАНІЗАЦІЇ МІЖДИСЦИПЛІНАРНОГО ПРОЄКТУ

З 2020/2021 навчального року викладачами кафедри теорії та технології програмування впроваджено використання міждисциплінарного проєктного підходу, який покликаний вирішити зазначені проблеми.

В 2020/2021-2022/2023 навчальних роках міждисциплінарна інтеграція здійснювалася на основі поєднання практичних частин дисциплін "Методи специфікації програм" і "Коректність програм та логіки програмування" (4 рік навчання) та "Інструментальні середовища та технології програмування" (2 рік навчання) освітньо-професійної програми "Інформатика" спеціальності 122 "Комп'ютерні науки". В ці роки проєкт реалізовувався у весняно-літньому семестрі [7-9].

Досвід імплементації міждисциплінарного підходу до організації командної роботи в зазначений період обумовив внесення змін в освітню програму і сприяв появі нових дисциплін "Основи управління ІТ-проєктами", "Soft Skills в інформаційних технологіях", "Інформаційні технології та правовий захист" та перегляду змісту наявних в освітній програмі дисциплін, задіяних в інтеграції.

На відміну від попередніх років, впровадження методики організації та проведення командних ІТ-проєктів в контексті міждисциплінарної інтеграції за освітньо-професійною програмою першого (бакалаврського) рівня вищої освіти "Інформатика",

що реалізується факультетом комп'ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка за спеціальністю 122 "Комп'ютерні науки" галузі знань 12 "Інформаційні технології" в 2023/2024 навальному році здійснювалося за наступними освітніми компонентами:

осінньо-зимовий семестр:

дисципліни з вибіркового блоку "Теорія та технологія програмування", студенти яких обов'язково беруть участь в проєктній роботі:

- "Розробка ПЗ під мобільні платформи" (4 рік навчання);
- "Основи управління ІТ-проєктами" (4 рік навчання);
- дисципліни освітніх програм *"Урбаністика та регіональний розвиток" та "Туризм", що реалізуються на* географічному факультеті (2 рік навчання магістратури);

весняно-літній семестр:

дисципліни з вибіркового блоку "Теорія та технологія програмування", студенти яких обов'язково беруть участь в проєктній роботі:

- "Soft Skills в інформаційних технологіях" (4 рік навчання);
- "Інформаційні технології та правовий захист" (4 рік навчання);

обов'язкова дисципліна, студенти якої беруть участь в проєктній роботі за заявою:

- "Інструментальні середовища та технології програмування" (2 рік навчання).

Для студентів, для яких участь у виконанні командних проєктів була обов'язковою, проєктна робота забезпечувала практичну складову їхніх дисциплін повною мірою. Частку практичної складової дисциплін, студенти яких брали участь у проєкті за заявою було визначено у відповідних робочих програмах.

Без втрати загальності впровадження аналогічних проєктів можливе і за іншими освітніми компонентами в межах інших освітніх програм галузі знань 12 "Інформаційні технології".

Реалізація інтеграції навчальних курсів по впровадженню методики проведення інтегрованих курсів (далі проєкт) успішно проведено в межах освітньо-професійної програми "Інформатика" бакалаврського рівня вищої освіти, що реалізується на факультеті комп'ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка за спеціальністю 122 "Комп'ютерні науки" в 2020-2024 навчальних роках.

В проєкті передбачалося досягнення таких цілей:

- посилення практичної складової навчальних дисциплін, задіяних у проєкті;
- підвищення у студентів рівня розуміння: усіх етапів життєвого циклу програмного забезпечення; підходів до управління ІТ-проєктами; міждисциплінарних зв'язків; зв'язків між теоретичним і практичним матеріалом;
- підвищення рівня професійних та соціальних навичок у студентів;
- врахування результатів проєкту при перегляді та оновленні освітньої програми та її компонентів.

Для досягнення цілей проєкту було поставлено такі задачі:

- інтеграція на прикладі теоретично орієнтованих курсів "Методи специфікації програм", "Коректність програм та логіки програмування" та практично орієнтованої курсу "Інструментальні середовища та технології програмування" (2020-2023 роки);

- інтеграція на прикладі практико-орієнтованих курсів "Розробка ПЗ під мобільні платформи", "Основи управління ІТ-проєктами" (4 рік навчання),"Soft Skills в інформаційних технологіях", "Інформаційні технології та правовий захист" (4 рік навчання), "Інструментальні середовища та технології програмування" (2 рік навчання) та дисциплін географічного факультету, що масштабувало проєкт та забезпечило кросфункціональність команд (2023-2024 роки);
- проведення прєкту, в рамках якого для виконання проєктних завдань формуються команди за різними критеріями: володіння технологіями розробки ПЗ, методологіями управління ІТ-проєктом, складом (віковий, соціально-спрямованими вподобаннями, ін.);
- розробка критеріїв оцінювання успішності запропонованого підходу до інтеграції дисциплін;
- аналіз результатів успішності проєкту, формування пропозицій щодо його вдосконалення та впровадження як кейсу на постійній основі;
- розробка програмного продукту підтримки інтеграції навчальних дисциплін.

Стейкголдерами проєкту є:

- студенти та викладачі Київського національного університету імені Тараса Шевченка бакалаврської освітньої програми "Інформатика" за спеціальністю 122 "Комп'ютерні науки", що реалізується на факультеті комп’ютерних наук та кібернетики, а також студенти та викладачі магістерських освітніх програм географічного факультету, які задіяні в рамках дисциплін (пряма аудиторія);
- замовники ІТ-продукту (пряма аудиторія);
- ІТ-компанії, студенти та викладачі інших освітніх програм за ІТ-спеціальностями (опосередкована аудиторія).

При реалізації проєкту визначено наступні показники досягнення цілей:

- підвищення рівня професійних і соціальних навичок у студентів, задіяних в проєкті (інструмент перевірки – вихідне опитування);
- наявність програмних систем, розроблених студентськими командами;
- наявність звітів за результатами роботи кожної студентської команди;
- наявність звіту викладачів про результати проєкту;
- публічний захист;
- підготовка публікацій для кожної команди за результатами проєкту.

ІМПЛЕМЕНТАЦІЯ МІЖДИСЦИПЛІНАРНОГО ПІДХОДУ ДЛЯ ОРГАНІЗАЦІЇ КОМАНДНОЇ РОБОТИ

На початку осінньо-зимового семестру студентам четвертого року навчання було запропоновано пройти анкетування, яке включало такі питання:

- чи маєте Ви досвід роботи в реальних колективних проєктах (не навчальних) (*був досвід, не маю досвіду, зараз в проєкті);*
- запропонуйте декілька варіантів предметних областей або конкретних систем Вашого майбутнього проєкту;
- вкажіть не більше, ніж 3 людини, з якими Ви б хотіли працювати в команді;

- на Вашу думку, в якій ролі в командній роботі Ви би *(почувалися впевнено, НЕ хотіли себе бачити, хотіли "прокачати" себе в проєкті / керівник проєкту, модератор, фахівець з документації, аналітик, фронтенд, бекенд, тестувальник);*
- які компетентності Ви би хотіли розвинути в рамках майбутнього проєкту (поглибити теоретичні знання, розвинути практичні вміння, програмувати, оформляти документацію, працювати в команді, планувати і організовувати свій час, вміння презентувати результати діяльності, комунікаційні навички, інше);
- досвід роботи з якою платформою і/або методологією управління IT-проєктами Ви маєте;
- в якій системі управління IT-проєктами (методології) Ви б хотіли працювати в проєкті;
- на Вашу думку, використання якої мови для Вас є *(близькою, Ви нею володієте на впевненому рівні, не бажаною, Ви не впевнені, що володієте нею на хорошому рівні, бажаною в проєкті, бо Ви хочете "прокачати" себе в ній / C#, C++, Java, PHP, Python, інше)*;
- Ваші пропозиції щодо організації проєкту.

На рис. 1-3 наведено деякі результати вхідного анкетування.

Рисунок 1 - Результати анкетування щодо досвіду участі в колективних проєєктах

Рисунок 2 - Результати анкетування щодо бажаних компетентностей

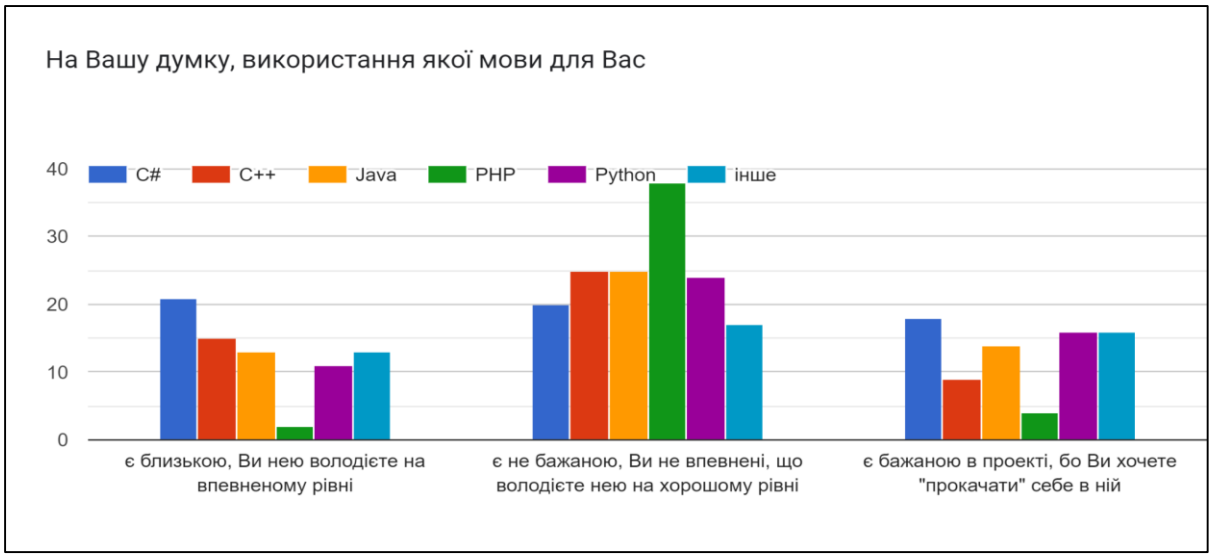


Рисунок 3 - Результати анкетування щодо рівня володіння мовами програмування

За результатами анкетування для формування команд було побудовано соціометричний орієнтований граф, який враховував досвід роботи в командних проєктах, ступінь володіння технологіями та вибір колег. З метою експерименту для формування команд застосовувалися різні підходи:

- за роком навчання: команди, що складалися виключно зі студентів четвертого року навчання та команди, що складалися зі студентів четвертого та другого років навчання;
- за соціальними вподобаннями (з урахуванням вибору бажаних партнерів по проєкту): команди з тісними соціальними зв'язками (взаємний вибір) та команди з відсутніми соціальними зв'язками;
- за досвідом участі в реальних колективних проєктах: усі учасники мали досвід, ніхто з учасників не мав досвіду, змішані команда;
- за технологіями: команди зі спільними побаженнями щодо технологій розробки та команди з різними вподобаннями.

В результаті було сформовано вісім команд, з яких сім успішно виконали проєкт. Варто зазначити, що команда, яка не завершила проєкт складалася зі студентів одного року навчання, які мали взаємний вибір (тісні соціальні зв'язки) та однакові технологічні вподобання.

В рамках проєкту здійснювалися щотижневі зустрічі учасників студентських команд з викладачами та організовано фінальний публічний захист студентських розробок. Звіти про виконання колективних робіт, розроблених в межах проєкту наведено в збірнику [7-9].

По завершенню проєкту було проведено два вихідних опитування: анонімне та авторизоване. Деякі результати цих опитувань наведено на рис. 4-8.

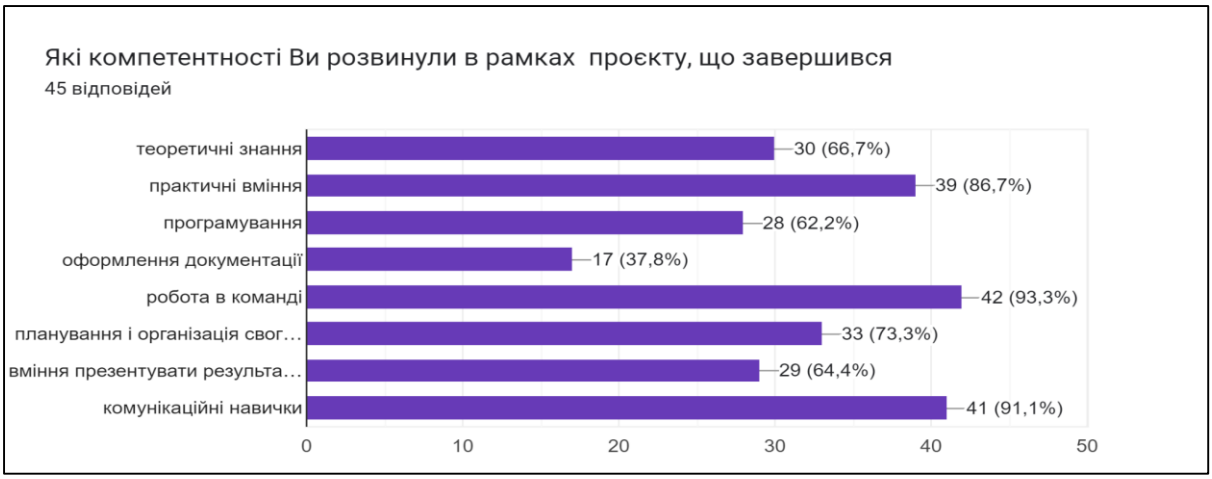


Рисунок 4 - Результати вихідного опитування щодо розвинених компетентностей

Рисунок 5 - Результати **вихідного** опитування щодо бажаних компетентностей

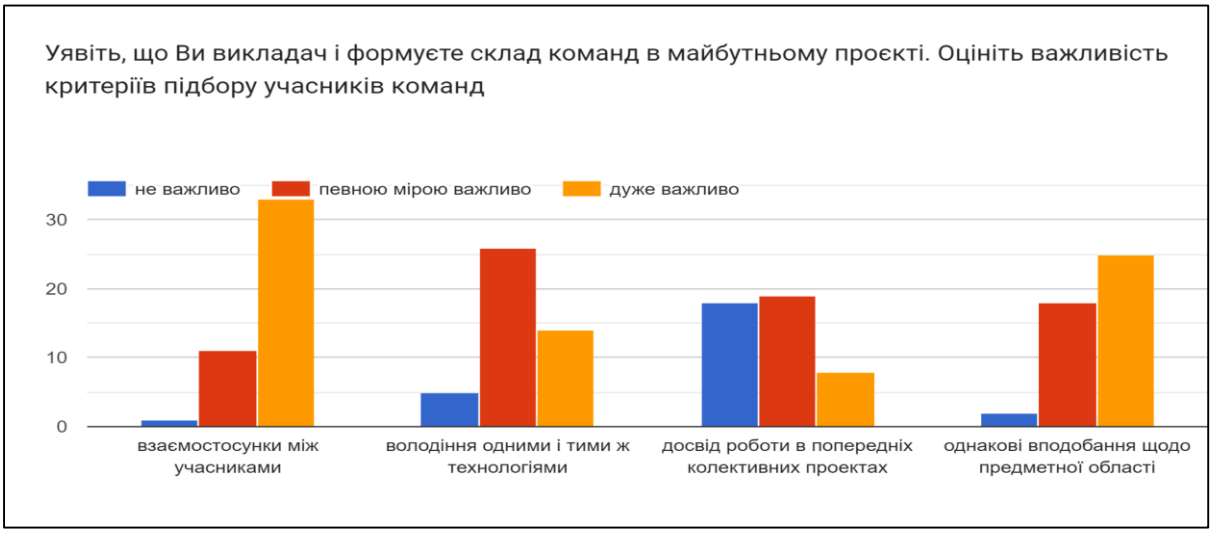


Рисунок 6 - Результати вихідного опитування щодо важливості критеріїв формування команд

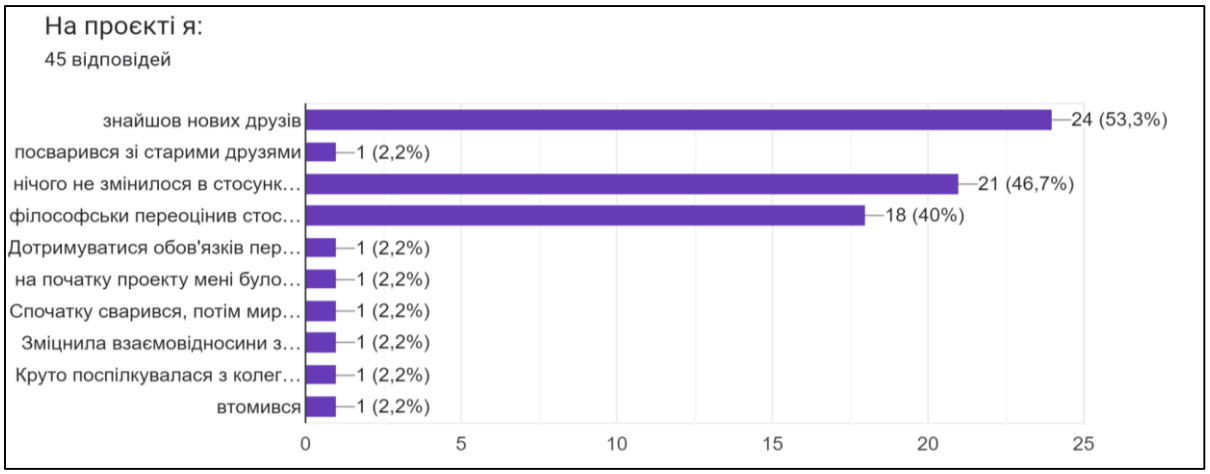


Рисунок 7 - Результати вихідного опитування – рефлексія

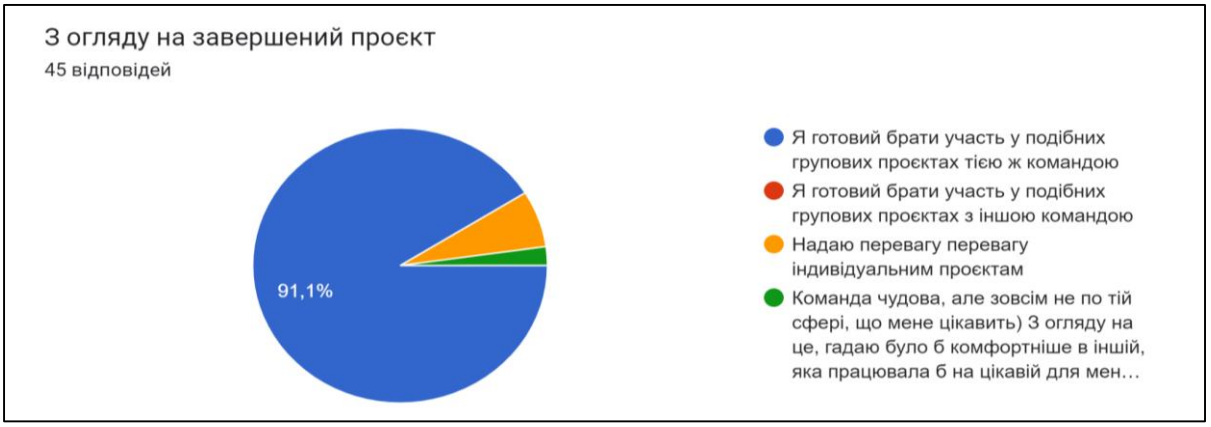


Рисунок 8 - Результати вихідного опитування – рефлексія

ВИСНОВКИ

В результаті проведення проєкту досягнуто наступні результати:

- розроблено програмні продукти підтримки інтеграції навчальних дисциплін;
- підвищено рівень професійних та соціальних навичок у студентів, задіяних в проєкті;
- підвищено у задіяних в проєкті студентів рівень розуміння (усіх етапів життєвого циклу програмного забезпечення; підходів до управління ІТ-проєктами; міждисциплінарних зв’язків в межах освітньої програми; зв’язків між теоретичним та практичним матеріалом);
- посилено практичну складову навчальних дисциплін, задіяних в проєкті;
- сформовано пропозиції щодо оновлення робочих програм навчальних дисциплін, задіяних в проєкті, та освітньої програми в цілому – впроваджувати міждисциплінарний проєктний підхід з використанням гнучких методологій управління проєктами.

Перспективи дослідження полягають у подальшому розробленні науково-методичного забезпечення освітнього процесу майбутніх фахівців у галузі інформаційних технологій. Зокрема:

- розробка та опис методики інтеграції навчальних курсів в галузі інформаційних технологій на основі проєктного підходу та гнучких методологій управління;
- вироблення рекомендацій щодо впровадження методики інтеграції навчальних курсів в галузі інформаційних технологій на основі проєктного підходу та гнучких методологій управління;
- поширення досвіду на інші навчальні дисципліни;
- впровадження програмного продукту підтримки інтеграції навчальних дисциплін, розробленого в межах проєкту;
- формування компетентностей роботи в кросфункціональних командах;
- підвищення якості освітньої програми, зокрема, в контексті формування професійних та соціальних навичок у випускників ІТ-спеціальностей.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Річний звіт Асоціації IT Ukraine 2023 [Електронний ресурс]. – Режим доступу: https://itukraine.org.ua/report/richnij-zvit-asotsiatsiyi-it-ukraine-2023/#
2. "Тенденції невтішні". Ми проаналізували обсяг українського ІТ-експорту за ІІІ квартал й запитали в експертів, чого очікувати до кінця року [Електронний ресурс]. – Режим доступу: https://dou.ua/lenta/articles/it-export-3-quarter-2023/
3. Омельчук Л.Л., Ткаченко О. М., Шишацька О. В. Впровадження методики інтеграції навчальних курсів на основі проєктного підходу та гнучких методологій управління // Програмування: теорія і практика. Збірник матеріалів за результатами ІТ-проєкту міждисциплінарної інтеграції. 2020-2021 навчальний рік / За редакцією Л.Омельчук, О.Ткаченка, О. Шишацької. – Одеса: Видавничий дім "Гельветика", 2021. С.4-10.
4. Омельчук Л.Л., Ткаченко О. М., Шишацька О. В. Інтеграція навчальних курсів на основі проєктного підходу та гнучких методологій управління // Програмування: теорія та практика. Збірник матеріалів за результатами ІТ-проєкту міждисциплінарної інтеграції. 2021-2022 навчальний рік. / За редакцією Л.Омельчук, О.Ткаченка, О. Шишацької. – Одеса: Видавничий дім "Гельветика", 2022. 155 с. С.4-11.
5. Омельчук Л.Л., Ткаченко О. М., Шишацька О. В. Методичні аспекти інтеграції навчальних курсів в галузі інформаційних технологій // Математична логіка та програмування. Досвід викладання. (колективна монографія), Одеса, Видавничий дім "Гельветика", 2022. - С.195-205.
6. Омельчук Л.Л., Ткаченко О. М., Шишацька О. В. Інтеграція навчальних курсів на основі проєктного підходу та гнучких методологій та гнучких методологій управління у 2022-2023 у навчальному році // Програмування: теорія та практика. Збірник матеріалів за результатами ІТ-проєкту міждисциплінарної інтеграції. 2022-2023 навчальний рік. / За редакцією Л.Омельчук, О.Ткаченка, О. Шишацької. – Одеса: Видавничий дім "Гельветика", 2023. С.7-14.
7. Програмування: теорія і практика. Збірник матеріалів за результатами ІТ-проєкту міждисциплінарної інтеграції. 2020-2021 навчальний рік / За редакцією Л.Омельчук, О.Ткаченка, О. Шишацької. – Одеса: Видавничий дім "Гельветика", 2021. 161 с.
8. Програмування: теорія та практика. Збірник матеріалів за результатами ІТ-проєкту міждисциплінарної інтеграції / За редакцією Л.Л. Омельчук, О.М. Ткаченка, О.В. Шишацької. – Одеса: Видавничий дім "Гельветика", 2022. – 155 с.
9. Програмування: теорія та практика. Збірник матеріалів за результатами ІТ-проєкту міждисциплінарної інтеграції / За редакцією Л.Л. Омельчук, О.М. Ткаченка, О.В. Шишацької. – Одеса: Видавничий дім "Гельветика", 2023. – 213 с.

ОНЛАЙН-ПЛАТФОРМА ДЛЯ ОБМІНУ ІНФОРМАЦІЇ МІЖ ПРИТУЛКАМИ ДЛЯ ТВАРИН ТА ВОЛОНТЕРАМИ

Артем Підгорний, Марія Кардаш, Данило Спіцин, Софія Яковишена, Олексій Степанюк, Олександра Волошина

Дана робота описує процес створення та роботи над платформою takeapet.me. Метою проєкту є спрощення процесу адопції тварин, підвищення ефективності комунікації між притулками та волонтерами, а також підвищення шансів тварин на знаходження нового дому. Робота описує повний цикл розробки продукту, починаючи з ініціації ідеї і закінчуючи фінальною реалізацією. Описано використання сучасних технологій та інструментів для аналізу, проєктування, розробки та тестування системи. Крім технічних аспектів, робота також розглядає методологію та процес організації роботи над проєктом в команді.

This work describes the process of developing the Takeapet.me online platform, which aims to simplify the process of animal adoption, improve communication between animal shelters and volunteers, and increase the chances of homeless animals finding new homes. It covers the entire product development cycle from the initial conception of an idea to its final implementation. The document describes the use of modern technologies and tools for analysis, design, development, and testing of the product. It describes how these tools are used to create a reliable and efficient system that can meet the needs of both animal shelters and potential adopters. In addition to technical aspects, the work describes the methodology and organization of project teamwork. It discusses management metodologies and teamwork tools that were important for the effective management of the project life cycle.

МОБІЛЬНИЙ ЗАСТОСУНОК, АДОПЦІЯ ТВАРИН, КОМУНІКАЦІЯ, РОЗРОБКА ВЕБПЛАТФОРМИ, УПРАВЛІННЯ ПРОЄКТОМ, КОМАНДНА РОБОТА, ЦИКЛ РОЗРОБКИ ПРОДУКТУ, ОПТИМІЗАЦІЯ СИСТЕМИ.

MOBILE APPLICATION, ANIMAL ADOPTION, COMMUNICATION, WEB PLATFORM DEVELOPMENT, PROJECT MANAGEMENT, TEAMWORK, PRODUCT DEVELOPMENT CYCLE, SYSTEM OPTIMIZATION.

ВСТУП

Постановка задачі. Оцінка сучасного стану об’єкта дослідження або розробки. Сучасний стан онлайн-платформ для взаємодії між притулками для тварин та волонтерами вимагає поліпшення. Хоча існують різноманітні ресурси для підтримки тварин, часто вони не забезпечують достатньої зручності, функціональності та взаємодії між зацікавленими сторонами.

Актуальність роботи. Актуальність проєкту обумовлена необхідністю покращення взаємодії між притулками та волонтерами. В сучасному світі, де цифрові технології пропонують нові можливості для вирішення соціальних проблем, важливо створити більш ефективний, інтуїтивний та доступний інструмент для допомоги тваринам.

Методи дослідження. Методи дослідження включають аналіз існуючих рішень, вивчення потреб користувачів – як притулків, так і волонтерів. Засоби розроблення включають сучасні вебтехнології та інструменти дизайну, зокрема використання мови програмування Python, вебфреймворку FastAPI для бекенду, системи управління базами даних PostgreSQL для забезпечення надійного зберігання та обробки даних, а також React для розробки фронтенду.

Мета та завдання роботи. Метою цього проєкту є створення функціональної онлайн-платформи, яка полегшує процес адопції тварин. Також, метою роботи є закріплення і поглиблення знань в області розробки вебплатформ, отримання практичного досвіду у комунікації з потенційними користувачами, ознайомитись на практиці з життєвим циклом продукту від ідеї до її втілення та вдосконалити навички роботи в команді та управління проєктами. Завдання включають розробку зручного інтерфейсу, забезпечення безпеки даних та інтеграцію інструментів для ефективної комунікації між притулками та потенційними волонтерами.

ПОШУК ТА ВАЛІДАЦІЯ ІДЕЇ

Опис пошуку ідеї продукту

На етапі пошуку ідей продукту, команда вибирала між трьома потенційними напрямками для розробки. Було проведено аналіз наступних ідей:

- **Реалізація системи електронного голосування на основі смарт-контрактів zero-knowledge proofs.** Перша ідея, яку розглядали, була спрямована на створення електронного способу голосування з використанням смарт-контрактів та zero-knowledge proofs. Цей підхід дозволяє автоматизувати процес голосування, забезпечуючи його анонімність та безпеку. Процес передбачав реєстрацію виборців з використанням ідентифікаційних документів та електронне подання голосів, які потім анонімно підраховувалися. Головною причиною для того щоб відкинути цю ідею стали потенційні складнощі з юридичної та політичної точок зору, які могли виникнути при впровадженні системи.
- **Написання децентралзованого криптовалютного гаманця.** Другою ідеєю було створення ієрархічного детермінованого гаманця для криптовалют, здатного зберігати, отримувати та відправляти кошти. Ця ідея базувалася на генерації та управлінні ключами за допомогою деревоподібної структури, забезпечуючи безпеку та ефективність управління криптовалютними активами. Однак, після аналізу, було вирішено відмовитися від цієї ідеї, оскільки виявили суттєві ризики, зокрема ризик втрати коштів користувачами через можливі помилки в реалізації та ризик великої конкуренції на ринку криптовалютних гаманців, що ускладнювало б просування нашого продукту
- **Платформа для взаємодії притулків для тварин та волонтерів.** Команда вирішила обрати третю ідею – розробку платформи для прихистку тварин Takeapet.me. Ідея сподобалась усім членам команди і була визнана як чудова можливість застосувати наші технічні знання для реалізації проєкту з соціальним значенням. Вона передбачала створення централізованої платформи, де притулки могли б легко розміщувати інформацію про тварин, що чекають на адопцію, а волонтери та потенційні власники – з легкістю переглядати цих тварин і подавати заявки на адопцію.

Огляд наявних на ринку IT-продуктів – конкурентів.

При створенні проєкту, метою якого є допомога тваринам, традиційне розуміння конкуренції відходить на другий план. Адже всі ініціативи в такому напрямку діють заради спільної мети - зменшення кількості безпритульних тварин. Проте, для ефективності проєкту прагнули зрозуміти, як працюють інші схожі вебресурси. Цей огляд допоміг порівняти функціонал, виділити переваги та недоліки, розповсюджені помилки, щоб не допустити їх у проєкті. Наш підхід до визначення потенційних конкурентів базувався на інтернет-дослідженні та спілкуванні з притулками.

Look4paws [1] підтримується компанією Kormotech і має за мету з'єднувати потенційних власників із чотирилапими друзями. Хоча основний акцент робиться на соціальному впливі, сайт також надає можливість подання онлайн-заявок на прихистку

тварин. Однак, обмеженість у функціоналі та відсутність внутрішньої форми для розміщення заявок може створити незручності для користувачів.

Adopt.ua [2], створений волонтерами, спрощує процес пошуку безпритульних тварин через оголошення. Сайт надає корисну інформацію та постійно оновлюється новими профілями тварин. Основним недоліком є відсутність форми для розміщення заявок, що змушує користувача телефонувати в притулок при бажанні прихистити тварину.

Happy Paw [3], як благодійний фонд, має широкий спектр розділів і забезпечує глибоке розуміння своєї місії через інформативний контент. Фонд має стабільне фінансування завдяки співпраці з великими компаніями і пожертвам. Незважаючи на сучасний дизайн та зручність навігації, сайт також зіштовхується з проблемою відсутності онлайн-форм для безпосереднього прихисту тварин, вимагаючи телефонного звернення.

Мапа продукту.

- **Проблема.** На сучасному ринку відсутній уніфікований вебсервіс у форматі "маркетплейсу", де будь-який притулок міг би створювати свій акаунт, зручно взаємодіяти з заявками на прихисток, а користувачі мали можливість переглядати різноманітні притулки, обирати тварин відповідно до своїх уподобань та подавати заявки на прилаштування. Існуючі рішення зазвичай пропонують лише інформацію про конкретний притулок і приймають заявки через неінтегровані гугл-форми, що робить процес менш зручним та інтуїтивним для користувачів.
- **Сегменти клієнтів:**

1. Притулки для тварин, які шукають платформу для представлення своїх мешканців.
2. Особи, які бажають прихистити тварину.
3. Компанії та бренди, які зацікавлені в рекламі на платформі, пов'язаній з доглядом за тваринами (або будь-якою іншою рекламою).

- **Унікальна ціннісна пропозиція.** Це буде перша у своєму роді платформа, де будь-який притулок може зареєструватися та представити своїх мешканців, забезпечуючи їм більші шанси на знаходження дому. Було проведено опитування існуючих притулків для тварин в соціальній мережі Instagram на тему того, чи скористались би вони вебсервісом, який ми пропонуємо та який функціонал хотіли б на ньому бачити. З відповідей притулків ми зробили висновок, що наша ідея є актуальною та такий сайт дійсно потрібен і буде користуватись попитом.
- **Рішення.** З нашим сайтом притулки отримають можливість створення власного акаунту, додавання профілів тварин і зручного перегляду та обробки заявок на прихисток. З іншого боку, користувачі будуть мати доступ до платформи, де вони можуть легко переглядати тварин та подавати заявки на прилаштування через інтегровану систему, відкидаючи необхідність користування сторонніми формами.
- **Джерела доходів.** Оскільки проєкт має волонтерську основу, основним джерелом доходу буде реклама. Компанії, що спеціалізуються на товарах та послугах для тварин, можуть розміщувати свою рекламу на сайті, використовуючи його як цільову платформу для своєї аудиторії.
- Ключові метрики:

1. Кількість зареєстрованих притулків.
2. Кількість тварин, доданих на платформу.
3. Кількість успішних прихистків через платформу.
4. Кількість активних користувачів сайту (щоденно, щотижнево, щомісячно).
5. Кількість переглядів реклами та кліків по рекламних оголошеннях.

Сильні і слабкі сторони розроблюваного продукту.

Сильні сторони:

1. Унікальність: Перша в своєму роді платформа, яка дозволяє притулкам створювати власні акаунти та представляти тварин.

2. Підтримка спільноти: Позитивні відгуки від притулків, що підтверджують актуальність та потребу у платформі, а також просування культури адопції тварин замість купівлі.

3. Зручність: Інтуїтивно зрозумілий інтерфейс для користувачів та притулків, що полегшує процес подання та обробки заявок.

Слабкі сторони:

1. Відсутність первинного фінансування: Проєкт має волонтерську основу, що може уповільнити розвиток та впровадження нових функцій.

2. Обмеженість рекламних можливостей: Залежність від реклами як основного джерела доходу може обмежити фінансову стабільність.

Можливості:

1. Зростаюча потреба в цифрових рішеннях для будь-яких сфер.

2. Розширення функціоналу для залучення більшої аудиторії (наприклад, інтеграція з соціальними мережами).

3. Партнерства з благодійними фондами та ветеринарними клініками.

Загрози.

Основна загроза полягає в залежності від волонтерів. Оскільки проєкт базується на волонтерській основі, існує ризик нестабільності у підтримці та розвитку платформи в довгостроковій перспективі.

Огляд використаних технологій. Розглянемо обрані засоби для програмної реалізації онлай платформи для взаємодії притулків та користувачів.

Python [4] є високорівневою інтерпретованою мовою програмування, яка володіє строгою динамічною типізацією. Її простий синтаксис і гнучкість дозволяють використовувати Python не тільки в різних областях розробки, але й у автоматизації повсякденних задач. У нашому проєкті було використано Python для розробки всієї бек-енд частини вебзастосунку, зокрема, було використано вебфреймворк FastAPI [5].

FastAPI – це сучасний, високопродуктивний вебфреймворк для створення API використовуючи Python 3.8+, в основі якого лежить стандартна анотація типів Python. Цей фреймворк ідеально підходить для створення високошвидкісних API, оскільки він пропонує автоматичну генерацію документації, валідацію запитів і відповідей, а також підтримує асинхронне програмування.

PostgreSQL [6] – це потужна відкрита реляційна база даних, що підтримує як SQL (реляційне), так і JSON (нереляційне) запити. Вона використовується у нашому проєкті для забезпечення надійного зберігання даних, включаючи інформацію про тварин та користувачів.

React [7] – це декларативна, ефективна та гнучка JavaScript бібліотека для побудови інтерфейсів користувача. У цьому проєкті React використовується для створення фронтенду. Його компонентний підхід дозволяє легко створювати взаємодіючі UI елементи, що забезпечує користувачам зручний досвід перегляду та взаємодії з платформою.

ОПИС ПРОГРАМНОГО ПРОДУКТУ

Призначення та цілі створення онлайн-платформи.

Онлайн-платформа призначена для спрощення та оптимізації взаємодії між притулками для тварин і волонтерами. Основна мета полягає у створенні централізованого онлайн простору, де притулки можуть ефективно розміщувати

інформацію про тварин, що чекають на адопцію, а волонтери мають змогу легко переглядати цих тварин і подавати заявки на адопцію. Платформа дозволяє користувачам створювати особисті акаунти, як для притулків, так і для волонтерів, та надає можливість переглядати сайт без авторизації, що забезпечує широку доступність інформації.

Вимоги до онлайн-платформи.

Платформа розробляється для збору, обробки та аналізу інформації про притулки та тварин з метою полегшення процесів реєстрації, входу, та взаємодії між притулками та користувачами. Це включає можливість перегляду тварин, управління заявками на прихисток, та аналіз статусу цих заявок. Призначена для оптимізації взаємодії між користувачами, що бажають прихистити тварину, та притулками, платформа забезпечує ефективне та зручне використання, сприяючи швидшому знаходженню дому для тварин. На рис. 1 наведена Use Case діаграма, яка відображає шляхи взаємодії користувачів з системою та забезпечує розуміння функціоналу вебплатформи.

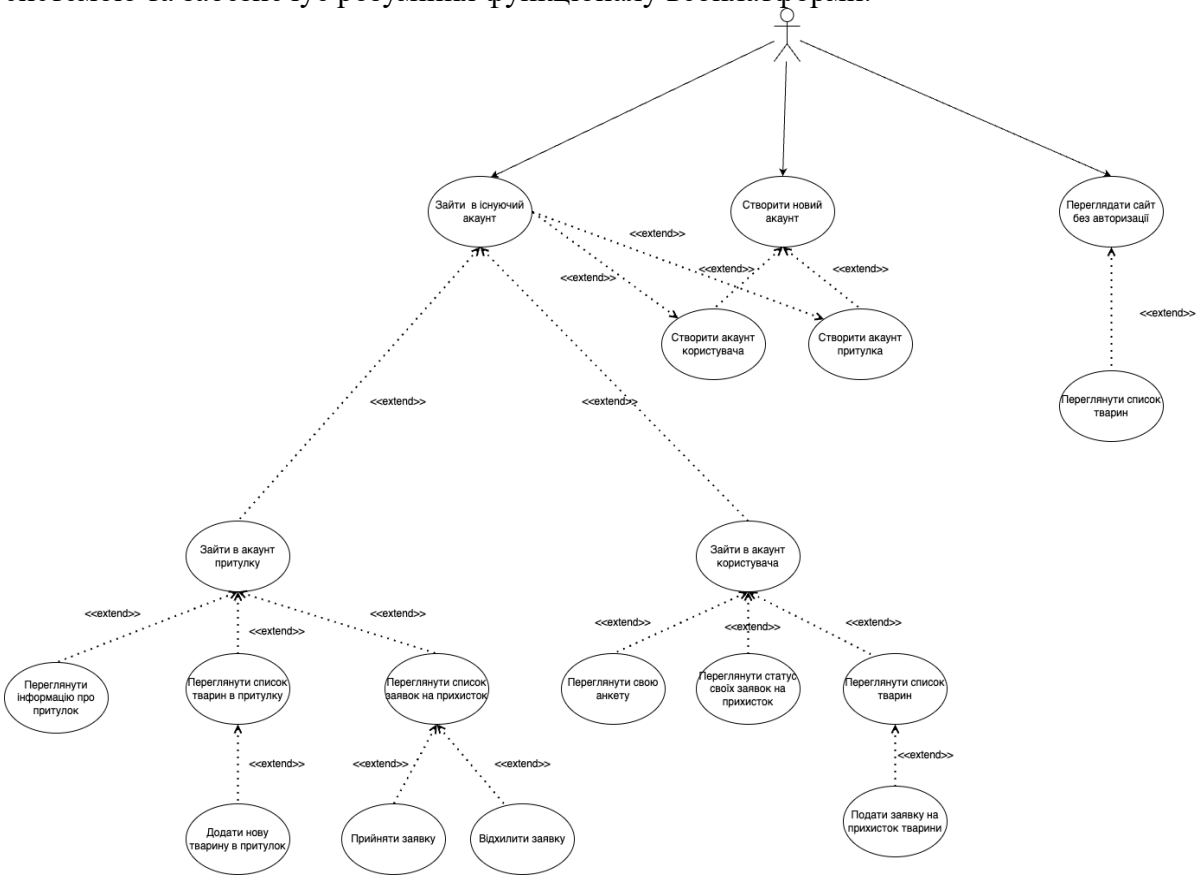


Рисунок 1 – Use Case діаграма

Вимоги до онлайн-платформи в цілому.

Вимоги до структури та функціонування.

Інтерактивність і доступність: розробка має бути легко доступною та зрозумілою для користувачів з різним рівнем технічних навичок. Інтерфейс повинен бути інтуїтивним та користувацьки привітним. Вебсайт доступний за посиланням takeapet.me і має бути доступний з будь-якого пристрою та браузера.

Безпека та конфіденційність: важливо забезпечити високий рівень безпеки персональних даних користувачів та притулків, включаючи захист від несанкціонованого доступу.

Функціональність: розробка має включати функції для реєстрації, подачі заявок на адопцію тварин, управління профілями тварин та притулків.

Надійність та стабільність: розробка повинна функціонувати стабільно та безперебійно, забезпечуючи постійний доступ до своїх сервісів.

Адаптація до мобільних платформ: вебсайт має бути повністю оптимізований для мобільних пристроїв, забезпечуючи зручне користування на смартфонах та планшетах.

Вимоги до функцій, які виконуються онлайн-платформою.

Сайт takeapet.me повинен надавати надійний інтерфейс для співпраці притулків з потенційними власниками тварин, автоматизації процесів подачі та обробки заявок на прихисток.

В розробці передбачається виділити наступні функціональні підсистеми:

Підсистема притулку, для якої передбачений функціонал розміщення оголошень про тварин та прийняття або відхилення заявок від користувачів.

Підсистема користувача, орієнтована на забезпечення простого та інтуїтивного інтерфейсу для огляду профілів тварин та подачі заявок на прихисток. Ця підсистема також включає особистий кабінет користувача, де він може відстежувати статус своїх заявок.

Підсистема притулку.

Таблиця 1. Перелік функцій, задач для автоматизації підсистеми притулку

Функція	**Задача**
Реєстрація та авторизація	Можливість зареєструвати акаунт та зайти в нього
Взаємодія з тваринами	Перегляд тварин
	Додавання тварини
Взаємодія з заявками	Перегляд заявок
	Прийняття або відхилення заявки

Підсистема користувача.

Таблиця 2. Перелік функцій, задач для автоматизації підсистеми користувача

Функція	**Задача**
Реєстрація та авторизація	Можливість зареєструвати акаунт та зайти в нього
Взаємодія з тваринами	Перегляд всіх тварин
	Застосування фільтрів та пошукового поля для вибору тварини
Взаємодія з заявками	Подача заявки
	Перегляд статусу своїх заявок
Взаємодія з притулками	Перегляд всіх притулків
	Перегляд тварин в обраному притулку

Системні вимоги:

Операційна система: Android 6.0 або новіша.

Процесор: Мінімум 4-ядерний процесор.

Оперативна пам'ять: Мінімум 2 ГБ оперативної пам'яті.

Внутрішня пам'ять: Доступне місце для інсталяції додатку та збереження даних.

Доступ до Інтернету: Для завантаження мап, інформації про місця та оновлення даних.

GPS-модуль: Для визначення місцезнаходження користувача та побудови маршрутів.

Дозволи: Додаток може вимагати дозволів на доступ до місцезнаходження, камери, мікрофону та збереження даних на пристрої.

Сумісність з екранами: Пристрій з екраном роздільної здатності, зручним для користування.

Інтернет-з'єднання: Для завантаження карт, оновлення даних та обміну інформацією з сервером.

Актуальна версія додатку: Користувач повинен завжди оновлювати додаток до останньої версії для отримання нових функцій та покращень.

Опис організації інформаційної бази.

Логічна структура бази даних.

Під час розробки використовувалась СУБД PostgreSQL. У таблиці 3-7 описано основні бази даних та їх призначення в розробці.

Таблиця 3. Перелік баз даних онлійн-платформи

Номер	База даних	Опис
1	AnimalDB	База даних для збереження інформації про тварин
2	ApplicationDB	База даних для збереження інформації про заявки
3	ShelterDB	База даних для збереження інформації про притулки
4	UserDB	База даних для збереження інформації про користувачів

Таблиця 4. Опис атрибутів таблиці AnimalDB

Атрибути таблиці AnimalDB	Тип	Опис
ID	integer	Ідентифікатор
Name	string	Ім'я тварини
Type	string	Тип тварини (кіт/собака чи інший)
Sex	string	Стать тварини
Month	string	Вік тварини
Year	string	Вік тварини
Shelter_id	integer	Ідентифікатор притулку, в якому знаходиться тварина
Description	string	Опис тварини

Таблиця 5. Опис атрибутів таблиці ApplicationDB

Атрибути таблиці ApplicationDB	Тип	Опис
ID	integer	Ідентифікатор
Shelter_id	string	Ідентифікатор притулка в який йде заявка
User_id	string	Ідентифікатор користувача який створив заявку
Animal_id	string	Ідентифікатор тварини яку бажають прихистити
Status	enum	Статус заявки

Таблиця 6. Опис атрибутів таблиці ShelterDB

Атрибути таблиці ShelterDB	Тип	Опис
ID	integer	Ідентифікатор
Email	string	Пошта притулка
Name	string	Назва притулка
Password	string	Пароль
Salt	string	Випадково згенерована стрічка, яка додається до пароля перед його хешуванням
Address	string	Адреса притулку
Number	string	Номер телефону притулка
Description	string	Опис

Таблиця 7. Опис атрибутів таблиці UserDB

Атрибути таблиці UserDB	Тип	Опис
ID	integer	Ідентифікатор
Email	string	Пошта користувача
Full_name	string	Повне ім'я користувача
Password	string	Пароль
Salt	string	Випадково згенерована стрічка, яка додається до пароля перед його хешуванням
Photo	string	Фото
Description	string	Опис

UI/UX-дизайн продукту.

Закони та принципи з UX-дизайну, створюваного продукту.

1. Закон Міллера:

"Короткострокова людська пам’ять не може запам’ятати і повторити більше 7 ±2 елементів."

Мінімізація кількості основних пунктів навігаційного меню, щоб полегшити користувачам вибір та навігацію. В хедері присутні 4 елементи, що відокремлені в групах за критерієм подібності свого функціоналу (рис. 2).

Рисунок 2 – Приклад застосування закону Мілера для хедера

Мобільна версія передбачає приховання групи другорядних функцій за допомогою кнопки ☰ у висувному меню (рис. 3).

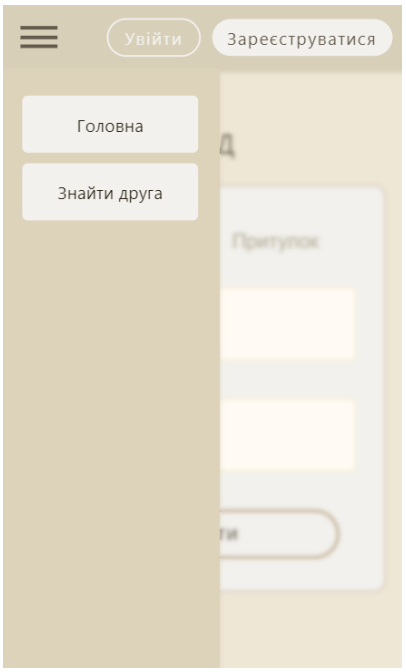


Рисунок 3 – Приклад застосування закону Мілера для розміщення навігаційних кнопок

- Логічна послідовність екранів взаємодії від одного завдання до іншого.
- Мінімізація переходів між сторінками від початкової до кінцевої точки алгоритму дій користувача. Прикладом є алгоритм додавання тварини притулком: Акаунт → Список тварин → Додати тварину → Заповнення форми додавання тварини

2. Закон Хіка:

"Час, необхідний для прийняття рішення, збільшується із збільшенням кількості елементів та складності вибору."

- Вивід на екран оптимальної кількості елементів списків, балансування розмірів.

На сторінці "Знайти друга" на рис. 4 виведено список тварин, яких можна прихистити. Елементи зображені по 4 в ряд, що не створює враження "загромадження" сторінки та дозволяє акцентувати увагу користувача на кожній тварині.

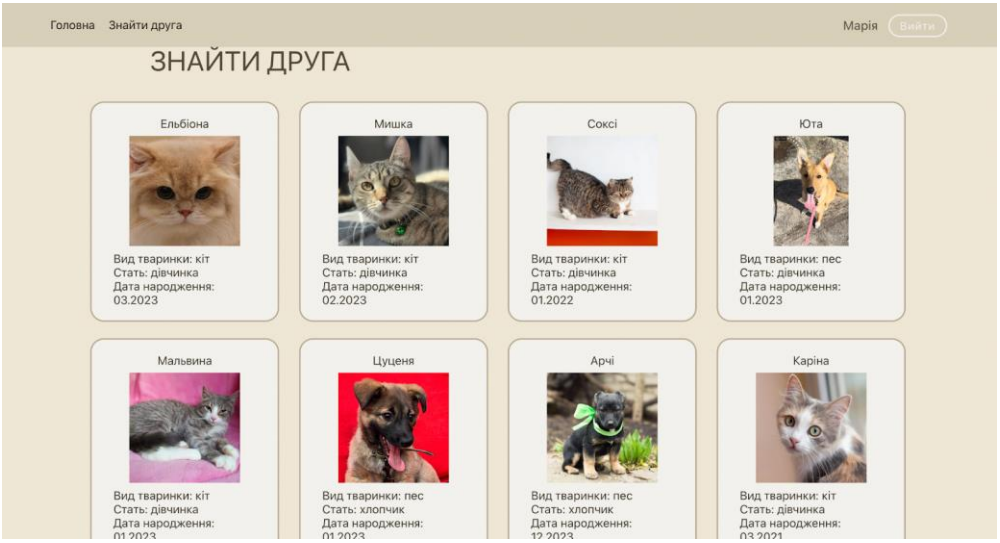


Рисунок 4 – Приклад застосування закону Хіка для виводу всіх тварин

● Вирівнювання тексту та правильно підібраний колір. Використана спокійна палітра, відповідна концепції сайту (рис. 5).

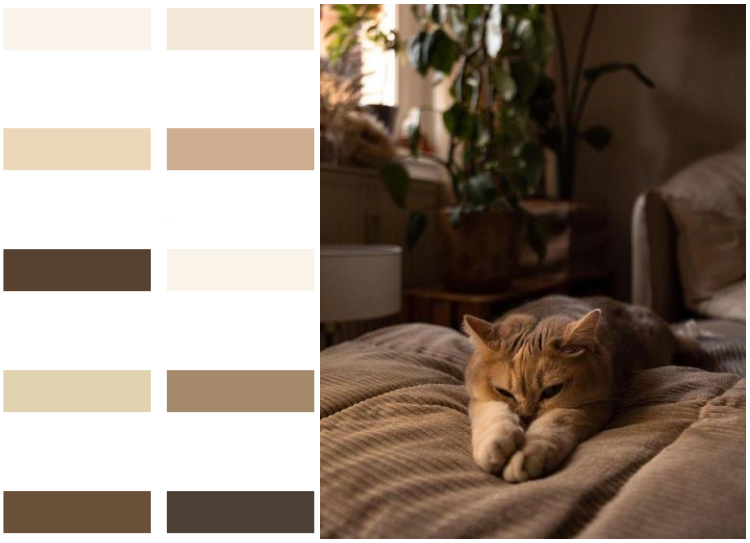


Рисунок 5 – Приклад застосування закону Хіка для вибору кольорової палітри сайту

● Дотримання яскравості і контрасту, щоб привернути увагу до критичних функцій.

● Уніфікований дизайн на всіх сторінках вебсайту.

● Закон Фіттса:

● *"Час, необхідний для швидкого переміщення до цільової області, залежить від відстані до цілі та ширини цілі."*

● Дотримання балансу між розміром важливих елементів та їхнім місцем розташування.

● Важливі функції помітні для користувача. На рис. 6 показано, що в акаунті притулку/користувача основний функціонал згруповано у виділений світлим кольором блок та розташовано по центру.

Рисунок 6 – Приклад застосування закону Фіттса для виділення основного функціоналу

4. Принцип "Бритва Оккама"

"Не слід використовувати більше речей, ніж необхідно":

- Використання мінімалізму під час створення дизайну інтерфейсу.
- Зменшення кількості тексту, графіки для покращення зосередженості користувача. Для переходу до акаунту (рис. 7) використовується кнопка з ім’ям користувача/притулку, що свідчить про зрозумілий на інтуїтивному рівні інтерфейс.

Рисунок 7 – Приклад застосування принципу "Бритва Оккама" для переходу до акаунту

- Зменшення кількості обов'язкових полів та етапів при реєстрації/вході. Наприклад, на рис. 8 бачимо, що для реєстрації необхідно лише обрати роль (користувач/притулок) та заповнити поля "Ім’я", "Ел. адреса" та "Пароль". Усю іншу інформацію можна вказати в акаунті за опцією "Редагувати профіль".

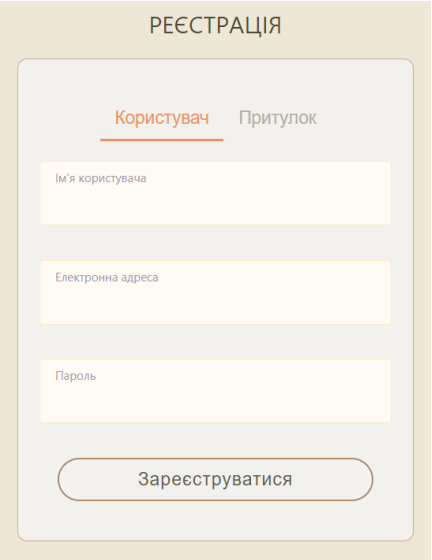


Рисунок 8 – Приклад застосування принципу "Бритва Оккама" для форми реєстрації/входу

ІМПЛЕМЕНТАЦІЯ ПРОДУКТУ

Серверна частина реалізована за допомогою мікросервісної архітектури, яка складається із дев'яти окремих сервісів:

- Сервіс користувачів – реалізує запити до БД користувачів;
- Сервіс притулків – реалізує запити до БД притулків;
- Сервіс тварин – реалізує запит до БД тварин;
- Сервіс заявок на прихисток – реалізує запит до БД заявок на прихисток;
- Сервіс бази даних користувачів – реалізує базу даних користувачів;
- Сервіс бази даних притулків – реалізує базу даних притулків;
- Сервіс бази даних тварин – реалізує базу даних тварин;
- Сервіс бази даних заявок на прихисток – реалізує базу даних заявок на прихисток;
- Сервіс nginx – реалізує доступ до усіх сервісів за одним портом 8080.

Імплементована архітектура має низку плюсів, таких, як:

- Гнучкість розробки та масштабування: Кожен сервіс може розвиватися, тестуватися, та розгортатися незалежно. Це сприяє більш швидким ітераціям розробки та здатності масштабувати окремі компоненти системи згідно з потребами.
- Легкість обслуговування та оновлення: Мікросервіси дозволяють вносити зміни або оновлення в окремі частини системи без необхідності перероблення всього додатка.
- Відмовостійкість: Якщо один мікросервіс зазнає збою, інші продовжують працювати, що зменшує ризик повного збою системи.

Оскільки одним із сервісів є nginx, то він може приймати запити для усіх мікросервісів одночасно, використовуючи один порт. Це реалізовано за допомогою його конфігурації зображеної на рис. 9.

nginx_config.conf

```
server {
  listen 8080;

  location /api/v1/users {
    proxy_pass http://user_service:8000/api/v1/users;
  }

  location /api/v1/animals {
    proxy_pass http://animal_service:8000/api/v1/animals;
  }

  location /api/v1/shelter {
    proxy_pass http://shelter_service:8000/api/v1/shelter;
  }

  location /api/v1/applications {
    proxy_pass http://application_service:8000/api/v1/applications;
  }
```

Рисунок 9 – Конфігураційний файл nginx

Кожен з мікросервісів реалізовано за допомогою Python FAST API фреймворку. FAST API відомий своєю високою продуктивністю та ефективністю, що дозволяє обробляти велику кількість запитів з мінімальною затримкою, що є критично важливим для мікросервісних архітектур. Цей фреймворк підтримує асинхронну обробку, що є важливою властивістю для ефективної роботи мікро-сервісів, оскільки вона дозволяє оптимізувати використання ресурсів і знизити час відгуку.

Розглянемо реалізацію мікросервісу User. На рис. 10 можна побачити, як виглядає загальна структура сервісу.

Рисунок 10 – Структура мікро-сервісу User

Розглянемо, як виглядає модель користувача для подальшої взаємодії із базою даних. Як можна побачити на рисунку 11, модель визначається максимально доступно за допомогою використаного фреймворку, встановлюються ключові поля в таблиці, тощо. Після цього, ця модель може доступно використовуватись у запитах, необхідних для коректної роботи сайту.

```
id = Column(Integer, primary_key=True, index=True, autoincrement=True)
email = Column(String, unique=True, index=True)
full_name = Column(String)
password = Column(String)
salt = Column(String)
photo = Column(String, nullable=True)
description =Column(String, nullable=True)
```

Рисунок 11 – Модель користувача

Для прикладу, розглянемо реалізацію інтерфейсу сервісу користувача (рис. 12), який додає гнучкості в подальшій розробці та визначає, які запити у ньому доступні.

```
class UserServiceInterface:
    def register_user(self, user_local:UserLocalRegistration) -> bool:
        pass

    def authorize_user(self, user_local:UserLocalAuthorization):
        pass

    def get_user(self, user_id: int) -> list[UserLocalOtput]:
        pass

    def is_user_exists(self, user_id: int) -> bool:
        pass
```

Рисунок 12 –Інтерфейс сервісу користувача

Вебсайт, або частина клієнта.

Загалом, ця частина застосунку була реалізована за допомогою React. Обрання React для реалізації вебсайту частини клієнта принесло кілька ключових переваг:

Компонентний підхід: React дозволяє створювати вебзастосунки, розділяючи інтерфейс на перевикористовувані компоненти. Це сприяє більш організованому та легшому для розуміння коду, а також полегшує тестування та обслуговування.

Широке співтовариство та підтримка: React є одним з найпопулярніших фреймворків для розробки вебзастосунків, що забезпечує доступ до великої кількості ресурсів, бібліотек і готових рішень.

Ефективність та швидкість: React оптимізує відображення сторінок через використання віртуального DOM, що покращує продуктивність застосунку, особливо при великій кількості динамічних змін у інтерфейсі.

Гнучкість та сумісність: React може легко інтегруватися з іншими бібліотеками та фреймворками, що дозволяє створювати складні додатки, використовуючи найкращі доступні інструменти.

Покращена користувацька взаємодія: React дозволяє легко створювати інтерактивні та відгукові користувацькі інтерфейси, підвищуючи задоволення користувачів.

Готовність до масштабування: Фреймворк підходить як для невеликих, так і для великих масштабних застосунків, дозволяючи рости проєкту без необхідності зміни технологічного стеку.

Розглянемо загальну структуру застосунку на рис. 13.

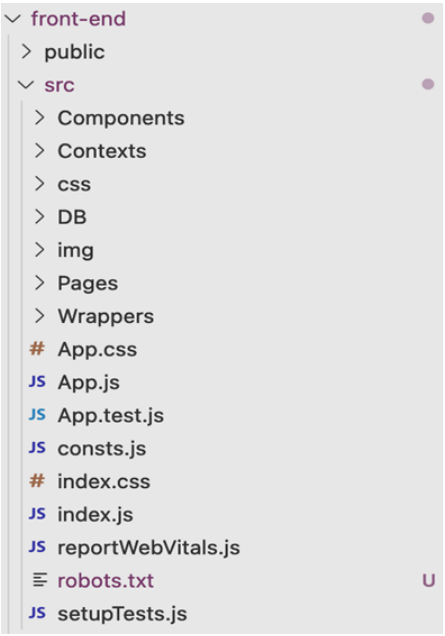


Рисунок 13 – Структура мікро-сервісу додатку

Розглянемо файл App.js, який є одним з основних файлів у структурі React-додатку. Він виступає як кореневий компонент, з якого починається рендеринг додатку в DOM. Цей файл відповідає за маршрутизацію в додатку та визначає, який компонент відображати відповідно до URL-адреси, яку відвідує користувач.

На рис. 14 ми бачимо код, що визначає роутинг у додатку за допомогою компоненту <Routes> з бібліотеки React Router. Компонент <Routes> містить декілька компонентів <Route>, кожен з яких відповідає за певний шлях (path) і рендерить відповідний компонент (element). Наприклад, коли користувач переходить на адресу '/login', рендериться компонент <Login />. Таким чином, App.js керує основною навігацією та організацією сторінок у додатку.

```
function App() {
  return (

      <div>
      <Header />
      <Routes>
        <Route path='/' element={<Home />} />
        <Route path='/donate' element={<Donate />} />
        <Route path='/contacts' element={<Contacts />} />
        <Route path='/login' element={<Login />} />
        <Route path='/signup' element={<Signup />} />
        <Route path='/shelter-account' element={<ShelterAcc />} />
        <Route path="/animal/:animalId" element={<Animal />} />
        <Route path="/animal-main" element={<AnimalMain />} />
      </Routes>
      </div>

  );
}
```

Рисунок 14 – Керування навігацією у додатку

Розглянемо, як виглядає запит на реєстрацію користувача. Як можна побачити на рис. 15 та 16, за допомогою axios клієнтська частина відправляє запит у на серверну

частину за API URL, і отримує відповідь, обробляючи її у двох випадках – успішної або неуспішної реєстрації.

Отже, за подібним принципом пов’язані і усі інші запити, такі, як авторизація, додавання тварини, зміна статусу заявки на прихисток, та інші.

```
class Signup extends Component {
    constructor(props) {
        super(props);
        this.state = {
            showUserReg: true,
            showShelterReg: false,
            registrationPath: '/user-account',
            name: '',
            email: '',
            password: ''
        };
    }
```

Рисунок 15 – Параметри для реєстрації у класі

```
const SIGNUP_API_URL = `http://127.0.0.1:8080/api/v1/${entity}/signup`;

axios.post(SIGNUP_API_URL, entityData).then(response => {
    console.log('Реєстрацію пройдено успішно:', response.data);
    this.props.navigate("/login");
}).catch(error => {
    console.error('Користувач із такою електронною поштою вже існує.', error.message);
});
```

Рисунок 16 – Процес відправки запиту і отримання відповіді

ТЕСТУВАННЯ

Проведене тестування онлайн-платформи виявилося ефективним у перевірці різних аспектів функціональності модулю UserService. Зокрема, було успішно виконано тести, які перевіряли процес реєстрації та авторизації користувачів, а також отримання даних про користувачів. Один із ключових тестів, test_register_user_successful, перевіряв здатність системи коректно реєструвати нових користувачів, використовуючи мок-об'єкти для імітації різних сценаріїв: успішної реєстрації та невдалого додавання до бази даних. Такий підхід дозволяє забезпечити відповідність реалізації до очікуваної поведінки методу, зокрема підтвердження успішної реєстрації поверненням значення True та невдалої – False.

Інші тести, такі як test_authorize_user та test_get_user, оцінювали здатність системи перевіряти облікові дані користувачів та коректно обробляти запити на отримання даних про користувачів. Тести на обробку виняткових ситуацій, такі як відсутність користувача у базі або помилкові облікові дані, виявилися особливо важливими, підкреслюючи здатність системи обробляти помилкм та виняткові стани, наприклад, через використання HTTPException з відповідними кодами статусу для кожного сценарію. В результаті цих тестів з'ясувалося, що система здатна ефективно реагувати на зміни та вимоги користувачів, забезпечуючи надійність і точність у обробці запитів на різних рівнях її функціональності.

Результати виконання тестів зображені на рис. 17 та рис. 18.

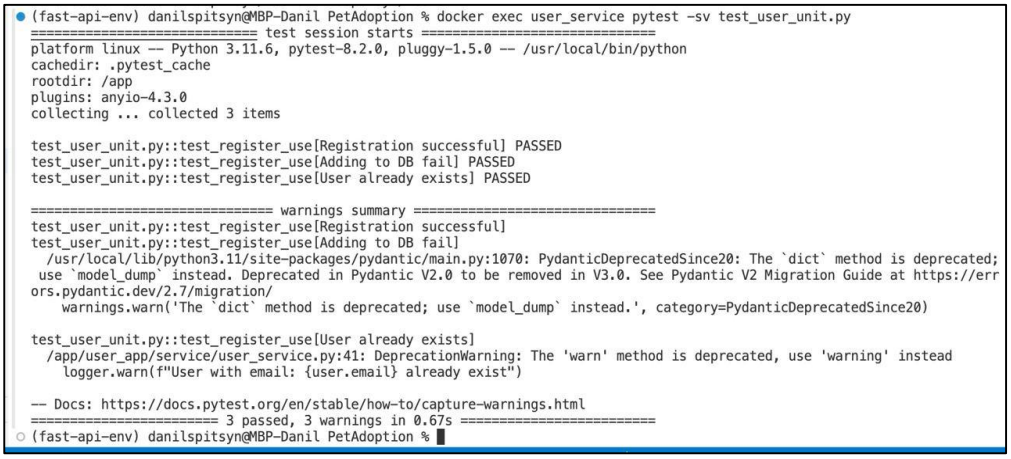


Рисунок 17 – Успішне проходження тестів для модулю UserService

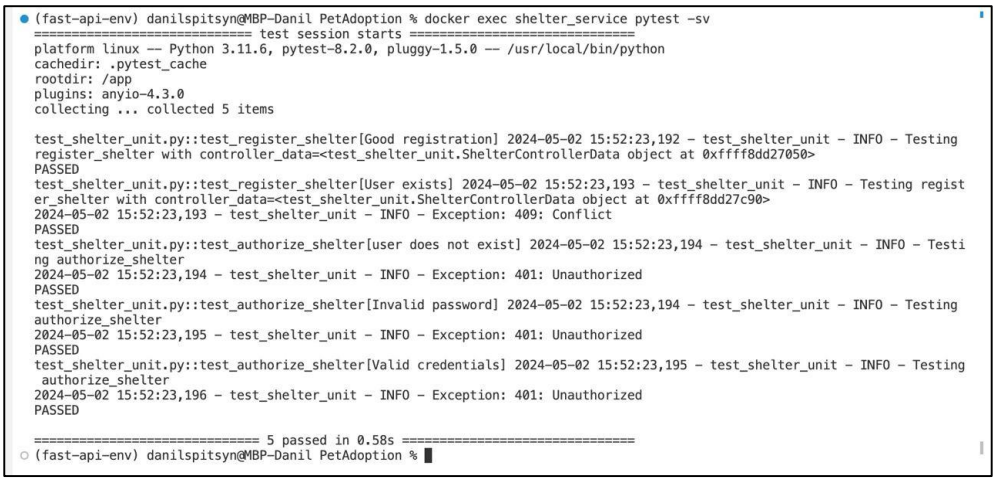


Рисунок 18 – Успішне проходження тестів для модулю ShelterService

Тести збірки (поєднання модулів).

Також було виконано тести збірки, які включали кілька ключових етапів. Основний процес збірки, під назвою "build", забезпечує перевірку правильності компіляції всього проєкту, готуючи його до детальніших тестів. Тести було успішно пройдені (рис. 19).

Рисунок 19 - Успішне проходження тестів збірки

ДОКУМЕНТАЦІЯ ПРОЄКТУ

Обрана модель та інструменти ведення документації.

Для ефективного ведення внутрішньої документації нашої команди, ми вирішили використовувати Trello, створивши окрему папку для документації (рис. 20). Цей підхід дозволив нам централізувати всі важливі матеріали, включаючи посилання, презентації та іншу необхідну інформацію. Це забезпечило можливість для кожного члену команди, включаючи потенційних нових учасників, легко та швидко зрозуміти суть та деталі проєкту, над яким ми працюємо. Використання Trello сприяло покращенню організації та доступності важливої інформації, забезпечуючи ефективну координацію та співпрацю всередині команди.

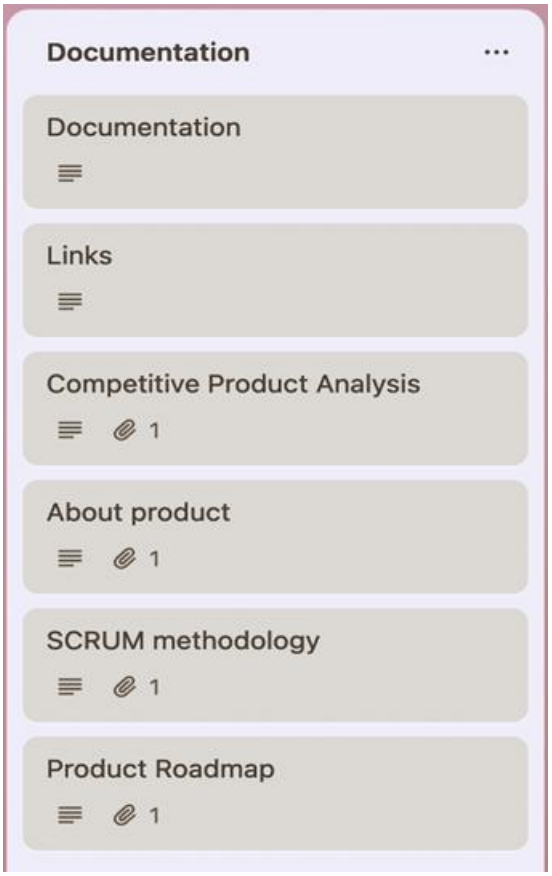


Рисунок 20 – Ведення внутрішньої документації в Trello

Інструкція користувача.

Спочатку, користувач або притулок має зареєструватись (рис. 21).

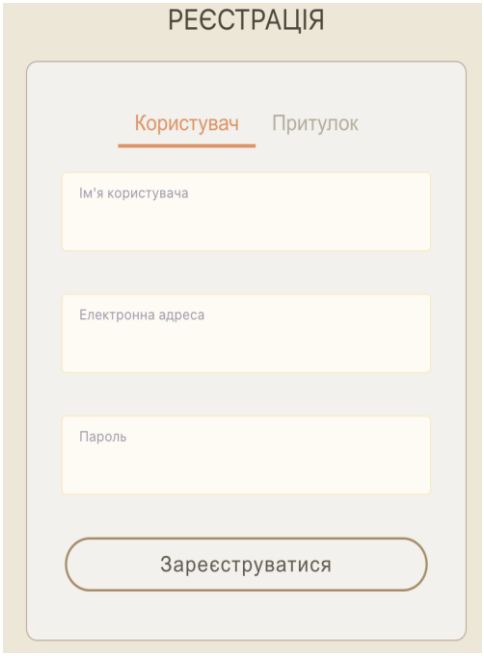


Рисунок 21 – Реєстрація користувача або притулку

Після чого, користувач має зайти в свій акаунт, щоб отримати повний доступ до його функцій (рис. 22).

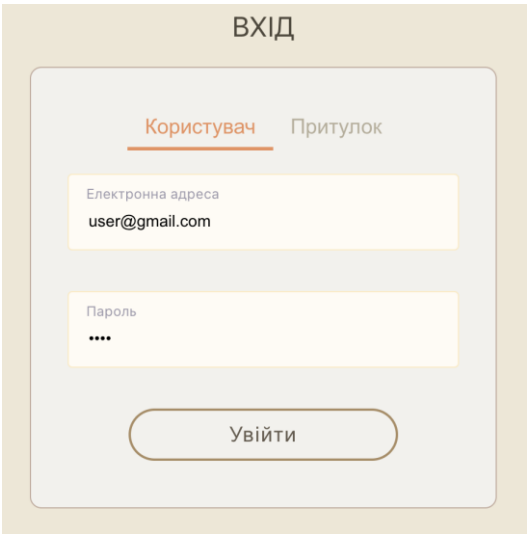


Рисунок 22 – Авторизація користувача або притулку

Після авторизації, користувачу автоматично доступна сторінка, з якої він може переглянути всіх тварин (рис. 23 та рис. 24).

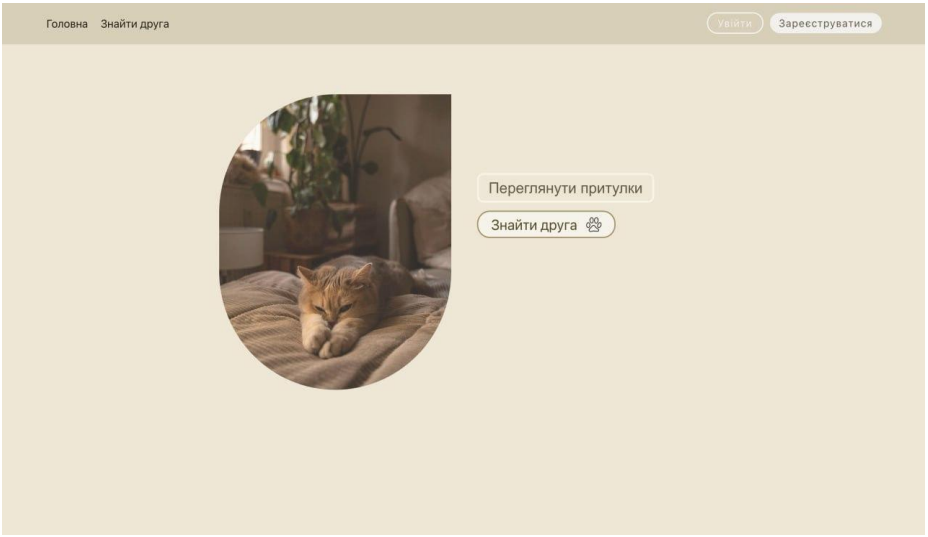


Рисунок 23 – Основна сторінка

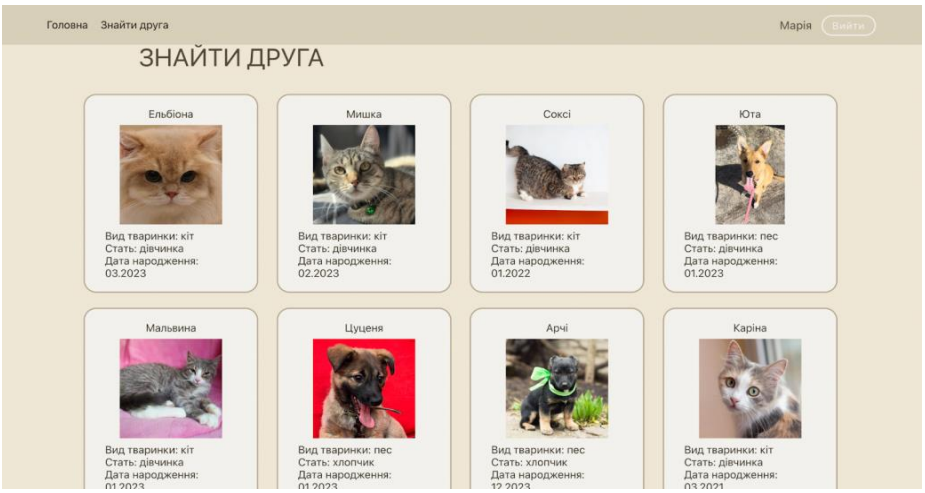


Рисунок 24 – Сторінка, де відображаються усі тварини

Користувач може натиснути на тварину, яка йому сподобалась, в результаті чого відкриється меню з детальною інформацією про неї та можливістю подати заявку на прихисток (рис. 25). Подана заявка буде направлена притулку, а в "Заявках на прихисток " в акаунті користувача з'являться його заявки з статусами (рис. 26).

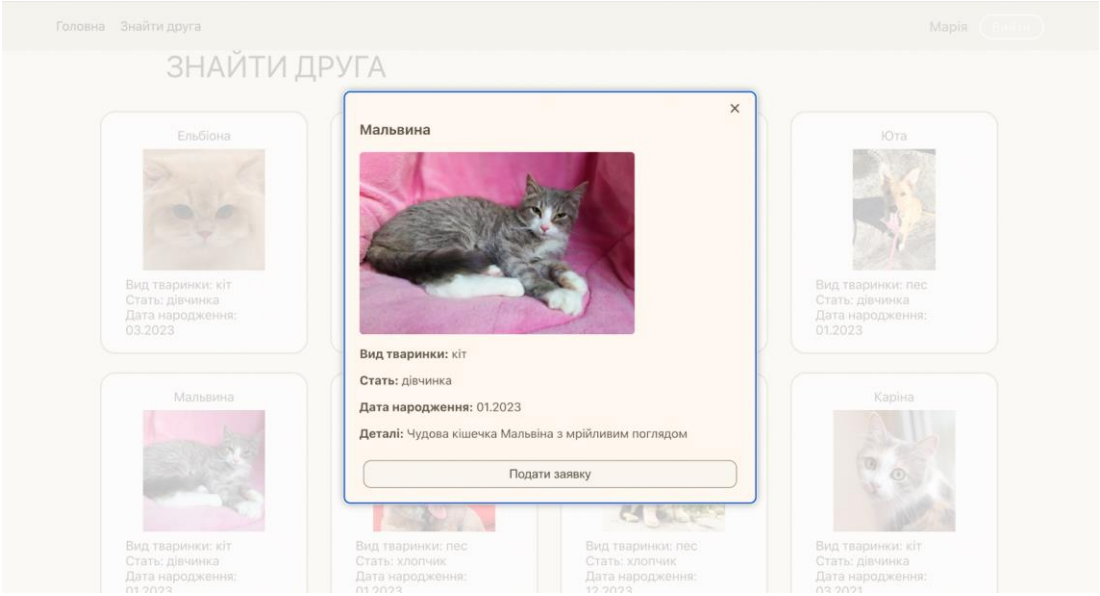


Рисунок 25 – Детальна інформація про тварину

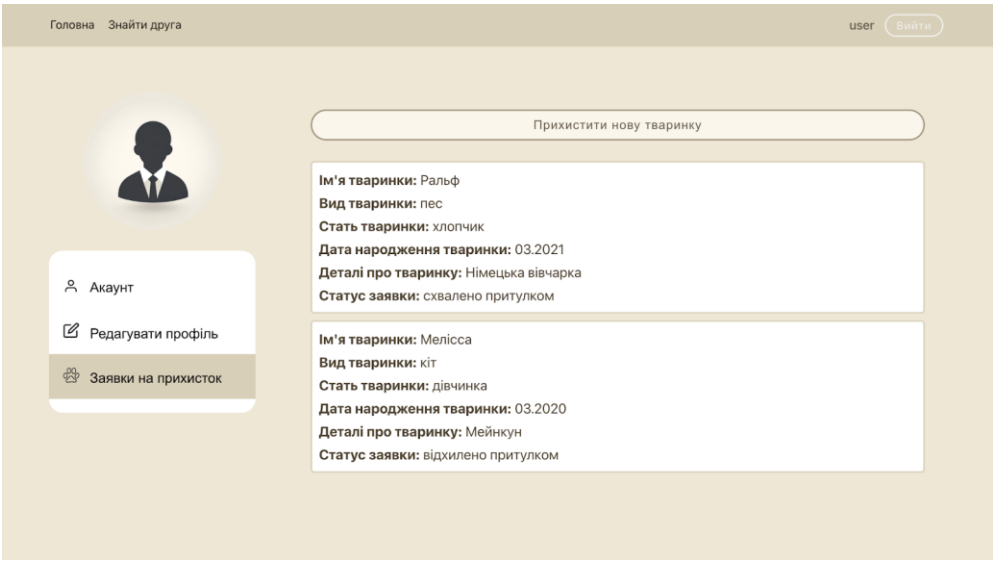


Рисунок 26 – Сторінка з заявками користувача

Притулок же, у свою чергу, після авторизації, може перейти в свій акаунт, натиснувши свій нікнейм праворуч зверху, та обрати сторінку "Список заявок " (рис. 27).

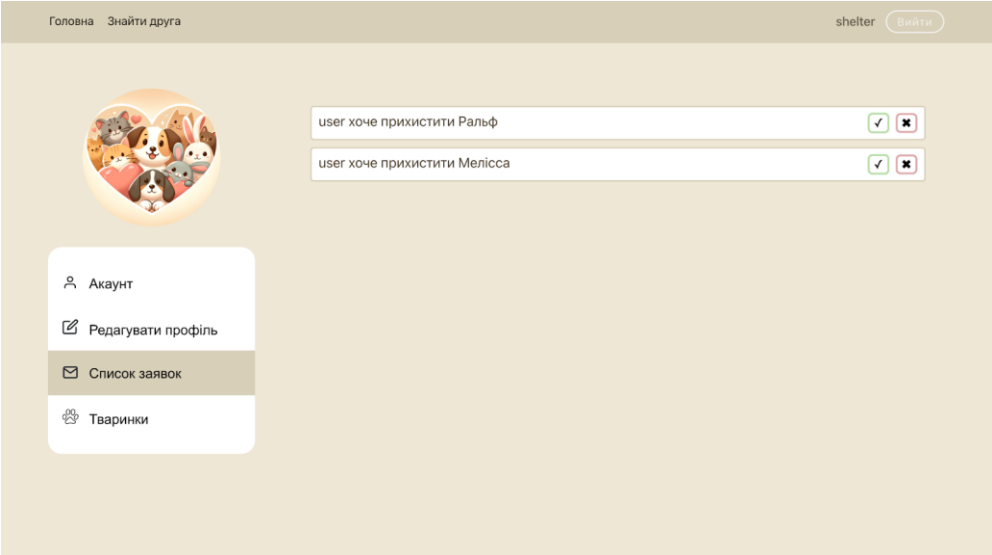


Рисунок 27 – Список заявок в профілі притулку

Після прийняття або відхилення заявки притулок побачить відповідний напис у себе в профілі (рис. 28 та 29).

Рисунок 28 – Прийняття заявки притулком

Рисунок 29 – Відхилення заявки притулком

Правові норми, які регулюють відносини в сфері застосування програмного продукту.

На діяльність, пов'язану з використанням нашого програмного продукту, впливають декілька ключових законів України, які забезпечують правовий регламент у сферах захисту тварин, волонтерської діяльності та відповідальності за жорстоке поводження з тваринами. Закон України "Про захист тварин від жорстокого поводження" [8] встановлює загальні принципи гуманного поводження з тваринами, забороняє їх жорстоке обіймання, мучення та інші дії, що можуть заподіяти тваринам біль, страждання або стрес. Цей закон визначає обов’язки організацій та осіб, які тримають тварин, і ставить під контроль умови їхнього утримання. Закон України "Про волонтерську діяльність" [9] регламентує діяльність волонтерів та волонтерських організацій, надає правові рамки для залучення громадськості до допомоги у соціально значущих проєктах, включаючи захист та підтримку тварин. Волонтерська робота через нашу платформу має відповідати вимогам цього закону, забезпечуючи права та обов’язки волонтерів. Стаття 299 Кримінального кодексу України (Жорстоке поводження з тваринами) [9] передбачає кримінальну відповідальність за жорстоке поводження з тваринами, що спричинило їхню смерть або інші серйозні наслідки. Ця стаття акцентує на необхідності відповідального ставлення до тварин і важливості дотримання етичних норм у поводженні з ними. Стаття 154 Кодексу про адміністративні

порушення [10] встановлює правила тримання домашніх тварин, зокрема собак і котів, і передбачає адміністративну відповідальність за їх порушення. Ця стаття регулює умови утримання домашніх тварин і зобов'язує власників забезпечувати належний догляд.

ОПИС ПРОЦЕСУ РОЗРОБКИ

Опис команди та обраних методології, інструментарію.

Команда складається з п'яти осіб. Склад команди та розподіл ролей можна переглянути в табл. 8.

Таблиця 8. Розподіл ролей у команді

Учасник	Роль учасника на початку проєкту	Демо 1	Демо 2
Підгорний Артем	голова команди, менеджер, frontend-розробник, scrum-майстер	frontend-розробник, scrum-майстер	frontend-розробник, scrum-майстер
Спіцин Данило	backend-розробник, DevOps	backend-розробник, DevOps	backend-розробник, DevOps
Яковишена Софія	frontend-розробник, дизайнер	frontend-розробник, дизайнер	frontend-розробник, дизайнер
Кардаш Марія	Спеціаліст із документації, тестувальник	аналітик, фахівець із документації	тестувальник, фахівець із документації
Степанюк Олексій	full-stack розробник	backend-розробник	full-stack розробник

Обраною методологією для ведення проєкту стала Scrum [8] через її гнучкість та спрямованість на швидку адаптацію до змінних вимог проєкту. Scrum дозволяє маленьким командам ефективно спілкуватися, швидко реагувати на зміни та доставляти результати через короткі ітерації - спринти. Для розробки ми вибрали Python та FastAPI, оскільки вони забезпечують баланс між швидкістю розробки та продуктивністю, що є важливим для нашої мікросервісної архітектури.

Демонстрація планування спринтів.

Під час планування етапів розробки на початку роботи над проєктом було обрано спринти довжиною в два тижні. Згодом, коли розробка і комунікація в команді стала більш стабільною, було прийнято рішення проводити спринти довжиною в один тиждень.

Кожен спринт починався з планувальної зустрічі для визначення ключових завдань і цілей та розподілу тімлідером завдань для кожного члена команди, який надавав звіт про досягнення та обговорював цілі на наступний спринт.

Коли спринт завершувався, відповідальний за документацію член команди підготовлював презентацію про досягнення доманди за спринт та цілі на наступний. Ці презентації були зібрані та збережені на дошці в Trello (рис. 30) у спеціальній папці, щоб кожен з команди мав до них легкий та швидкий доступ.

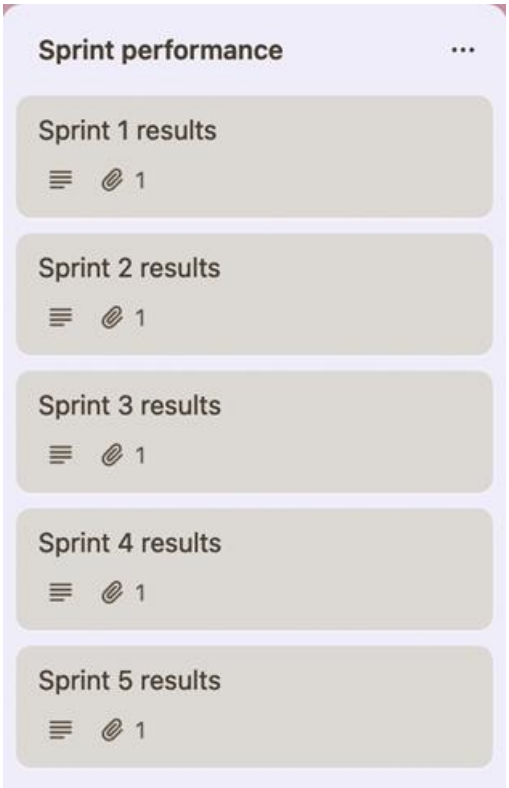


Рисунок 30 – Папка в Trello для зберігання результатів кожного спринту

Рефлексія.

На початку розробки виникла проблема щодо визначення чіткої архітектури проєкту, мови та фреймоворку проєкту, і це виявилась досить складна задача. Спочатку була обрана мова C++, та бібліотека Poco, яка дозволяла досить зручно розробити сервіси. Однак, після довгих обговорень, було прийнято рішення реалізовувати саме мікросервісну архітектуру за допомогою Python Fast API. Цей фреймворк дозволяє реалізувати необхідну для проєкту серверну архітектуру ще простіше, ніж це було б на мові C++.

Після обрання усіх необхідних вимог для розробки, внутрішні процеси в команді було успішно налаштовано.

ВИСНОВКИ

Була розроблена Takeapet.me – онлайн-платформа для взаємодії притулків для тварин та волонтерів. Процес розробки включав декілька етапів: від пошуку та валідації ідеї продукту до імплементації та тестування продукту. Під час пошуку ідеї ми розглянули три різні концепції, зокрема електронне голосування та децентралізований криптовалютний гаманець, але вирішили зосередитися на платформі для прихистку тварин, оскільки це було найбільш відповідно до інтересів команди і мало велике соціальне значення.

У процесі розробки ми вивчили та поглибили наші знання з різних аспектів, включаючи роботу з сучасними вебтехнологіями, такими як Python, FastAPI, PostgreSQL, React, а також принципами UI/UX-дизайну. Ми також успішно впровадили систему управління інформаційною базою та виконали комплексне тестування продукту, включаючи юніт та інтеграційні тести.

Завдання розробки зручного інтерфейсу, забезпечення безпеки даних та інтеграції інструментів для ефективної комунікації між притулками та волонтерами були успішно виконані. Розроблена онлайн-платформа відповідає вимогам до функціональності,

безпеки та інтерактивності, забезпечуючи ефективне та зручне використання для користувачів.

Таким чином, проєкт виявився успішним та актуальним, а команда отримала цінний досвід у розробці вебплатформ та управлінні IT-проєктами.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Look4paws [Електронний ресурс]. Режим доступу до ресурсу: https://look4paws.club4paws.com/.
2. Adopt.ua [Електронний ресурс]. Режим доступу до ресурсу: https://www.adopt.ua/
3. Happy Paw [Електронний ресурс]. Режим доступу до ресурсу: https://happypaw.ua/
4. Python – Offical website [Електронний ресурс]. Режим доступу до ресурсу: https://www.python.org/about/
5. FastAPI – Offical website [Електронний ресурс]. Режим доступу до ресурсу: https://fastapi.tiangolo.com/advanced/generate-clients/
6. PostgreSQL – Offical webste [Електронний ресурс]:[Вебсайт] – Режим доступу до ресурсу: https://www.postgresql.org/
7. React – Offical webste [Електронний ресурс].Режим доступу до ресурсу: https://uk.legacy.reactjs.org/
8. Scrum – Навчальна стаття [Електронний ресурс]. Режим доступу до ресурсу: https://www.scrum.org/resources/what-scrum-module
9. Закон України \"Про захист тварин від жорстокого поводження\" [Електронний ресурс]. Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/3447-15#Text.
10. Закон України \"Про волонтерську діяльність\" [Електронний ресурс]. Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/3236-17#Text
11. Стаття 299 Кримінального кодексу України (Жорстоке поводження з тваринами) [Електронний ресурс]. Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2120-19#Text
12. Стаття 154 Кодексу про адміністративні порушення [Електронний ресурс]. Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/80731-10#Text

МОБІЛЬНИЙ ЗАСТОСУНОК "CASUAL TRAVEL"

Олександр Ємець, Богдан Катрич, Владислав Нискогуз, Юлія Рак, Олена Рак, Інна Фисюк

Дана робота описує розробку мобільного застосунку "CasUAlTravel", який допомагає користувачам планувати індивідуалізовані подорожі з урахуванням особистих інтересів і часу. В роботі детально представлено процес розробки як фронтенду, так і бекенду застосунку, а також обговорюються основні технології та інструменти, що були використані. Застосунок пропонує ряд ключових функцій, включаючи персоналізоване планування маршрутів, автоматичну оптимізацію маршрутів з урахуванням актуальних умов, інтерактивні карти та систему гейміфікації. Звіт також розглядає аспекти користувацького інтерфейсу і досвіду, а також підходи до тестування та оптимізації застосунку. Робота висвітлює потенціал застосунку для забезпечення зручного та ефективного планування подорожей, роблячи процес більш насиченим та захоплюючим для користувачів.

This work details the development of the mobile application "CasUAlTravel", designed to assist users in planning personalized travel experiences based on their individual interests and available time. The document extensively discusses the development processes of both the frontend and backend components of the application, highlighting the key technologies and tools employed. "CasUAlTravel" offers several essential features, including personalized route planning, automatic route optimization considering real-time conditions, interactive maps, a gamification system. The report also covers user interface and experience aspects, as well as approaches to testing and optimizing the application. This work underscores the application's potential to facilitate convenient and efficient travel planning, enriching the user experience and making the process more engaging and enjoyable.

МОБІЛЬНИЙ ЗАСТОСУНОК, ПЕРСОНАЛІЗОВАНЕ ПЛАНУВАННЯ ПОДОРОЖЕЙ, ІНТЕРАКТИВНІ КАРТИ, ГЕЙМІФІКАЦІЯ, ОПТИМІЗАЦІЯ МАРШРУТІВ, РОЗРОБКА ПЗ, ІНТЕРФЕЙС.

MOBILE APPLICATION, PERSONALIZED TRIP PLANNING, INTERACTIVE MAPS, GAMIFICATION, ROUTE OPTIMIZATION, SOFTWARE DEVELOPMENT, INTERFACE.

ВСТУП

Сучасний світ туризму невпинно розвивається, відкриваючи нові можливості для пізнання та відкриття нових місць. У цьому контексті актуальним стає розробка додатків, які допомагають мандрівникам ефективно планувати свої подорожі. Один із таких проєктів – "CasUAlTravel", український застосунок, який забезпечує персоналізовані путівники відповідно до індивідуальних інтересів користувачів.

У процесі дослідження та розробки цього проєкту було застосовано ряд методів: аналіз сучасних трендів у сфері туризму, опитування потенційних користувачів, та вивчення існуючих аналогів. Це дослідження мало на меті не тільки розробку функціонального та зручного продукту, але й закріплення знань та навичок у галузі програмування, дизайну інтерфейсів, аналізу даних та командної комунікації.

ПОШУК ТА ВАЛІДАЦІЯ ІДЕЇ

Опис пошуку ідеї продукту. На етапі пошуку ідеї продукту, команда проєкту розглянула декілька можливих напрямків для розробки. Перед тим як обрати найбільш цікавий учасники провели аналіз та обговорення наступних ідей:

● **Сучасний будівельний сайт:** команда розглядала можливість створення сучасного вебсайту для будівельних компаній. Основною ідеєю було надати клієнтам можливість знайти та вибрати підходящого підрядника для будівництва. Основною перевагою цієї ідеї була наявність зацікавленого замовника і можливість розробляти даний проєкт як комерційний продукт. Проте після аналізу конкурентного ринку та розгляду ресурсів, цей напрямок було відкинуто через високу конкуренцію та складність масштабування.

● **Платформа для збору благодійних коштів:** Іншою ідеєю була розробка платформи для збору благодійних коштів. Метою було забезпечити можливість організаціям та особам-волонтерам організовувати збори коштів для благодійних потреб України. Проте після дослідження цього сектору ринку ми визнали, що такий проєкт вимагав б великих зусиль для організації додаткових бізнес процесів таких як: перевірка зборів на шахрайство, верифікація осіб волонтерів/військових тощо, тому ця ідею була відкинута.

● **Застосунок для відслідковування IT подій в Україні:** Третьою ідеєю було створення застосунку, який би дозволяв користувачам відслідковувати та отримувати оновлення щодо IT подій в Україні. Ми розглядали цей напрямок через актуальність та популярність IT галузі в Україні. Проте після аналізу конкуренції та реалізації, що важливо забезпечити інноваційну та унікальну пропозицію, ми вирішили не продовжувати розвивати цю ідею через обмежену можливість привернення аудиторії.

● **Персоналізований планувальник подорожей:** Після уважного аналізу різних ідей та їх потенціалу, ми обрали дану сферу як наш напрямок. Спираючись на аналіз загального інтересу до туризму та подорожей в Україні, ми вирішили реалізувати застосунок "CasUAlTravel", який допомагає користувачам ефективно планувати свої подорожі відповідно до їхніх індивідуальних інтересів.

Отже, після ретельного аналізу і розгляду різних ідей, ми прийняли рішення розробляти "CasUAlTravel", що виявився успішним та актуальним проєктом для нашої команди.

Огляд наявних на ринку застосунків.

На етапі роздумів над ідеєї застосунку наша команда активно вивчила сучасний ринок IT продуктів і проаналізувала деяких потенційних конкурентів, які пропонують унікальні функції та сервіси. Це дозволило нам зрозуміти, які переваги та недоліки мають ці продукти у порівнянні один з одним. Огляд конкурентів CasUAlTravel включає такі IT-продукти (рис. 1):

VisitUkraine [10]: Ця платформа надає різноманітні функції для туристів в Україні та українців за кордоном. Вона включає онлайн-базу турів по Україні. Недоліки: відсутність можливості створення кастомного туру та перегляду маршруту на карті.

WalQlike [11]: Мобільний додаток для прогулянок у місті у формі квесту. Він пропонує унікальні маршрути з історіями та секретними локаціями. Недоліки: відсутність можливості створення кастомної прогулянки.

Tripio Travel App [12]: Додаток для планування подорожей, який пропонує персоналізовані маршрути. Недоліки: обмежена інформація про визначні місця в Україні та відсутність гейміфікації.

Рисунок 1 – Порівняльна таблиця конкурентів застосунку CasUAlTravel

Мапа продукту

Проблема, що вирішує продукт. Основна проблема, яку вирішує "CasUAlTravel", полягає у недостатності персоналізації у плануванні подорожей. Більшість існуючих додатків пропонують стандартні путівники, які не враховують індивідуальні інтереси та переваги користувача.

Актуальність. Додаток є актуальним, оскільки він забезпечує демократичний доступ до культурних, рекреаційних та спортивних локацій, підтримує туристичні та культурні установи, відповідає тенденціям сталого розвитку, пропонує персоналізовані рекомендації, забезпечує цифрову зручність.

Місія. Покращити культурний та рекреаційний досвід, надаючи доступну та зручну платформу для користувачів, щоб досліджувати арт-центри, природні локації столиці, створювати нові асоціації зі столицею та підтримувати сталі.

Сегменти клієнтів:

Молоді професіонали

Вік: 25-35 років. Молоді професіонали віддають перевагу ефективності та зручності при плануванні своїх подорожей, цінуючи свій час і шукаючи оптимальне поєднання відпочинку та витраченого на нього часу.

- Студенти.

Вік: 18-24 роки. Студенти, як активні користувачі мобільних додатків, часто обмежені у бюджеті, але водночас прагнуть відкрити для себе нові місця та культурні пам'ятки.

- Молоді сім'ї.

Вік: 30-45 років. Молоді сім’ї шукають безпечні та комфортні умови для сімейного відпочинку, де можна б заздалегідь спланувати все необхідне.

- Іноземні туристи.

Вік: 18-70 років. Для іноземних туристів важливе глибоке занурення в місцеву культуру та історію.

Унікальна ціннісна пропозиція.

"CasUAlTravel" відрізняється від конкурентів тим, що пропонує повністю персоналізовані маршрути, засновані на унікальних інтересах користувачів. Це не просто додаток для планування подорожей, а персональний помічник, який допомагає відкрити місто з нового боку.

Рішення для проблем:

- Для молодих подорожуючих: пропонуються маршрути з акцентом на сучасне мистецтво, модні місця та незвичайні пам'ятки.
- Для досвідчених туристів: індивідуальні маршрути, що включають історичні та культурні пам'ятки.
- Для сімей: пропозиції, які враховують інтереси як дорослих, так і дітей.

Завдання: Персоналізація подорожей. Застосунок дозволяє користувачам пройти опитування, щоб визначити їхні унікальні інтереси, що допомагає у підборі місць для відвідування. Повна автоматизація планування пропонує опцію "автоматично згенерувати маршрут", де алгоритм враховує особисті інтереси та пропонує оптимальний маршрут. Візуалізація та інтерактивність інформації візуально підсилює досвід користувача, виділяючи важливі місця та пам'ятки (не скучний застосунок з текстом і картинкою). Гнучкість вибору дозволяє користувачам мати свободу вибору та змінювати маршрути за бажанням, забезпечуючи гнучке та зручне планування подорожі.

Канали(потенційні):

- Цифрові медіа: соціальні мережі, блоги, співпраця з популярними туристичними блогерами.
- Традиційні медіа: реклама у туристичних журналах, участь у туристичних виставках.
- Партнерства з туристичними агенціями та готелями.

Джерела доходів(потенційні):

Freemium-модель для мобільного застосунку. Безкоштовний доступ до основних функцій приваблює користувачів, заохочуючи їх до активного використання застосунку та в подальшому до оновлення до преміум-версії для розширеного функціоналу.

Інтеграційна модель для вебсайту. Монетизація вебсайту буде здійснюватися шляхом співпраці з постачальниками туристичних послуг, отримуючи комісію за бронювання та інші послуги через нашу платформу.

Економічна оцінка(потенційна):

Монетизація за моделлю Freemium: користувачі = 6000 × 0.10 (10% потенційних користувачів, що перейшли на підписку) = 600. Дохід (Freem реклама) = 600 * 75 грн (за підписку) = 45 000 грн Дохід (in-app реклама) = 6000 * 1 * $0.05 = $300 (10 971 грн).

Витрати: витрати на маркетинг та рекламу до 900 грн/міс. (10 800/рік) Чистий грошовий потік = (Дохід від Freemium+Дохід від реклами)−(Загальні витрати на маркетинг) 45 000 + 10 971 - 10 800 = 45 171 NPV= 45,171 / (1 рік + 0,10 диск ставка) - 10 800 (поч. інвест) = 30,264. Виконавши розрахунок, отримаємо NPV для цього проєкту.

Результат позитивний, отже можна вважати, що проєкт прибутковий при заданих умовах.

Сильні і слабкі сторони продукту подані в рамках SWOT-аналізу (рис. 2)

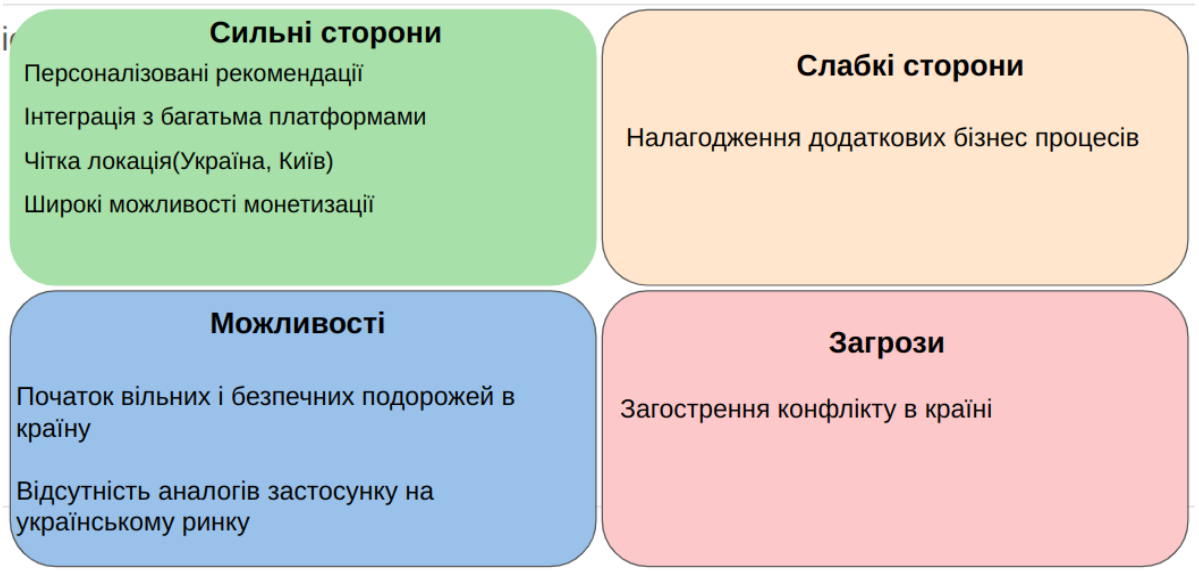


Рисунок 2 – SWOT-аналіз застосунку

Огляд використаних технологій

Java Spring для Backend:

Огляд: Java Spring є потужним та гнучким фреймворком для створення вебзастосунків. Він підтримує розширену конфігурацію та різноманітність модулів, що полегшує розробку складних бекенд-систем.

Чому обрано: Вибір Java Spring обумовлений його надійністю, високою продуктивністю, та великою спільнотою розробників. Цей фреймворк дозволяє ефективно управляти безпекою, транзакціями, та зв'язками з базами даних.

React для Frontend:

Огляд: React є одним з найпопулярніших JavaScript фреймворків для розробки інтерактивних інтерфейсів. Він забезпечує високу продуктивність завдяки використанню віртуального DOM.

Чому обрано: React був обраний через його гнучкість, компонентний підхід, та широку підтримку спільноти. Це дозволяє швидко розробляти візуально привабливі та відгукові інтерфейси, що є критично важливим для користувацького досвіду у додатках для подорожей.

PostgreSQL як база даних:

Огляд: PostgreSQL є потужною, відкритою системою управління базами даних. Вона підтримує складні запити, великі обсяги даних та має хорошу продуктивність.

Чому обрано: Вибір PostgreSQL зумовлений його високою надійністю, підтримкою розширених функцій SQL та вбудованої підтримки JSON, що є важливим для зберігання та обробки даних користувачів та маршрутів.

Git та GitHub:

Огляд: Git є відомою системою контролю версій, а GitHub надає хмарну платформу для зберігання репозиторіїв та співпраці.

Чому обрано: Використання Git та GitHub забезпечує ефективне управління кодом, спрощує співпрацю в команді та дозволяє зберігати історію змін. Це є ключовим для координації роботи розробників та забезпечення стабільності проєкту.

Комбінація цих технологій надає "CasUAlTravel" міцний фундамент для створення стабільного, продуктивного та масштабованого додатку, який відповідає сучасним вимогам ринку та користувачів.

ОПИС ПРОГРАМНОГО ПРОДУКТУ

Призначення і цілі створення застосунку

Призначення системи. Застосунок "CasUAlTravel" був створений з метою надання мандрівникам зручного та інтуїтивно зрозумілого інструменту для планування персоналізованих подорожей. Ця система дозволяє користувачам виходити за межі звичайних туристичних маршрутів, пропонуючи унікальні варіанти відпочинку, що відповідають їхнім індивідуальним інтересам та перевагам.

Цілі застосунку. Персоналізація досвіду подорожей: Основна мета "CasUAlTravel" – забезпечити користувачам можливість відкривати місця та активності, які відповідають їхньому стилю життя та інтересам. Через унікальні опитування, система збирає інформацію про переваги користувачів та використовує ці дані для створення індивідуальних пропозицій.

Спрощення процесу планування подорожей: Застосунок призначений для зниження зусиль, пов'язаних з плануванням подорожей, шляхом надання користувачам докладної інформації про пам'ятки, місця відпочинку, ресторани та інші активності, які відповідають їхнім запитам.

Підтримка та розвиток місцевого туризму: "CasUAlTravel" ставить за мету не лише допомогу мандрівникам, але й підтримку місцевих громад та бізнесів. Це досягається за допомогою включення в застосунок інформації про локальні атракції та невеликі підприємства, які можуть бути цікавими для туристів.

Забезпечення інтерактивного досвіду: З використанням інтерактивної карти, користувачі мають можливість візуально вибирати та планувати свої маршрути, відчуваючи більшу залученість у процес планування подорожей.

Стимулювання культурного обміну: "CasUAlTravel" спрямований на стимулювання культурного обміну між мандрівниками та місцевими громадами, надаючи інформацію про культурні заходи, традиції та особливості різних регіонів.

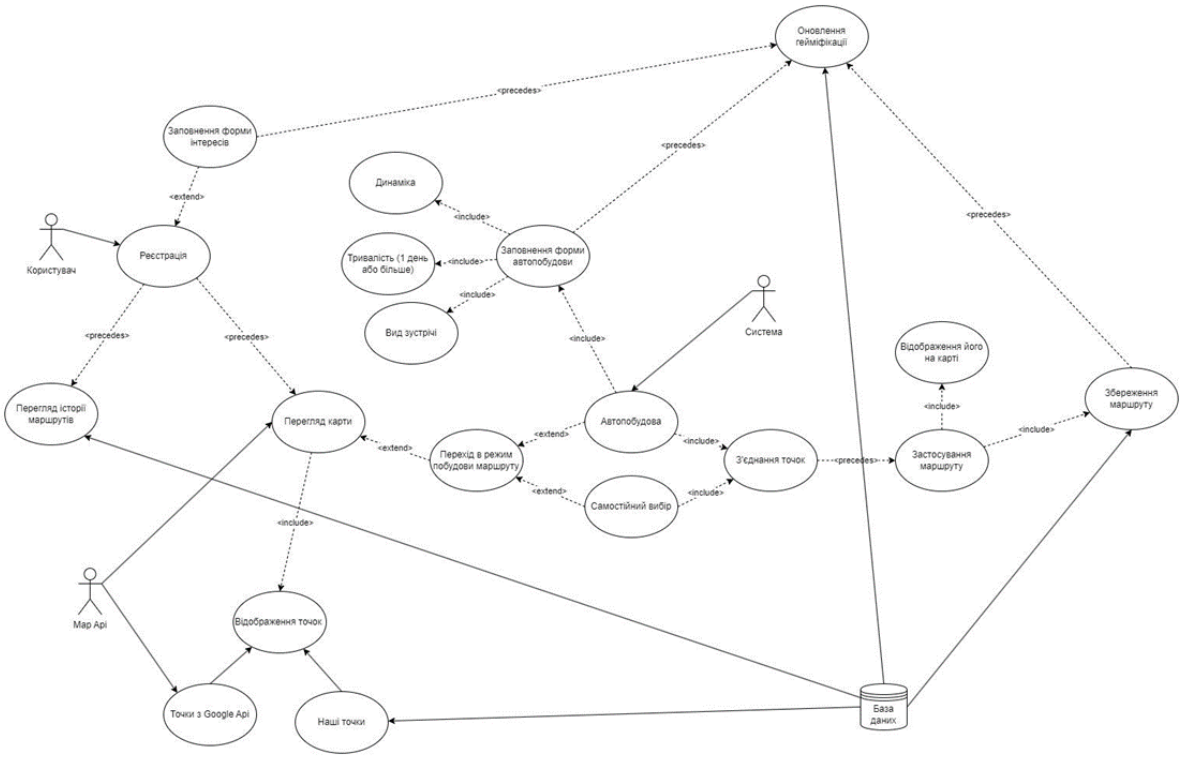


Рисунок 3 – Use Case діаграма застосунку CasUAlTravel

Вимоги до застосунку

Зрозумілий користувацький інтерфейс: Застосунок повинен бути інтуїтивно зрозумілим та зручним для користувачів різного рівня технічної підготовки.

Персоналізація: "CasUAlTravel" має надавати можливість персоналізації досвіду користувача, враховуючи їхні індивідуальні інтереси та вподобання.

Інформативність: Застосунок повинен надавати користувачам докладну інформацію про місця, пам'ятки, ресторани та інші об'єкти, що можуть бути цікавими для них.

Мобільність: "CasUAlTravel" повинен бути доступний на мобільних платформах (iOS та Android), щоб користувачі могли використовувати його під час подорожей.

Вимоги до структури та функціонування застосунку

Авторизація та реєстрація: **Застосунок** повинен мати функціонал авторизації та реєстрації користувачів з можливістю збереження персональних даних.

Опитування користувачів: "CasUAlTravel" має надавати можливість користувачам заповнити опитування щодо їхніх інтересів та вподобань.

Персоналізовані рекомендації: На основі заповнених опитувань, система повинна надавати персоналізовані рекомендації щодо маршрутів та активностей.

Інтерактивна карта: Застосунок повинен включати інтерактивну карту, на якій користувачі можуть вибирати та планувати свої маршрути.

Інформація про місця: "CasUAlTravel" має надавати детальну інформацію про місця відпочинку, пам'ятки, ресторани та інші активності, включаючи години роботи, вартість, контакти та відгуки користувачів.

Можливість збереження відвіданих місць: Користувачі повинні мати можливість зберігати та планувати свої маршрути для подальшого використання.

Мовна локалізація: Застосунок повинен підтримувати різні мови для комфортного використання користувачами з різних країн.

Безпека даних: Застосунок повинен забезпечувати безпеку та конфіденційність персональних даних користувачів.

Системні вимоги:

- Операційна система: Android 6.0 або новіше**.**
- Процесор: Мінімум чотирьохядерний процесор.
- Оперативна пам'ять: Мінімум 2 ГБ оперативної пам'яті.
- Внутрішня пам'ять: Доступне місце для інсталяції додатку та збереження даних.
- Доступ до Інтернету: Для завантаження мап, інформації про місця та оновлення даних.
- GPS-модуль: Для визначення місцезнаходження користувача та побудови маршрутів.
- Дозволи: Додаток може вимагати дозволів на доступ до місцезнаходження, камери, мікрофону та збереження даних на пристрої.
- Сумісність з екранами: Пристрій з екраном роздільної здатності, зручним для користування.
- Інтернет-з'єднання: Для завантаження карт, оновлення даних та обміну інформацією з сервером.
- Актуальна версія додатку: Користувач повинен завжди оновлювати додаток до останньої версії для отримання нових функцій та покращень.

Опис логічної структури бази даних

Застосунок "CasUAlTravel" використовує реляційну базу даних для збереження та організації інформації. Для цього використовується система управління базами даних (СУБД) PostgreSQL.

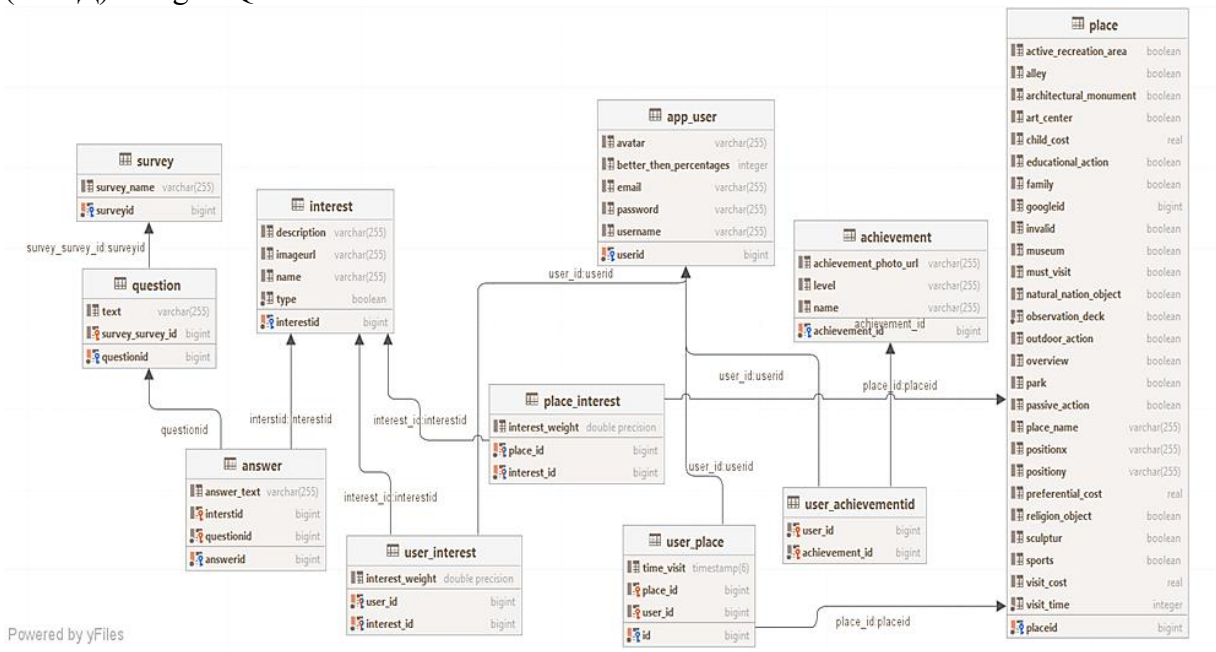


Рисунок 3 – Схема бази даних

Опис і організація таблиць в застосунку

Таблиця 1. Повний перелік таблиць БД

№	Таблиця	Опис
1	survey	Таблиця для збереження інформації про опитування.
2	question	Таблиця, що містить питання, які включені до опитувань.
3	answer	Таблиця для збереження відповідей на питання з опитувань.
4	interest	Таблиця з інформацією про інтереси, які можуть бути пов'язані з користувачами або місцями.
5	app_user	Таблиця з даними користувачів застосунку, включаючи інформацію для входу та профілю.
6	achievement	Таблиця для відстеження досягнень користувачів.
7	user_achievement	Таблиця для зв'язування користувачів з їх досягненнями.
8	place_interest	Таблиця, яка пов'язує місця з інтересами та їх значимістю.
9	user_interest	Таблиця для зв'язування користувачів з їх інтересами.
10	user_place	Таблиця для запису відвідувань користувачами різних місць.
11	place	Таблиця для збереження інформації про місця, включаючи коефіцієнти за якими автоматично будується маршрут для користувача.

Інформаційний модуль "Опитування"

Суть даного інформаційного модулю полягає в об’єднанні таких сутностей як опитування користувача(Survey), питання цього опитування (Question) та відповіді на конкретне питання(Answer).

Таблиця 2. Інформаційний модуль "Опитування"

Таблиця	Атрибут	Тип Даних	Опис
survey	survey_name	varchar(255)	Назва опитування.
survey	surveyid	bigint	Унікальний ідентифікатор опитування.
question	text	varchar(255)	Текст питання.
question	survey_survey_id	bigint	Ідентифікатор опитування, до якого належить питання.
question	questionid	bigint	Унікальний ідентифікатор питання.
answer	answer_text	varchar(255)	Текст відповіді.
answer	interestid	bigint	Ідентифікатор інтересу, пов'язаного з відповіддю.
answer	questionid	bigint	Ідентифікатор питання, до якого належить відповідь.
answer	answerid	bigint	Унікальний ідентифікатор відповіді.

Таблиця 3. Інформаційний модуль "Користувач і інтереси"

Таблиця	Атрибут	Тип Даних	Опис
interest	description	varchar(255)	Опис інтересу.
interest	imageurl	varchar(255)	URL зображення, асоційованого з інтересом.
interest	name	varchar(255)	Назва інтересу.
interest	type	boolean	Тип або категорія інтересу.
interest	interestid	bigint	Унікальний ідентифікатор інтересу.
app_user	avatar	varchar(255)	URL аватара користувача.
app_user	better_then_percentages	integer	Статистика або рейтинг користувача, порівняно з іншими.
app_user	email	varchar(255)	Електронна пошта користувача.
app_user	password	varchar(255)	Пароль для входу користувача.
app_user	username	varchar(255)	Ім'я користувача для входу в систему.
app_user	userid	bigint	Унікальний ідентифікатор користувача.
achievement	achievement_photo_url	varchar(255)	URL фотографії, що асоційована з досягненням.
achievement	level	varchar(255)	Рівень досягнення.
achievement	name	varchar(255)	Назва досягнення.
achievement	achievement_id	bigint	Унікальний ідентифікатор досягнення.
user_achievement	user_id	bigint	Ідентифікатор користувача, що пов'язаний з досягненням.

Таблиця	Атрибут	Тип Даних	Опис
user_achievement	achievement_id	bigint	Ідентифікатор досягнення, що пов'язане з користувачем.
user_interest	interest_weight	double precision	Вага інтересу, що відображає зацікавленість користувача.
user_interest	user_id	bigint	Ідентифікатор користувача, що має цей інтерес.
user_interest	interest_id	bigint	Ідентифікатор інтересу, що пов'язаний з користувачем.
user_place	time_visit	timestamp(6)	Час відвідування місця користувачем.
user_place	place_id	bigint	Ідентифікатор місця, яке відвідував

Інформаційний модуль "Локації". Даний модуль має єдину основну таблицю Place. ЇЇ опис наведено в таблиці нижче.

Таблиця 4. Таблиця Place

Атрибут	Тип Даних	Опис
placeID	Long	Унікальний ідентифікатор місця.
googleID	Long	Ідентифікатор місця в Google Services.
placeName	String	Назва місця.
positionX	String	Географічна широта місця.
positionY	String	Географічна довгота місця.
visitTime	int	Час, рекомендований для відвідування місця.
visitCost	Float	Вартість відвідування для дорослих.
childCost	Float	Вартість відвідування для дітей.
preferentialCost	Float	Пільгова вартість відвідування.
sports	Boolean	Позначка спортивного характеру місця.
overview	Boolean	Чи є місце оглядовим пунктом.
interests	Map<Interest, Double>	Асоціації з інтересами та їх вагою.
userPlaces	List<UserPlace>	Список відвідувань користувачів цього місця.

UI/UX-дизайн продукту

Закони та принципи UX-дизайну для продукту "CasUAlTravel":

- **Закон Простоти (Law of Simplicity):** Принцип стверджує, що дизайн має бути мінімалістичним та простим для користувача. Використання простих інтерфейсів та мінімальної кількості кроків для досягнення мети полегшує використання додатку.
- **Закон Розташування (Law of Proximity):** Елементи, які знаходяться близько один до одного, сприймаються як пов'язані. У дизайні це може бути використано для групування схожих елементів, таких як місця або активності, для зручності користувача.
- **Закон Гестальта (Gestalt Law):** Цей закон вказує на те, що елементи дизайну мають бути організовані так, щоб користувач міг сприймати їх як цілісну сутність. Наприклад, використання групування місць за категоріями на карті допомагає користувачеві знайти інформацію швидше.

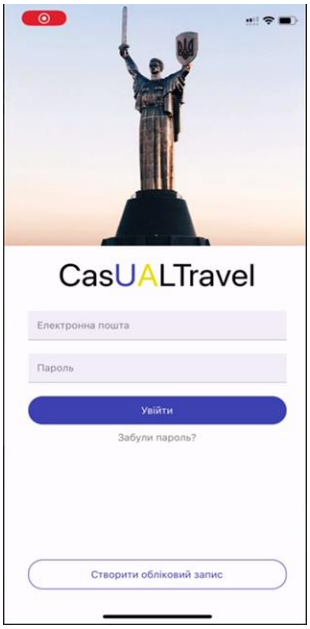


Рисунок 4 – Сторінка реєстрації користувача

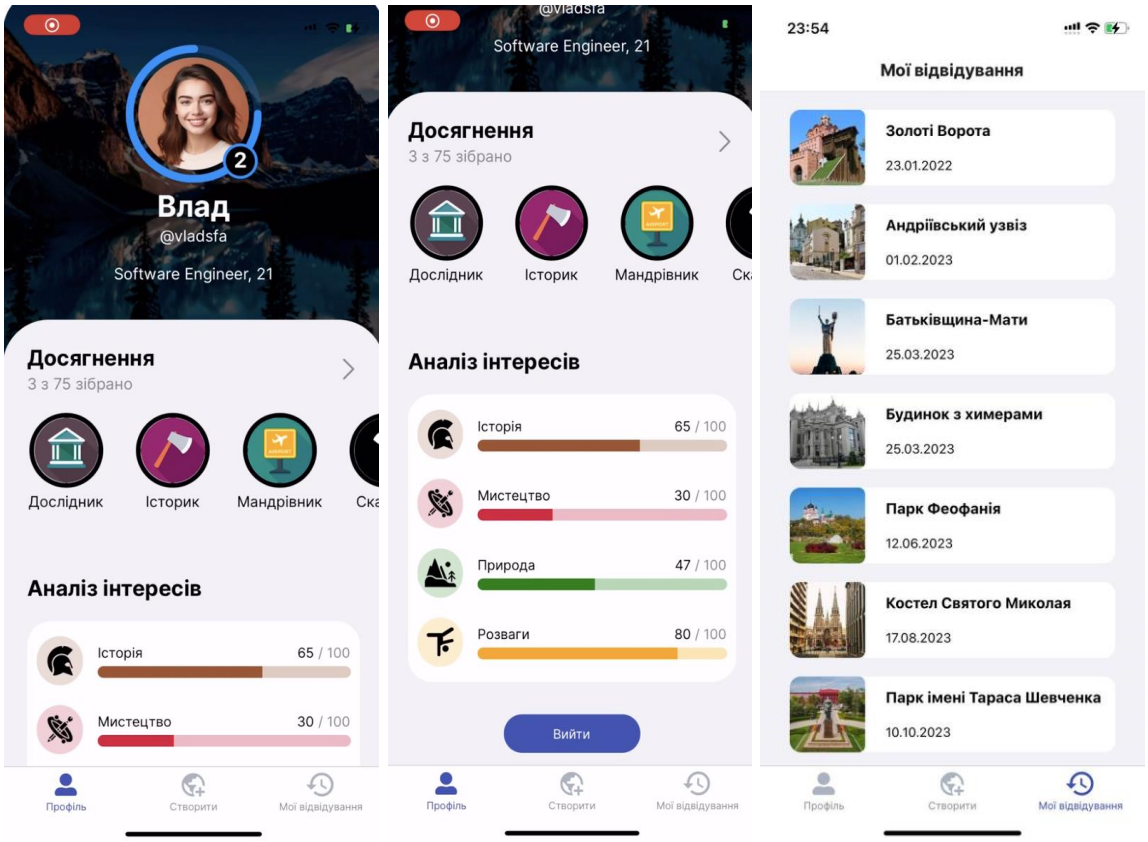


Рисунок 5 – Вигляд сторінки профіля користувача

ІМПЛЕМЕНТАЦІЯ ПРОДУКТУ

Загальна структура.

Процес реалізації продукту CasUAlTravel базувався на використанні сучасних технологій та методологій розробки. Основна інфраструктура системи була побудована на Java Spring для розробки бекенду та React для фронтенду. Такий вибір технологій дозволив створити продукт, який ефективно взаємодіє з користувачами та забезпечує зручний інтерфейс для планування подорожей.

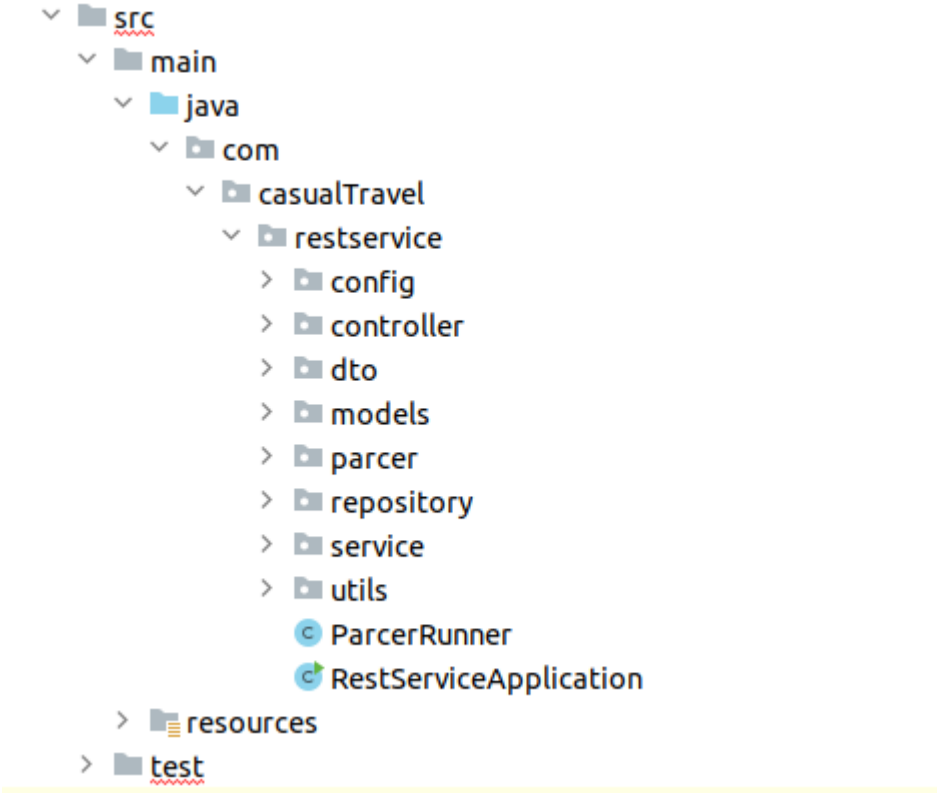


Рисунок 6 – Структура Java Spring проєкту

Концепція Code First.

Під час розробки CasualTravel використовувалася концепція "Code First". Ця підхід передбачає створення моделей даних та логіки додатку безпосередньо в коді, після чого на їх основі автоматично генерується схема бази даних. Це дозволило нам більш гнучко працювати з даними та вносити зміни без необхідності вручну оновлювати базу даних, так як наші моделі і структура час від часу зазнавали певних змін протягом реалізації застосунку.

Автоматична побудова маршруту. Основна Концепція:

Автоматична побудова маршруту в CasualTravel - це інноваційна функція, яка дозволяє користувачам генерувати персоналізовані маршрути подорожей автоматично, враховуючи їхні інтереси, переваги та часові обмеження. Ця функція використовує алгоритм Дейкстри та дані про пам'ятки для створення оптимального та ефективного маршруту подорожі.

Функціональність:

- Персоналізація: Після проходження опитування або введення індивідуальних уподобань, система використовує ці дані для побудови маршруту, який відповідає унікальним інтересам користувача.

● Оптимізація Маршруту: Алгоритм враховує географічне розташування пам'яток, час відвідування та пересування, щоб мінімізувати час переїздів та максимізувати час, проведений у місцях відвідування.

● Гнучкість: Користувачі можуть налаштувати довжину маршруту, вибрати конкретні дні відвідування, та додавати або видаляти пункти з маршруту, надаючи додаткову гнучкість.

Деталі реалізації.

Як було описано вище, застосунок CasualTravel використовує алгоритм Дейкстри для знаходження найбільш оптимального шляху для подорожі користувача. Даний алгоритм знаходить найкоротший шлях від однієї вершини графу до всіх інших вершин.

Етап 1. Попередній аналіз і розрахунок характеристик розташування.

Розрахунок "Цікавості" кожної точки для користувача в певному радіусі(залежить від обраного користувачем часу подорожі). Розрахунок відбувається на основі аналізу інтересів користувача:

```
Цікавість(A) =
(Зацікавленість в історії користувача * Коефіцієнт історичності точки A) +
(Зацікавленість в природі користувача * Коефіцієнт природи точки A) +
(Зацікавленість в культурі користувача * Коефіцієнт культури точки A) + ...
```

Етап 2. Генерація масиву котрий описує ребра графу.

За допомогою описаного раніше методу ми порахували умовну "Цікавість" кожної точки на карті. Але в теорії графів немає такого поняття як вага вершини. Необхідно знайти підхід як вирахувати вагу всіх ребер між існуючими вершинами. Загалом, ми робимо це по наступній формулі(спрощена для демонстрації): W=I/t. Тут W - вага ребра, I - обрахована раніше цікавість точки куди це ребро вказує, t - час, котрий необхідний, щоб туди добратись

Етап 3. Дизайн і генерація графу локацій.

Проєктування роботи алгоритму здійснено таким чином що на вхід алгоритму подається всього один набір параметрів, масив наступних структур:

```
Struct Edge
{
    uint8 vertex_to;
    uint8 vertex_from;
    uint8 weight;
}
```

Масив описаних структур задає зважений граф, на котрому ми і будемо далі шукати найбільш оптимальний(найкоротший) шлях

Етап 4. Побудова оптимального маршруту і відображення результату.

Алгоритм зберігає найоптимальніший маршрут і повертає на вихід масив вершин з котрих складається даний шлях. Масив обробляється і надсилається на Фронт-енд частину застосунку для відображення.

Заповнення БД та аналіз характеристик локацій

Заповнення бази даних та аналіз характеристик локацій є ключовими компонентами для забезпечення функціональності застосунку CasualTravel. Метою створення бази даних є побудова демоверсії маршрутів за результатами опитувань щодо інтересів та вподобань споживачів. Ця робота вимагає детального збору, організації та аналізу інформації про різні туристичні місця.

Критерії у виборі об'єктів: цікавість обраній цільовій аудиторії розташування відсутніть необхідності в прямії комунікації з представниками локації (актуальність та доступність даних).

Типи об'єктів: Музей, Арт-центр, Релігійний об‘єкт, Пам'ятка архітектури, Скульптура, Парк, Природні об‘єкти національного значення, Алея, Оглядовий майданчик.

Критерії характеристики типів об'єктів: Спортивний, Оглядовий, Пасивний відпочинок, Просвітницький, Розташування, Доступний для людей з інвалідністю, Сімейний, Must visit, Зона активного відпочинку.

У застосунку CasualTravel ми використовуємо чотири основні коефіцієнти для характеристики міста: Історія, Природа, Культура, та Розваги. Кожен коефіцієнт відображає важливі аспекти міста, що допомагає користувачам вибрати місця для відвідування на основі їхніх персональних інтересів. Наприклад, для певного міста коефіцієнти можуть бути розподілені таким чином: Історія (0.37), Природа (0.18), Культура (0.19), Розваги (0.47).

Внесок колег з географічного факультету. Цю важливу частину роботи виконали наші колеги з географічного факультету, які застосували свої спеціалізовані знання для забезпечення точності та якості даних у БД. Основні аспекти роботи:

- **Збір Даних:** Колеги провели збір даних про місцевості, включаючи історичні пам'ятки, природні об'єкти, культурні та розважальні заклади. Зосередили увагу на детальному описі кожної локації, включаючи її історію, значимість, доступність та інші важливі атрибути.
- **Класифікація та Аналіз:** Проведено аналіз та класифікацію локацій за різними категоріями, такими як культура, історія, природа, розваги. Оцінено кожну локацію за певними параметрами, щоб визначити її внесок у загальний туристичний потенціал міста.

Результатом цієї спільної роботи є добре структурована та інформативна база даних, яка дозволяє CasualTravel точно та ефективно відповідати на запити користувачів, створюючи персоналізовані подорожі на основі комплексного аналізу локацій.

Опитування інтересів і вподобань користувача.

Опитування в нашому застосунку виконують важливу роль формування повної картини вподобань користувача, що має допомогти формувати персоналізовані маршрути. Гейміфікація в опитуванні активніше залучає користувача у процес проходження тесту на інтереси, дозволяє самостійно керувати етапами проходження опитування. В результаті, опитування дозволяє визначити який з 4 інтересів домінує в користувача: історія мистецтво природа розваги: історія, мистецтво, природа, розваги.

Опитування №1 (для розширення інформації про користувача в особистому кабінеті).

Мета опитування: формування повної картини вподобань користувача, що має допомогти формувати персоналізовані маршрути. "Легендарний Світ Відкриттів". Кожен вибір веде до нових відкриттів та захопливих пригод.

Опитування №2 (при формуванні конкретного маршруту).

Мета опитування: збір поточної інформації про користувача для створення індивідуального маршруту.

ТЕСТУВАННЯ

Тестування є невід'ємною частиною процесу розробки додатку CasualTravel. Воно включає ретельну перевірку всіх аспектів програми, від функціональності до продуктивності. Наша мета тестування полягає в забезпеченні того, що кінцевий продукт відповідає всім вимогам та очікуванням користувачів, а також є стабільним та надійним.

Ми впровадили юніт тестування на ранніх етапах розробки для перевірки індивідуальних компонентів нашого додатку. Цей підхід дозволив нам швидко виявляти та виправляти помилки, а також сприяв постійному покращенню якості коду. Ключові модулі застосунку були ретельно протестовані для переконання в тому, що вони працюють як очікувалося ізольовано від інших частин системи.

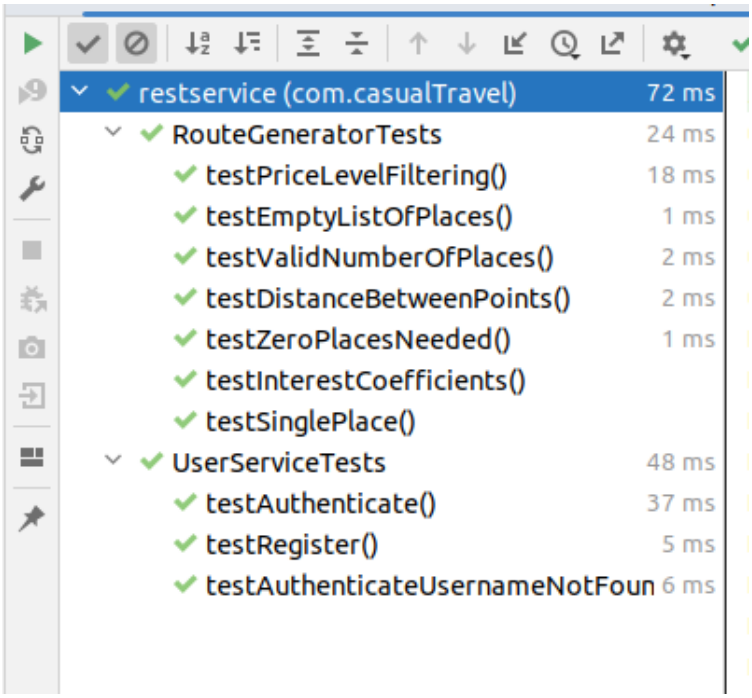


Рисунок 7 – Структура юніт-тестів

ДОКУМЕНТАЦІЯ ПРОЄКТУ

Загальний огляд документації. У проєкті CasUAlTravel ми приділяємо особливу увагу підготовці документації, яка включає в себе як технічні аспекти, так і інструкції для кінцевих користувачів. Це забезпечує глибоке розуміння продукту нашою командою та допомагає забезпечити безперебійне впровадження та експлуатацію додатку.

Технічна документація. Технічна документація включає в себе всі специфікації, архітектурні описи, алгоритми, кодові бази та інструкції з розгортання. Вона є фундаментом для розробників та інженерів, які працюють над проєктом, і служить довідником для нових членів команди або зовнішніх розробників.

Використання Trello для документації. Для організації нашої документації ми використовуємо популярний інструмент управління проєктами Trello. В рамках нашої дошки Trello, ми створили окремі колонки для технічної та користувацької документації. Це дозволяє нам підтримувати чистоту та організацію, забезпечуючи швидкий доступ до необхідних документів для всіх членів команди. Кожна карточка в колонці містить посилання на документи, версії документації, а також статус оновлення. Це забезпечує прозорість та ефективність робочого процесу, а також дозволяє з легкістю відстежувати зміни та оновлення документації.

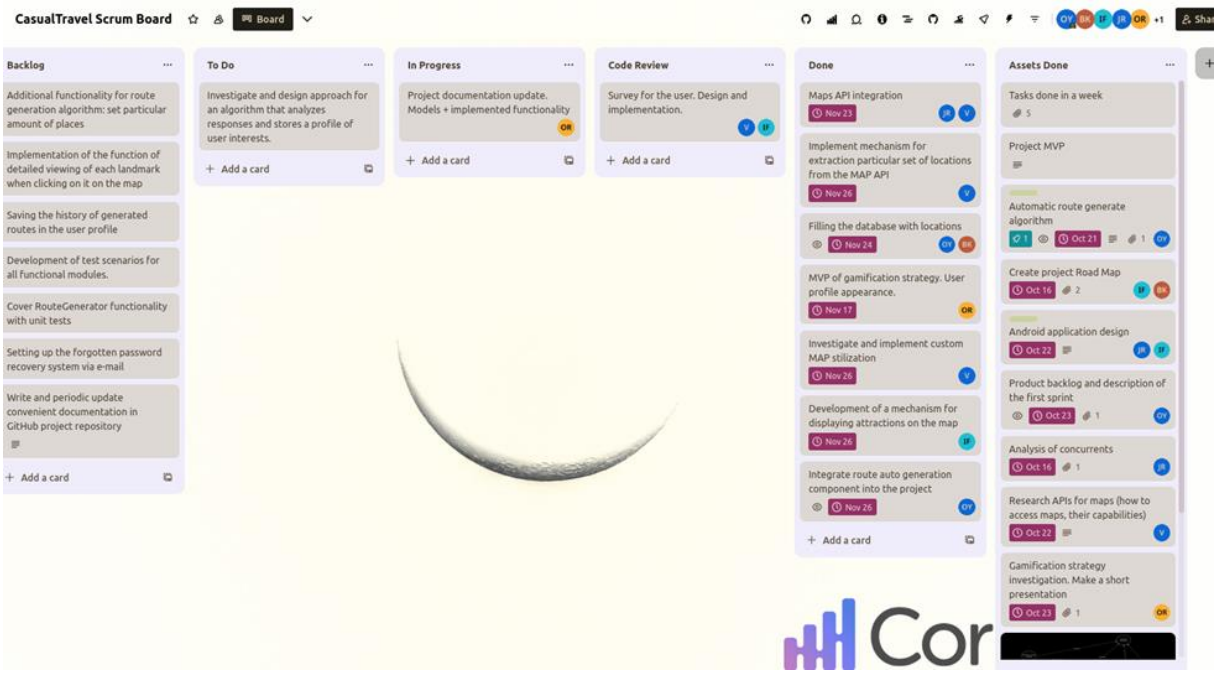


Рисунок 8 – Дошка Trello

ОПИС ПРОЦЕСУ РОЗРОБКИ

Опис команди та ролей в команді.

Команда, яка працювала над розробкою проєкту CasUAlTravel, складалася з наступних ролей:

- **Керівник, Scrum Master**: Ця роль відповідала за забезпечення ефективної роботи команди за методологією Scrum. Майстер Scrum допомагав команді вирішувати проблеми, координував спринти та дотримання Scrum-процесів.
- **Фронтенд-розробники:** Фронтенд-розробники виконували фактичну розробку користувацького інтерфейсу продукту за допомогою React та інших фронтенд технологій. Кожен фронтенд-розробник мав свою спеціалізацію та відповідав за певну частину функціональності на стороні клієнта.
- **Бекенд-розробники:** Бекенд-розробники виконували фактичну розробку серверної частини продукту на платформі Java Spring. Кожен бекенд-розробник відповідав за реалізацію бізнес-логіки, обробку запитів від фронтенду та взаємодію з базою даних.
- **Дизайнер інтерфейсів** (UI/UX Designer): Дизайнер інтерфейсів був відповідальний за створення дизайну користувачського інтерфейсу додатку в Figma, враховуючи принципи UX-дизайну та потреби користувачів.
- **Тестувальник**: Тестувальник проводив тестування продукту для виявлення помилок та недоліків. Він вносив звіти про помилки і допомагав команді виправляти їх.
- **Модератор:** Модератор відповідав за модерування контенту, що додавався користувачами, забезпечуючи відповідність правилам і стандартам платформи.
- **Експерт з документації:** Експерт з документації відповідав за створення та підтримку документації проєкту, включаючи технічну документацію, опис функціональності та інструкції для користувачів.

Опис команди проєкту:

- Ємець Олександр, група ТТП-41 - Scrum Master, модератор, бекенд розробник
- Катрич Богдан, група ТТП-41 - бекенд розробник

- Низкогуз Владислав, група ТТП-42 - фронтенд розробник
- Рак Юлія, група ТТП-41 - бекенд розробник
- Рак Олена, група ТТП-41 - експерт з документації
- Фисюк Інна, група ТТП-41 - UI/UX дизайнер, фронтенд розробник

Опис обраної методології та інструментарію.

Для розробки проєкту CasualTravel була обрана методологія Scrum. Вибір Scrum був обґрунтований потребою у гнучкому та ітеративному підході до розробки, а також можливістю працювати над проєктом в невеликих спринтах. Scrum дозволив ефективно керувати пріоритетами завдань, швидко вносити зміни та сприяв комунікації в команді.

Основні принципи Scrum, які використовувалися:

● Ітеративний підхід: Розробка відбувалася у коротких ітераціях (спринтах), кожен з яких тривав два тижні.

● Планування спринту: Щотижневе планування спринтів, визначення завдань та їх оцінка.

● Ретроспектива. По завершенню кожного спринту в команді проводилася ретроспектива, яка є важливою частиною методології Scrum.

● Спільна робота команди: Всі члени команди спільно приймали рішення та визначали пріоритети.

Планування спринту.

Планування спринту в рамках методології Scrum було однією з ключових складових розробки проєкту CasUAlTravel. Кожен спринт тривав один тиждень і розпочинався зустріччю планування, де команда визначала, які завдання будуть виконані протягом цього спринту. Основні етапи планування спринту включали:

● Визначення завдань: Команда визначала, які завдання і функціональність повинні бути реалізовані протягом спринту. Завдання формулювалися конкретно і мали чітку постановку.

● Оцінка завдань: Кожне завдання оцінювалося за складністю та тривалістю виконання. Це допомагало зрозуміти, скільки робочого часу буде витрачено на реалізацію кожного завдання.

● Визначення пріоритетів: Завданням надавались пріоритети відповідно до важливості для проєкту. Це допомагало визначити порядок виконання завдань.

● Створення беклогу: Список завдань та їх оцінки заносилися до беклогу спринту. Беклог слугував основним джерелом для відстеження прогресу та виконання завдань.

Рефлексія (Ретроспектива).

По завершенню кожного спринту в команді проводилася ретроспектива, яка є важливою частиною методології Scrum. Ретроспектива мала наступні основні етапи:

● Зібрання вражень: Кожен член команди висловлював свої враження щодо пройденого спринту, вказував на позитивні та негативні моменти.

● Аналіз результатів: Зібрані враження та відгуки аналізувалися з метою визначення, що пройшло добре і що можна покращити.

● План дій: Команда створювала план дій для покращення робочого процесу. Цей план включав конкретні заходи, які слід було б вжити для вирішення виявлених проблем.

● Заходи до покращення: Заходи, визначені на ретроспективі, реалізовувалися в майбутніх спринтах з метою покращення ефективності та співпраці в команді.

Ретроспектива дозволяла команді постійно вдосконалювати свій робочий процес, вирішувати проблеми та підтримувати високий рівень співпраці та продуктивності.

Інструментарій.

Для ефективної роботи команди та управління проєктом використовувалися наступні інструменти:

- **Trello**: Trello використовувався для ведення списку завдань, планування та відстеження прогресу виконання завдань. Кожен спринт мав свою дошку на Trello, де були розміщені завдання та їхні статуси.
- **GitHub**: GitHub використовувався для зберігання та керування версіями коду. Кожен розробник мав власну гілку для роботи над функціональністю, після чого вносив зміни через Pull Request для огляду та злиття з основною гілкою.
- **Google Meet**: Google Meet був використаний для онлайн-зустрічей та спілкування команди. Відеоконференції проводилися під час планування спринтів, обговорення важливих питань та відстеження прогресу.
- **Telegram**: Груповий чат у Telegram використовувався для швидкого обміну повідомленнями та інформацією між членами команди. Це дозволяло швидко реагувати на питання та попереджати про важливі події.

Інтеграція між цими інструментами дозволила забезпечити ефективну роботу команди та керування проєктом на всіх етапах розробки. Такий підхід сприяв спрощенню спілкування, відстеженню прогресу та забезпеченню високої продуктивності розробки.

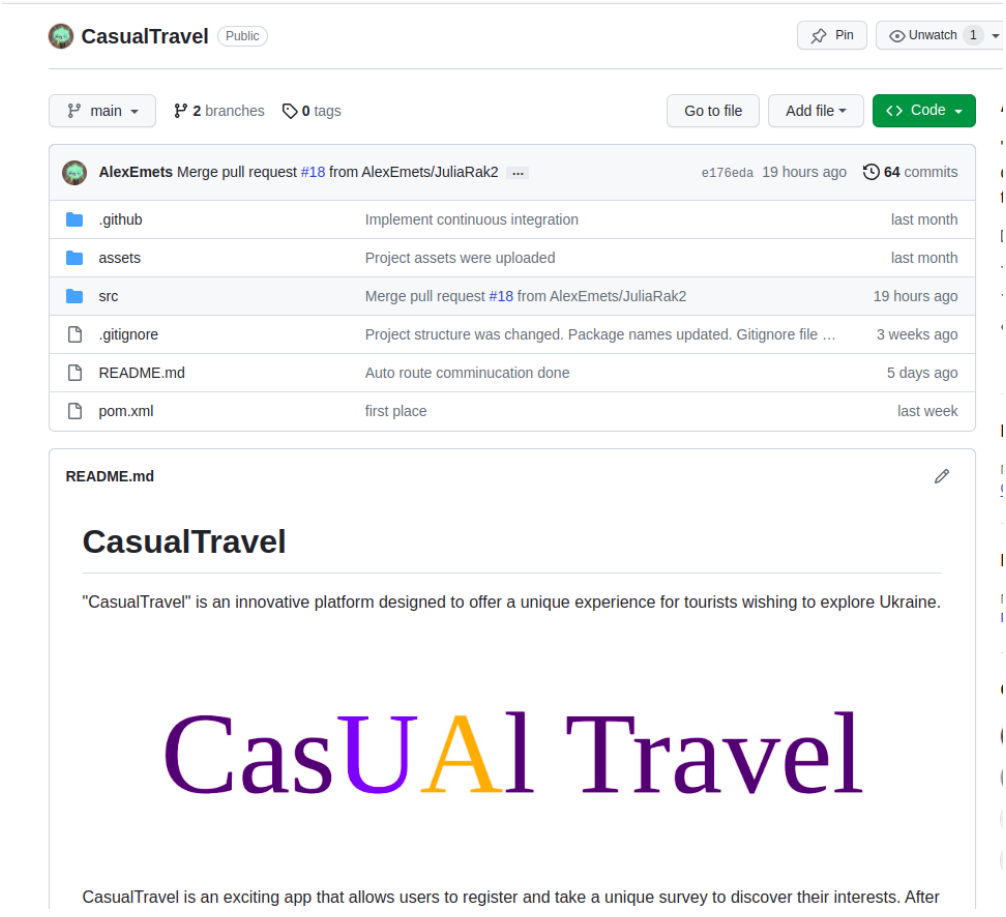


Рисунок 9 – GitHub проєкту

Чого навчилися. Під час розробки проєкту CasualTravel, команда отримала безцінний досвід і вивчила ряд важливих аспектів у сфері програмування, розробки програмного забезпечення та командної роботи. Деякі з ключових навчальних моментів включають:

● Технічні навички: Кожен член команди покращив свої технічні навички, виконуючи роль розробника бекенду або фронтенду. Вони вивчили роботу з Java Spring, React, і іншими технологіями, необхідними для створення системи.

● Робота з базою даних: Команда отримала досвід роботи з базою даних PostgreSQL, розробивши логічну структуру бази даних та взаємодіючи з нею для зберігання та отримання інформації.

● UX/UI дизайн: Деякі члени команди займалися UX/UI дизайном продукту в середовищі Figma. Вони вивчили принципи та закони UX/UI дизайну, що допомогли створити зручний та привабливий інтерфейс.

● Методологія Scrum: Команда оволоділа методологією Scrum, яка дозволила ефективно планувати, виконувати та оцінювати завдання під час спринтів.

● Рефлексія та покращення: Завдяки ретроспективам, команда навчилася визначати проблеми та вирішувати їх, постійно покращуючи свій робочий процес.

Цей проєкт надав команді можливість навчитися та покращити багато навичок, які вони в подальшому змогли використовувати у своїй професійній діяльності.

ВИСНОВКИ

Під час розробки проєкту "CasUAlTravel" команда зіткнулася з численними викликами та завданнями, які було вирішено з високою якістю. Важливі аспекти проєкту, а також оцінка їх виконання:

● Призначення системи: Визначення цілей та завдань було виконано на високому рівні. "CasualTravel" успішно вирішує проблему планування подорожей для користувачів з різними інтересами.

● Вимоги до системи: Вимоги до системи були ретельно визначені і втілені. Вони включають вимоги до функціональності, структури бази даних та інтерфейсу користувача.

● Організація інформаційної бази: Логічна структура бази даних була розроблена з урахуванням потреб системи. Використана система керування базами даних - PostgreSQL.

● UI/UX дизайн: Закони та принципи UX/UI дизайну були враховані при розробці інтерфейсу. Дизайн системи виглядає зручно та привабливо.

● Технології та імплементація: Для розробки бекенду було використано Java Spring, фронтенд реалізовано на React. Концепція Code first успішно втілена.

● RestFul API: Система використовує RestFul API для взаємодії з клієнтами та мобільним додатком.

● Ролі в команді: Команда включала розробників бекенду та фронтенду, дизайнера, модератора та експерта з документації.

● Методологія та інструментарій: Методологія Scrum була успішно використана для планування та управління розробкою. Інструментарій включав Trello, GitHub, Google Meet, і Telegram.

● Планування спринтів: Щотижневе планування спринтів та визначення завдань допомогли у вчасному вирішенні задач.

● Рефлексія та навчання: Команда проводила ретроспективи та покращувала свій робочий процес. Навчилася вирішувати труднощі та вдосконалювати продукт.

В цілому, проєкт "CasUAlTravel" був успішно реалізований, відповідаючи поставленим цілям та завданням. Команда набула величезний досвід у розробці програмного забезпечення та забезпечила користувачам зручний інструмент для планування своїх подорожей.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Java Spring project getting started. Документація. [Електронний ресурс]. Режим доступу: https://spring.io/guides/gs/spring-boot/
2. React Native introduction. Документація. [Електронний ресурс]. Режим доступу: https://reactnative.dev/docs/getting-started
3. Using Postgres Effectively in Spring Boot Applications. Документація. [Електронний ресурс]. Режим доступу: https://hackernoon.com/using-postgres-effectively-in-spring-boot-applications
4. Google Maps Platform Documentation. Документація. [Електронний ресурс]. Режим доступу: https://developers.google.com/maps/documentation
5. Алгоритм Дейкстри і його застосування. Документація. [Електронний ресурс]. Режим доступу: https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/
6. Github + Trello integration. Документація. [Електронний ресурс]. Режим доступу: https://blog.trello.com/github-and-trello-integrate-your-commits
7. Google Maps стилізація. Документація. [Електронний ресурс]. Режим доступу: https://snazzymaps.com/
8. Java Spring структура моделей. Документація. [Електронний ресурс]. Режим доступу: https://www.baeldung.com/spring-mvc-model-model-map-model-view
9. Google карта для отримання інформації щодо локацій. Документація. [Електронний ресурс]. Режим доступу: https://www.google.com/maps
10. VisitUkraine [Електронний ресурс] – Режим доступу до ресурсу: https://visitukraine.today/uk
11. WalQlike [Електронний ресурс] – Режим доступу до ресурсу: https://walqlike.com/
12. Tripio Travel App [Електронний ресурс] – Режим доступу до ресурсу: https://www.tripioapp.com/

ВЕБЗАСТОСУНОК "GARDENHUB" ДЛЯ ЗАМОВЛЕННЯ ПОСЛУГ У СФЕРІ САДІВНИЦТВА

Сергій Олійник, Вікторія Дволінська, Ярослав Ткаченко, Дмитро Прохорчук, Владислав Таран, Ольга Сірікова, Роман Козакевич, Іван Аксані, Ярослав Росновський

Проєкт "Gardenhub" створено для забезпечення зручної співпраці між власниками будинків та садівниками. У цьому варіанті реалізовано розширену версію MVP застосунку, в яку включено систему чатів, сповіщень та інші покращення. Проведено аналіз ринку, визначено стратегічні напрямки розвитку, та збагачено знання в області розробки програмного забезпечення та методологій управління проєктами. Отримані результати будуть корисними для подальшого розвитку платформи та підвищення ефективності співпраці між учасниками.

The "Gardenhub" project is aimed at facilitating convenient collaboration between homeowners and gardeners. In this research, an expanded version of the MVP application has been implemented, incorporating a chat system, notifications, and other enhancements. Market analysis has been conducted, strategic development directions have been identified, and knowledge in software development and project management methodologies has been enriched. The obtained results will be valuable for further platform development and enhancing the efficiency of collaboration between participants.

GARDENHUB, САДІВНИК, ДОМОВЛАСНИК, SIGNALR, TRELLO, WEBAPI, FRONTEND, КОМАНДНА РОЗРОБКА

GARDENHUB, GARDENER, HOMEOWNER, SIGNALR, TRELLO, WEBAPI, FRONTEND, TEAM DEVELOPMENT

ВСТУП

Актуальність обраної теми. У сучасному світі, де зростає екологічна свідомість та прагнення до комфортного життя, платформа, що з'єднує власників будинків і садівників, має велику актуальність. Вона вирішує проблеми пов'язані з ефективним доглядом за прибудинковою територією, надаючи зручний доступ до послуг садівників. Замовники отримують можливість швидко та якісно вирішувати завдання з області ландшафтного дизайну та догляду за рослинами. Це сприяє створенню приємного та естетичного оточення, а також відповідає загальному тренду активного впливу громадян на своє оточення та екологічну стабільність.

Методи дослідження. Для розробки такого застосунку методи дослідження включають аналіз потреб та проблем, з якими стикаються різні категорії користувачів, аналіз конкурентів, опитування та спостереження, тестування та оцінка ефективності застосунку.

Мета і завдання роботи. Метою роботи є розробка застосунку, який є зручним інструментом для онлайн-замовлень у сфері садівництва. Ключовою особливістю додатку є зрозумілий та швидкий інтерфейс. Для досягнення цієї мети було поставлено наступні завдання:

- вивчення існуючих на ринку застосунків для замовлення послуг;
- огляд та вибір існуючих технологій для проєктування та реалізації застосунку;
- проєктування інтерфейсу та архітектури додатку з акцентом на зручність використання для обох сторін - замовників та садівників;

● реалізація, тестування та впровадження розробленого застосунку для ефективного взаємодії учасників платформи.

ПОШУК ТА ВАЛІДАЦІЯ ІДЕЇ

Опис пошуку ідеї продукту.

Процес пошуку ідей для розробки стартапу є критичним етапом у становленні успішного підприємства. Важливість цього процесу визначається кількома ключовими аспектами:

● ринковий потенціал: правильно обрана ідея стартапу повинна відповідати реальним потребам цільової аудиторії. Пошук ідей дозволяє виявити невирішені проблеми або покращення в існуючих ринкових сегментах, що сприятиме вдалому входженню на ринок

● інновації та конкурентоспроможність високий рівень конкуренції у бізнес-середовищі вимагає унікальності та інноваційності ідеї. Пошук ідей допомагає виявити та розвинути конкурентні переваги, що стануть основою для успіху

● фінансова привабливість: важливо оцінювати фінансовий потенціал обраної ідеї. Пошук ідей дозволяє аналізувати ринкові тенденції, попит та потенційний дохід

● можливість масштабування: ідея стартапу повинна мати потенціал для масштабування, щоб забезпечити стійкий розвиток підприємства.

Розуміючи дані фактори було проведено брейнштормінг, де команда обговорила наступні ідеї:

1. Платформа для волонтерства.

Сфера ідеї: Civic tech (громадські технології) — це IT-рішення, що спрощують комунікацію між державою, бізнесом і громадськістю та залучають громадян до спільного вирішення проблем місцевого чи національного значення.

Ідея в тому, щоб створити платформу, де користувачі зможуть підтримувати актуальні збори в країні. Для того, щоб розмістити збір платформі, волонтер/військовий має зареєструватись, надати документи/підтвердження тобто пройти верифікацію. Після того як відбудеться перевірка, збір буде додано на платформу та користувачі матимуть змогу робити донати. SWOT-аналіз даної ідеї наведено на рис. 1.

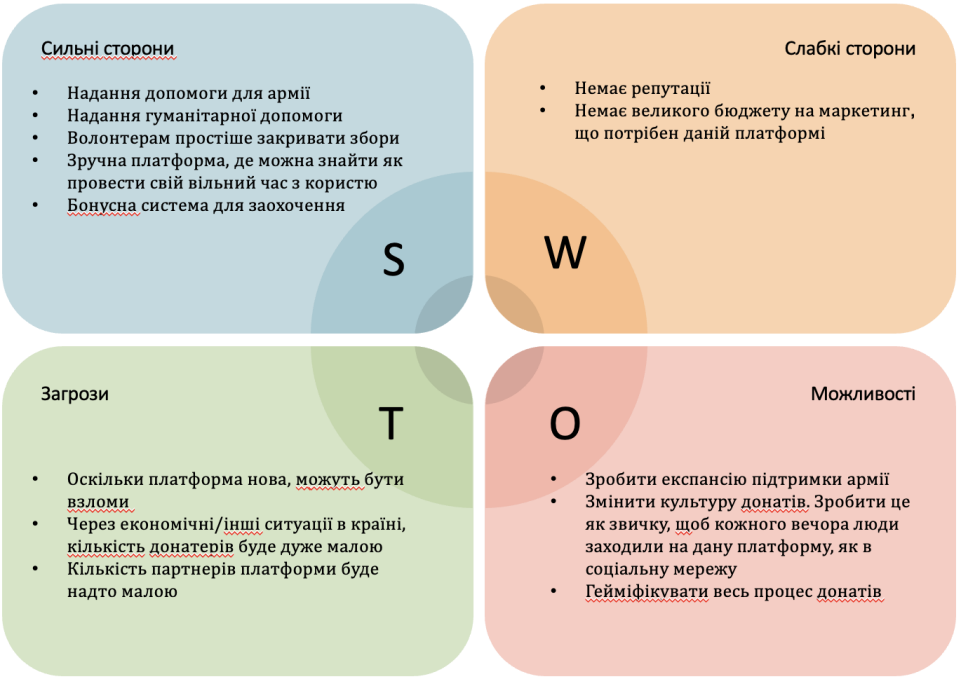


Рисунок 1 – SWOT-аналіз волонтерської платформи

2. Платформа для подарунків.

Сфера ідеї: Entertainmenttech, others.

Ідея полягає в тому, щоб створити платформу, де люди зможуть робити один одному подарунки. Наприклад, ви хочете віддячити людині, що вам якось допомогла, при цьому, ви цю людину можете не знати. Вам достатньо знати її номер телефону чи нікнейм, щоб зробити їй подарунок. Це може бути кава, солодощі, електроніка, тощо. SWOT-аналіз даної ідеї наведено на рис. 2.

Рисунок 2 – SWOT-аналіз платформи для подарунків

3. Симулятор для інвестування.

Сфера ідеї: Fintech, Edtech

Ідея заключається в тому, щоб створити платформу, де люди зможуть навчитись грамотному інвестуванню в акції компаній/криптовалюти тощо.

SWOT-аналіз даної ідеї наведено на рис. 3.

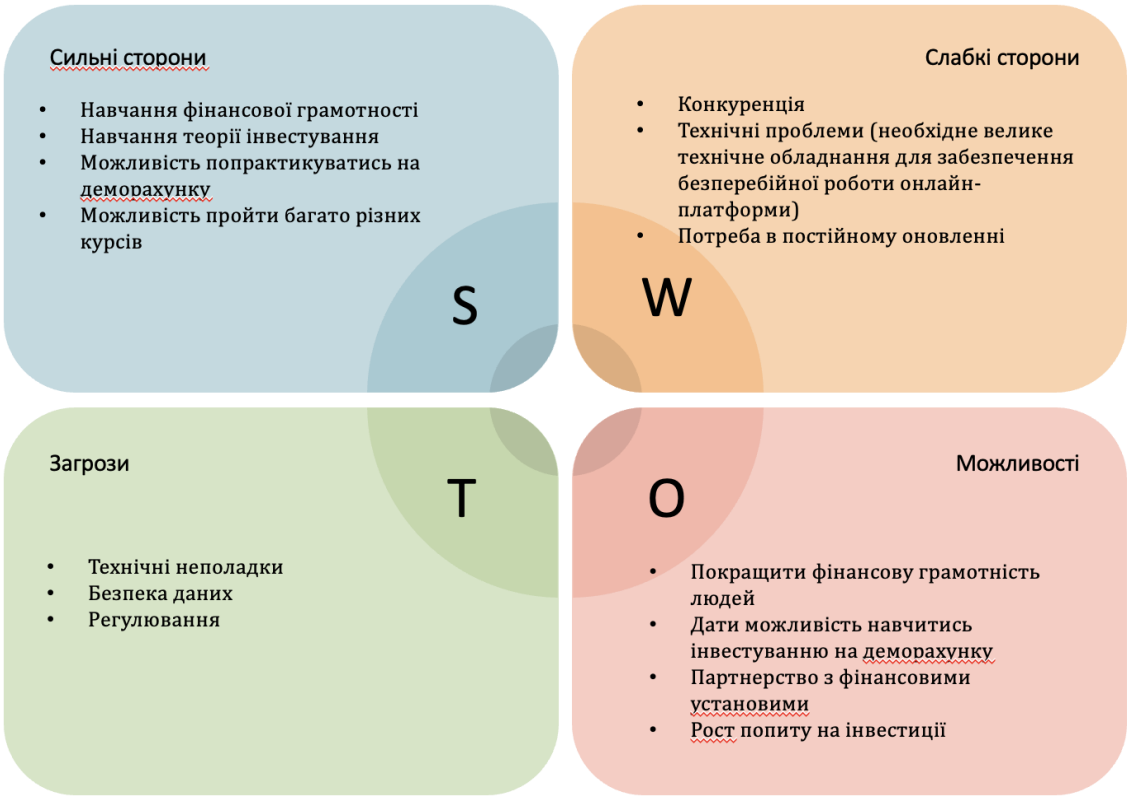


Рисунок 3 – SWOT-аналіз симулятора для інвестування

4. Застосунок для замовлення послуг у сфері садівництва.

Сфера ідеї: Gardening, others.

Ідея полягає в тому, щоб створити платформу, що об’єднує власників будинків з професіоналами садівниками для взаємодії на прибудинковій території замовників.

Створення платформи, що об'єднує власників будинків і професіональних садівників, може сприяти покращенню догляду за прибудинковою територією та створенню більш комфортного середовища для мешканців. В епоху зростаючої екологічної свідомості та бажання досягти комфортного способу життя, платформа, що об'єднує власників будинків та садівників, стає вкрай актуальною. Ця ініціатива вирішує проблеми, пов'язані з ефективним доглядом за прибудинковою територією, надаючи зручний доступ до послуг садівників. Замовники отримують можливість швидко та якісно вирішувати завдання у сфері ландшафтного дизайну та догляду за рослинами. Це сприяє створенню приємного та естетичного оточення, відповідаючи загальному тренду активного впливу громадян на своє оточення та екологічну стабільність.

Проаналізувавши всі ідеї та зробивши аналіз переваг і недоліків кожної, ми відкинули всі ідеї, окрім застосунку для замовлення послуг у сфері садівництва. Нам сподобалась як сфера, так і сама ідея, адже ми вирішуємо реальну потребу цільової аудиторії. Також обрана ідея має гарну фінансову привабливість.

Огляд наявних на ринку IT-продуктів – конкурентів

1) Mapus [7]

Mapus.ua — це соцмережа послуг, яка допомагає знайти спеціалістів або замовників в себе на районі. Вирішує наступні потреби та болі цільової аудиторії: надає необхідні послуги користувачам та збільшує дохід майстрів/виконавців послуг.

Mapus не бере % від замовлень, а має дохід від рекламних інтеграцій.

Загальна кількість візитів щомісячно згідно SimilarWeb 3.9K. Mapus має застосунок для Android та IOS. Загальна кількість завантажень застосунку для Android 10-50K (Sensotower). Mapus присутні в Instagram (184 підписники) та Facebook (494 підписники). Згідно Google аналітики та проведеного аналізу найчастіше звертаються до цього сервісу за допомогою в наступних сферах: автодопомога, ремонт та будівництво, догляд за тваринами, кур'єрські послуги, допомога по дому та саду. Найбільше користувачів в Україні заходять на Mapus з Черкаської області.

2) Kabanchik [8]

Kabanchik.ua — онлайн сервіс пошуку приватних фахівців для вирішення побутових та бізнес завдань. Майданчик об'єднує замовників послуг, яким необхідно виконати яку-небудь роботу, та компетентних фахівців, які шукають підробіток або додатковий заробіток. Виконавці сплачують комісію сервісу. Раніше мінімальна комісія була на рівні 10%. У 2019 році її диференціювали залежно від категорії робіт та міста. Нині вона становить від 5 до 20%. Загальна кількість візитів щомісячно згідно SimilarWeb 1.4M. Kabanchik має застосунок для Android та IOS. Загальна кількість завантажень застосунку для Android 500K-1M (Sensotower).

Kabanchik присутні в Instagram (6526 підписників) та Facebook (20K підписників). Найбільше користувачів в Україні заходять на Kabanchik з Києва та Львова.

3) Rabotniki ua [9]

Сайт Rabotniki ua це найбільший будівельний сайт України, який допомагає приватним замовникам знайти будівельників, а будівельникам допомагає отримати нові замовлення та заробляти. Також і замовники, і будівельники можуть дивитись розцінки на всі види робіт в Україні. Замовникам це допомагає торгуватись, а Будівельникам правильно формувати ринкову ціну на свої послуги.

Монетизація сайту заснована на підписках. Будівельники можуть платно отримувати доступ до контактів замовників на умовах PRO пакета, займати перші три місця в каталозі будівельників і піднімати свою сторінку.

Загальна кількість візитів щомісячно згідно SimilarWeb 738K.Дана платформа крім веб версії має застосунок для Android та IOS. Загальна кількість завантажень застосунку для Android 10K-50K (Sensotower). Rabotniki ua присутні в Instagram (540 підписників) та Facebook (101K підписників).Найбільше користувачів в Україні заходять на Rabotniki ua з Києва та Харкова.

4) Gigsmart [10]

GigSmart — це інноваційна технологічна компанія, яка об’єднує підприємства з кваліфікованими працівниками на вимогу для короткострокових і довгострокових робіт у всіх галузях промисловості по всій країні. Компанія представлена на ринку Сполучених Штатів Америки.

GigSmart бере 25% з кожного успішного замовлення робітника. Також GigSmart має платні підписки для створення оголошень.

Загальна кількість візитів щомісячно згідно SimilarWeb 85K. Дана платформа крім веб версії має застосунок для Android та IOS. Загальна кількість завантажень застосунку для Android 500K-1M (Sensotower). GigSmart присутні в Instagram (2620 підписників) та Facebook (1.5K підписників).

На рис. 4 зображена порівняльна таблиця усіх конкурентів.

	Кількість кор.	% від Замовлен.	Наявність платних підписок	Наявність мобільного додатку	Кількість завантажень Google Play	Наявність відгуків	Цільова аудиторія вік	Наявність інших послуг окрім садів.	Наявність карти
MAPUS	3.9K	0	-	+	10-50k	+	-	+	+
KABANCHIK	1.4KK	5-20	+	+	500-1kk	+	25-34	+	-
RABOTNIKI	738K	-	+	+	10-50k	+	35-44	+	-
GIGSMART	85K	25%	+	+	50—1kk	+	25-34	+	-
ДОМАШНІК САДІВНИК	-	0	+	+	-	+	…	-	+

Рисунок 4 – Порівняльна таблиця конкурентів

Підсумовуючи, дещо планується запозичити у конкурентів, наприклад наявність карти у Mapus, але деякі речі у нас будуть унікальними, наприклад: таск-менеджер та система рекомендацій.

Мапа продукту

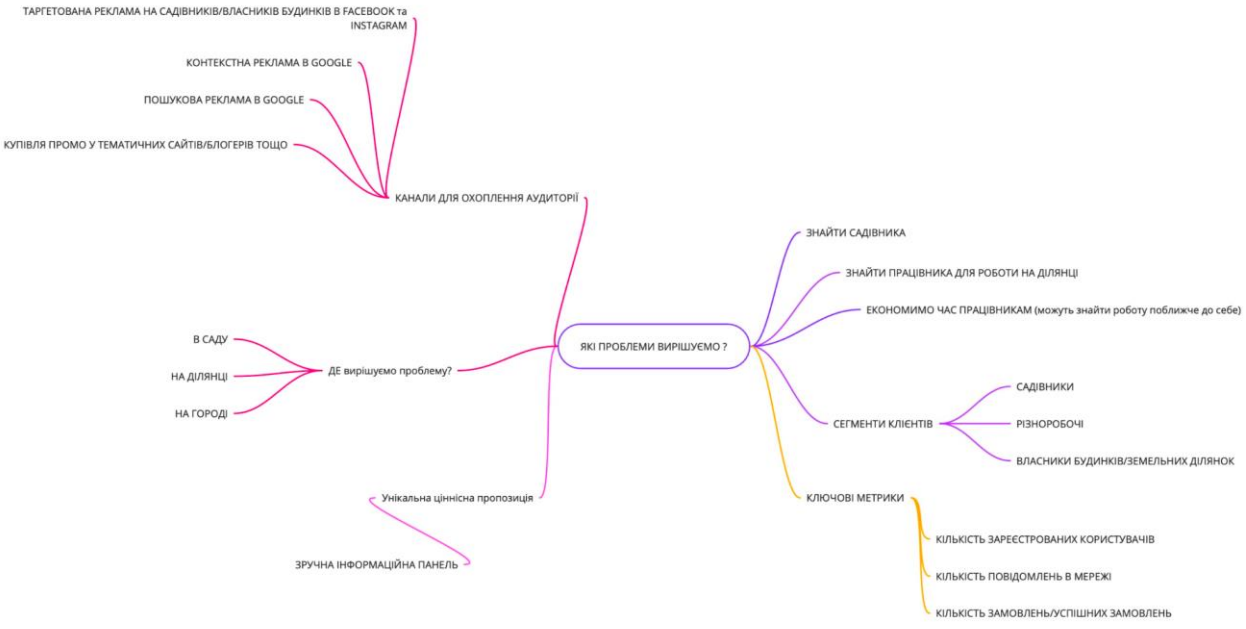


Рисунок 5 – Мапа продукту

Наш продукт вирішує одразу декілька проблем, а саме допомагає власникам земельних ділянок знайти професійну та компетентну людину для вирішення певних задач

по садівництву та догляду за земельною ділянкою. Відповідно, садівникам професіоналам допомагаємо знайти замовлення.

Ми локалізуємо сферу послуг. Більшість послуг прив'язані до місця проживання. Замовники, бажають, щоб потрібний фахівець був поблизу. А виконавці мріють, щоб замовлення були "на районі ", знижуючи витрати на дорогу і обслуговуючи більшу кількість клієнтів протягом робочого дня. Саме цю проблему ми і вирішуємо.

Можна виділити декілька сегментів клієнтів: садівники, різноробочі, та власники будинків/земельних ділянок. Можемо зробити гіпотезу, що садівники це люди з великим досвідом, тож коли будемо налаштовувати рекламу, можна спробувати показувати рекламу людям 30+ та використовувати Facebook замість Instagram, де може знаходитись потенційно більш цільова аудиторія.

Ми також маємо унікальну ціннісну пропозицію − зручну інформаційну панель, систему рекомендацій та пропонуємо економію грошей, адже наш сервіс буде брати менший відсоток комісійних з замовлень, ніж у наших конкурентів, а деякий час після виходу на ринок не буде брати жодних відсотків для охоплення більшої аудиторії.

Канали охоплення аудиторії будуть, як онлайн, так і офлайн. Серед онлайн каналів буде таргетована реклама на садівників та власників будинків в соціальних мережах, а саме Facebook, Instagram, Tik Tok. Також пошукова та контекста реклама в Google. Не менш важливо купувати рекламу у тематичних каналів в соціальних мережах. Серед офлайн каналів можна використовувати зовнішню рекламу, флаєри, тощо.

При розробці необхідно врахувати, що будуть постійні витрати на сервер, щомісячні гонорари та зарплати робітникам, витрати на офлайн та онлайн рекламу.

Для вимірювання ефективності бізнесу необхідно враховувати ключові метрики, такі як: конверсію відвідувачів в клієнтів, органічний та платний трафік. Не менш важливо оцінювати залучення користувачів: кількість нових користувачів та ретеншн. Також, стежити за розвитком та залученням трафіку з соціальних мереж. Для цього потрібно стежити за кількістю підписників, вподобань, коментарями, тощо.

Сильні і слабкі сторони розроблюваного продукту.

На рис. 6 можна побачити SWOT-аналіз обраної ідеї.

Рисунок 6 – SWOT-аналіз обраної ідеї

ОПИС ПРОДУКТУ

Призначення застосунку.

Призначенням застосунку "GardenHub" є автоматизація процесу замовлення послуг садівників та догляду за прибудинковою територією. Застоунок дозволяє користувачам створювати запити на різноманітні садові роботи, переглядати роботи фахівців, а також відстежувати та змінювати статуси їхніх замовлень. Головною метою є забезпечення зручності та ефективності у взаємодії між замовниками та садівниками, сприяючи створенню доглянутих та естетичних прибудинкових територій. Робота передбачає аналіз аналогічних платформ, визначення функціональностей, проєктування та програмну реалізацію застосунку.

Цілі створення продукту

Цілі створення застсунку "GardenHub" включають:

- забезпечення ефективного збору даних про замовлення послуг садівників
- сприяння оптимальному вибору садівників для конкретних завдань зі сторони замовників
- реалізація можливості проаналізувати ефективність та обсяги виконаних робіт для подальшого вдосконалення сервісу та задоволення потреб користувачів

Застосунок також призначений для створення доглянутих та естетичних прибудинкових територій, сприяючи покращенню якості життя користувачів.

Вимоги до застосунку.

Вимоги до застосунку в цілому.

- забезпечення максимальної зручності для користувачів, зокрема, інтуїтивно зрозумілий та легко навігований інтерфейс
- розробка дизайну, який гармонійно поєднує стилістику, кольори, шрифти та логотип, створюючи єдиний та естетичний образ
- забезпечення адаптивного дизайну для оптимального відображення та функціонування на різних пристроях, включаючи мобільні
- сприяння ефективній взаємодії між замовниками та садівниками, зокрема шляхом зручної системи замовлень
- забезпечення високої якості роботи системи та оптимізація швидкодії, зокрема під час завантаження сторінок
- врахування питань безпеки для захисту конфіденційності користувачів та їхніх даних, включаючи ефективні методи автентифікації та збереження інформації
- використання сучасних технологій розробки для підтримки функціональності та можливості легкої інтеграції з іншими платформами чи сервісами

Вимоги до структури та функціонування застосунку

Програмний продукт повинен мати архітектуру на основі сучасних технологій для зберігання, обробки, аналізу та доступу до даних; забезпечувати одночасну роботу кількох користувачів. Функціонально розробка повинна бути вебзастосунком та уніфікованою з точки зору програмної та апаратної платформи. Розробка повинна забезпечувати єдиний та зручний, як простий та інтуїтивно зрозумілий, інтерфейс користувача.

В Застосунку передбачається виділити наступні функціональні підсистеми:

- підсистема професіонала (садівника)
- підсистема домовласника

Вимоги до функцій, які виконуються застосунком.
Підсистема професіонала (таб.1).
Таблиця 1. Перелік функцій, задач що підлягають автоматизації

Функція	**Задача**
Реєстрація	Введення імені, прізвища
	Введення імені користувача
	Введення електронної адреси
	Придумати пароль
Авторизація	Введення коректної електронної адреси
	Введення паролю, який був записаний на початку реєстрації
Головна сторінка	Налаштувати профіль
	Пошук замовлення
	Переглянути ТОП-садівників
	Переглянути усі активні замовлення
	Фільтрація замовлень за локацією та категорією
Налаштування профілю	Додати основну інформацію
	Налаштувати портфоліо
	Обрати види послуг
Додати основну інформацію	Додати номер телефону
	Додати головне фото
	Ввести дату народження
	Обрати міста обслуговування
	Написати інформацію про себе
Налаштування портфоліо	Переглянути додані проєкти та за потреби видалит з портфоліо
	Додати новий проєкт
Створення нового проєкту для портфоліо	Ввести заголовок
	Обрати категорії проєкту
	Ввести локацію
	Ввести вартість проєкту у грн
	Додати опис
	Додати фото
Деталі активного замовлення	Переглянути повну інформацію по замовленню
	Відгукнутись на замовлення
	Перейти на сторінку автора замовлення
Деталі виконаного замовлення	Переглянути повну інформацію по замовленню
	Додати до портфоліо
	Перейти на сторінку автора

Підсистема домовласника (таб.2).

Таблиця 2. Перелік функцій, задач що підлягають автоматизації

Функція	**Задача**
Реєстрація	Введення імені, прізвища
	Введення імені користувача
	Введення електронної адреси
	Придумати пароль
Авторизація	Введення коректної електронної адреси
	Введення паролю, який був записаний на початку реєстрації
Головна сторінка	Переглянути ТОП-садівників
	Створити замовлення
Налаштування профілю	Змінити наявні дані
	Добавити фото, номер та дату народження
Створення нового замовлення	Додати заголовок
	Обрати категорії замовлення
	Обрати локацію
	Ввести вартість замовлення
	Додати опис
	Прикріпити фото
Деталі активного замовлення	Переглянути повну інформацію по замовленню
	Видалити замовлення
	Перейти на сторінку автора
Сповіщення	Переглянути профіль кандидата
	Відхилити/підтвердити запит на виконання завдання
Деталі замовлення "в роботі"	Переглянути повну інформацію по замовленню
	Перейти на сторінку виконавця
	Перейти на сторінку автора
	Відмітити як виконане -> залишити відгук про виконавця
Сторінка "Мої замовлення"	Переглянути активні/"в роботі"/виконані замовлення

Системні вимоги.

Браузери: Сайт повинен коректно відображатися в інтернет браузерах

MS Internet Explorer 5.5 і вище, Mozilla 1.7 і вище, Opera 7.54 і вище.

На довільні некоректні дії користувача, пов’язані з введенням невірних даних, не заповненням обов’язкових полів введення в формах та інші, які можуть бути оброблені системою, генеруються відповідні повідомлення про помилки українською мовою, в межах загального дизайну сайту.

Джерелом даних для системи повинна бути інформаційна система (СУБД SQL Server Management Studio 2022). Операційна система: Windows 10.

Опис організації інформаційної бази.

Логічна структура бази даних.

База даних складається з двох окремих логічних кластерів - AspNet Identity БД та основна БД з сутностями предметної області. Розглянемо першу частину на рис. 7, таблиця 3:

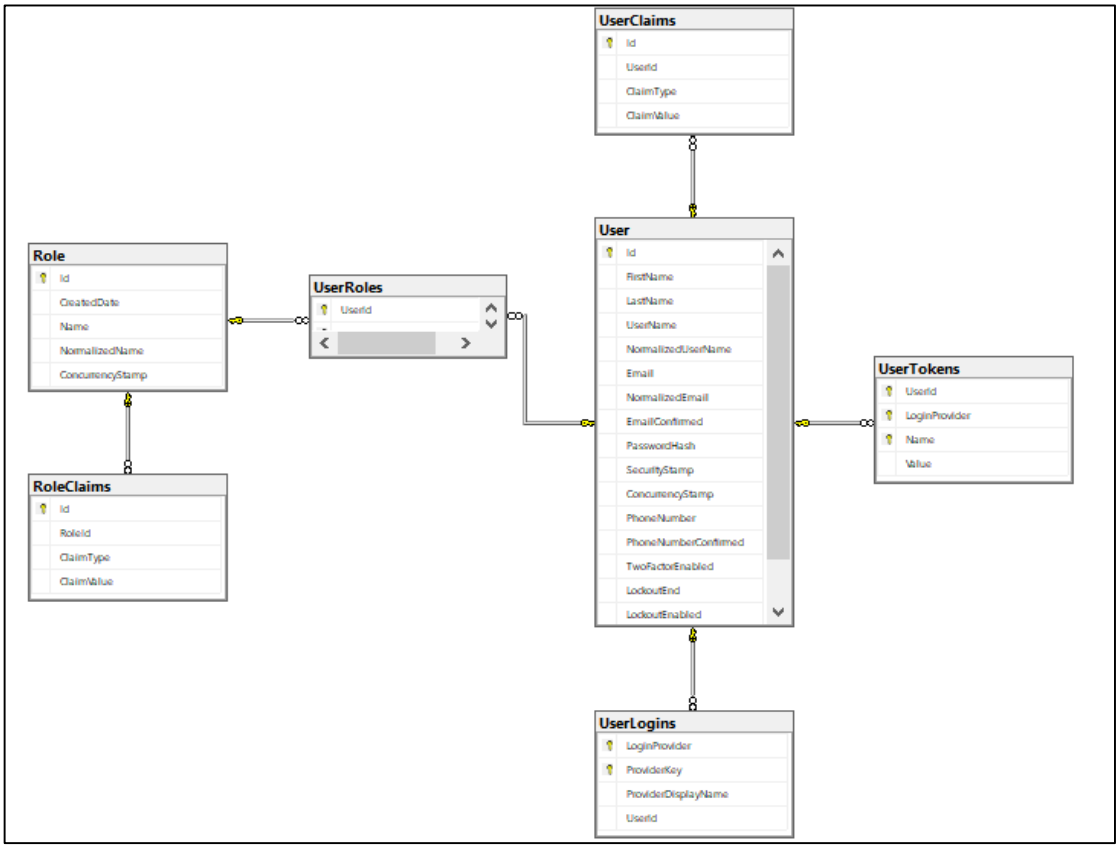


Рисунок 7 – AspNet Identity БД

Таблиця 3. Опис таблиць.

Номер	Таблиця	Опис
1	Users	Зберігає основну інформацію про користувачів, таку як ідентифікатор (Id), ім'я користувача (UserName), електронна пошта (Email), хеш пароля (PasswordHash), інформація про блокування та інше.
2	UserClaims	Містить дані про додаткові атрибути користувача в форматі (тип, значення). Ці атрибути можуть використовуватися для зберігання додаткових інформаційних даних про користувача.
3	UserTokens	Забезпечує механізм для зберігання маркерів аутентифікації, таких як токени оновлення, які використовуються для підтвердження ідентичності користувача.
4	UserLogins	Зберігає дані про зовнішні сервіси аутентифікації (наприклад, Google, Facebook), які користувач використовує для входу.
5	UserRoles	Визначає відносини між користувачами та ролями. Допомагає визначити, які права доступу має користувач.

6	Roles	Зберігає список ролей, які можуть бути використані для надання прав доступу користувачам.
7	RoleClaims	Містить атрибути (тип, значення) для ролей. Це дозволяє додавати додаткові атрибути для кожної ролі.

Розглянемо основні сутності, необхідні для опису предметної області та зв'язки між ними на рис. 8, таблиця 4:

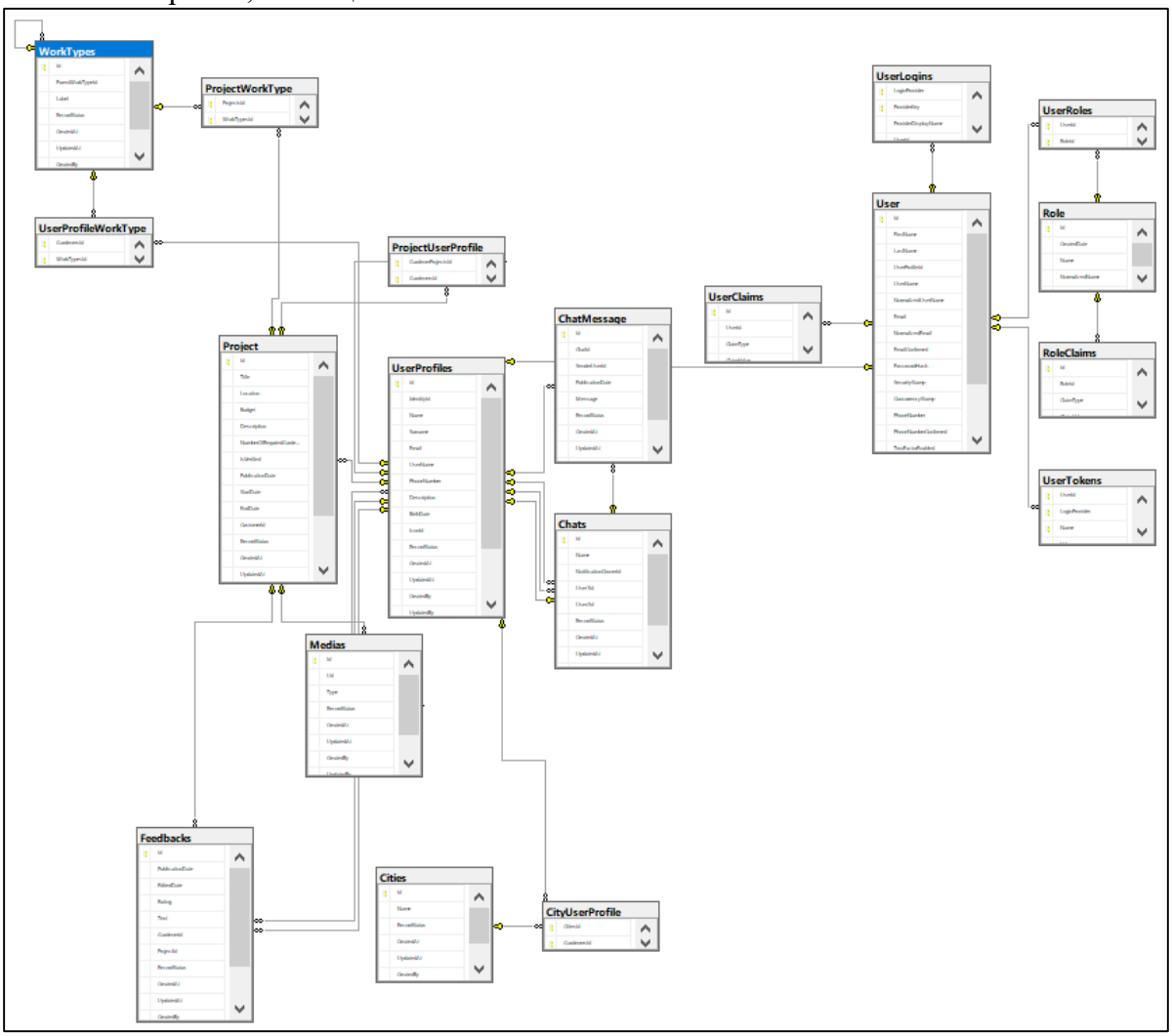


Рисунок 8 – Основна БД

Таблиця 4. Опис таблиць основної БД.

Номер	**Таблиця**	**Опис**
1	UserProfiles	Загальна сутність, яка описує поля спільні для сутностей Gardener та Customer, такі як Name, Email, PhoneNumber etc.
2	GardenerProfiles	Розширення сутності User, яке зберігає інформацію, яка відноситься до досвіду садівника.
3	CustomerProfiles	Розширення сутності User, яке зберігає інформацію, яка відноситься до замовника.

Номер	Таблиця	Опис
4	Medias	Сутність для збереження посилань на медіа файли - малюнки та відео.
5	Projects	Сутність, яка відповідає проєкту - містить в собі особу замовника, особи виконавців, тип робіт etc.
6	Feedbacks	Сутність відгуку на садівника. Зберігається з унікальними Project та GardenerProfile.
7	GardenerProfileProject	Таблиця для відношень many-to-many серед таблиць GardenerProfiles та Projects.
8	WorkTypes	Типи можливих робіт для садівників. Містить generl та specific рівні вкладенності.
9	GardenerProfileWorkType	Таблиця для відношень many-to-many серед таблиць GardenerProfiles та WorkTypes.
10	Cities	Таблиця переліку міст, які використовуються у нас в системі.
11	GardenerProfileCity	Таблиця для відношень many-to-many серед таблиць GardenerProfiles та Cities.

Опис таблиць представлено в таблицях 5-8.

Таблиця 5. Базові поля

Атрибут	Тип	Опис
Id	bigint	Ідентифікатор запису
RecordStatus	enum	Стан запису
CreatedAt	datetime2(7)	Час створення
CreatedBy	nvarchar(max)	Ім’я користувача, хто створив
UpdatedAt	datetime2(7)	Час зміни
UpdatedBy	nvarchar(max)	Ім’я користувача, хто змінив

Таблиця 6. UserProfiles

Атрибут	Тип	Опис
IdentityId	uniqueidentifier	Foreign Key до кластера AspNet Identity
IconId	bigint	Foreign Key до таблиці Medias
Name	nvarchar(max)	Ім’я користувача
Surname	nvarchar(max)	Прізвище користувача
Email	nvarchar(max)	Електронна пошта користувача
UserName	nvarchar(max)	Ім’я користувача в системі
PhoneNumber	nvarchar(max)	Телефон користувача
Description	nvarchar(1000)	Вільний текст про користувача
BirthDate	datetime2(7)	Дата народження користувача
CustomerProfileId	bigint	Foreign Key до таблиці CustomerProfiles

GardenerProfileId	bigint	Foreign Key до таблиці GardenerProfiles

Таблиця 7. Projects

Атрибут	**Тип**	**Опис**
Title	nvarchar(max)	Назва проєкту
Location	nvarchar(max)	Назва міста, де може бути виконан робота
Budget	decimal(1,2)	Бюджет проєкту (в USD)
Description	nvarchar(max)	Опис проєкиу, де можуть бути як
NumberOfRequriedGardeners	int	Кіл-сть необхідних садівників для проєкту
IsCompleted	bit	Чи завершений проєкт
IsVerified	bit	Чи верифіковано проєкт
Description	nvarchar(1000)	Опис проєкту в деталях
PublicationDate	datetime2(7)	Дата створення проєкту
StartDate	datetime2(7)	Дата початку робіт
EndDate	datetime2(7)	Дата кінця робіт
CustomerId	bigint	Foreign Key для таблиці CustomerProfines

Таблиця 8. WorkTypes

Атрибут	**Тип**	**Опис**
GeneralType	nvarchar(max)	Загальний тип робіт
SpecificType	nvarchar(max)	Підтип роботи
ProjectId	bigint	Foreign Key для таблиці Projects

UI/UX-дизайн продукту

Закони та принципи UX-дизайну, що були застосовані до створюваного продукту:

● Закон Якоба: "Користувачі проводять більшу частину свого часу в інших програмах. Тому вони хочуть, щоб ваша програма працювала подібно іншим, які вони вже знають".

У нашому дизайні меню локалізовано в лівому кутку для зручного й звичного доступу користувачів. Воно легко впізнаване, відповідаючи шаблону, що полегшує швидкий доступ до різних розділів. Крім того, впроваджено зручний перемикач ролей - від домовласника до садівника. Такий підхід сприяє інтуїтивному взаєморозумінню для користувачів, забезпечуючи знайомий та ефективний досвід взаємодії з вебсайтом. Постійність у розміщенні елементів дозволяє користувачам легко зорієнтуватися та виконати необхідні завдання, враховуючи їхні звички та очікування (рис. 9).

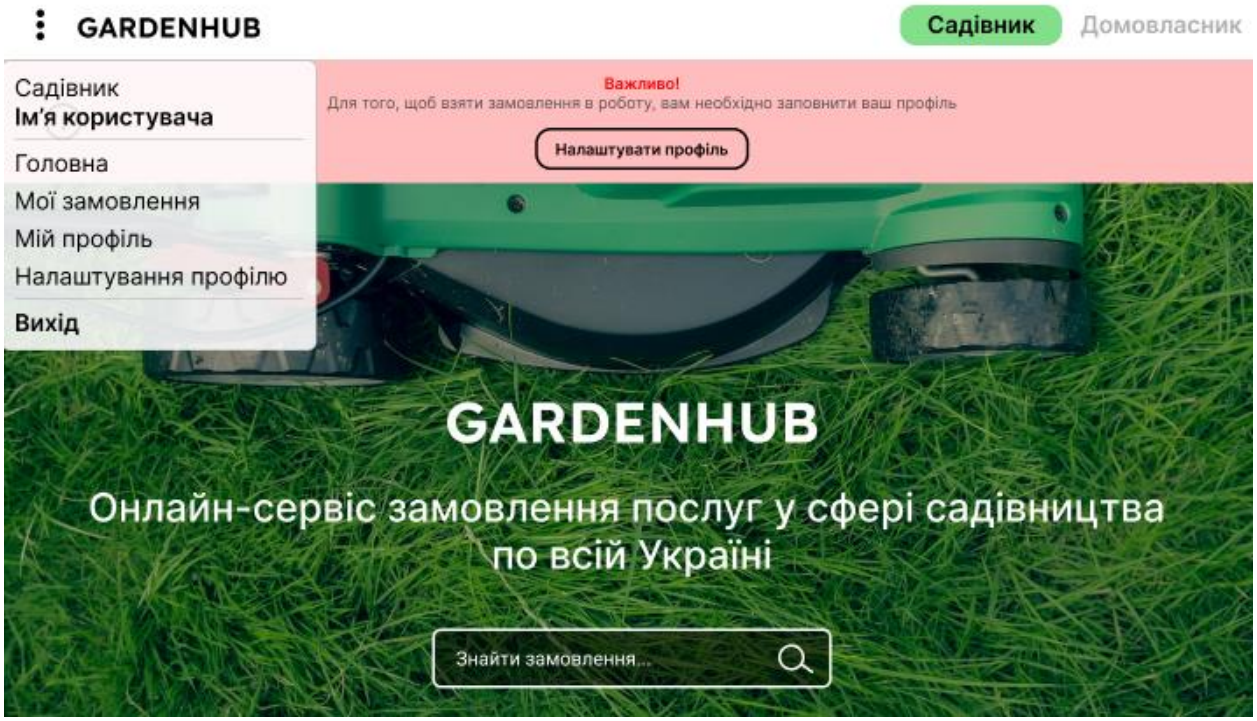


Рисунок 9 – Головна сторінка сайту

● Закон Міллера: "Людина може утримати у своїй короткочасній пам'яті лише 7±2 елементів. Якщо їх більше, мозок буде перевантажений".

Враховуючи цей закон, головна сторінка обмежується лише 7 топ-садівниками (рис. 10). Це стратегічне рішення допомагає уникнути перевантаження користувача і сприяє оптимальному сприйняттю інформації. Представлення лише найважливіших елементів полегшує користувачам зберігання та відтворення цільової інформації у своїй короткочасній пам'яті. Такий підхід максимізує зручність взаємодії, дозволяючи користувачам швидко та ефективно здійснювати вибір серед найкращих спеціалістів.

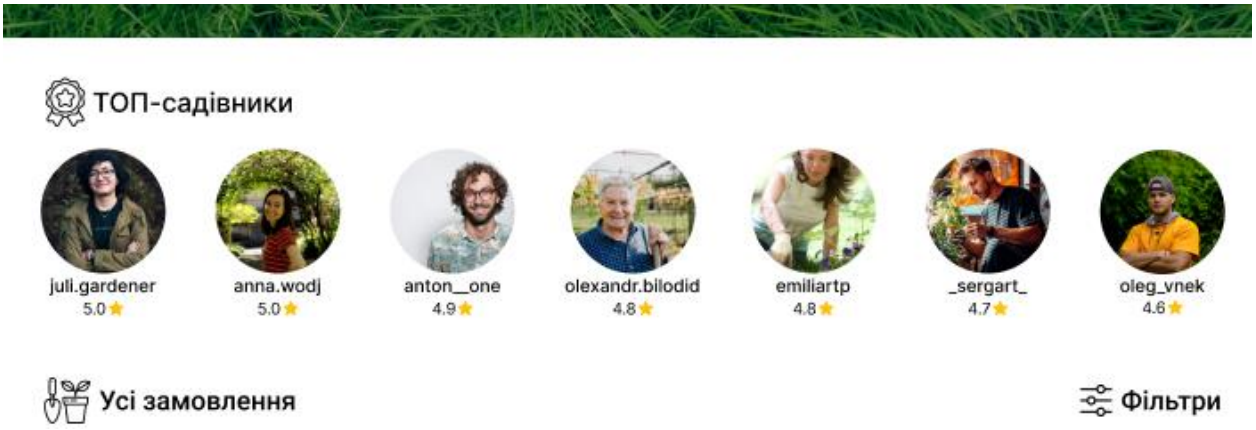


Рисунок 10 – Топ-садівники

● Закон Хіка: "Час, необхідний для прийняття рішення, збільшується зі збільшенням кількості та складності вибору".

В налаштуваннях профілю садівника використано принцип закону Хіка, об'єднуючи всю інформацію в трьох обмежених блоках: "Основна інформація", "Портфоліо", "Види послуг" (рис. 11). Зменшення кількості блоків спрощує вибір користувача та зменшує ймовірність перевантаження. Кожен блок стає фокусованим елементом, де користувач може концентруватися на конкретній інформації. Це полегшує процес взаємодії, дозволяючи користувачу швидко знаходити та обирати необхідну інформацію.

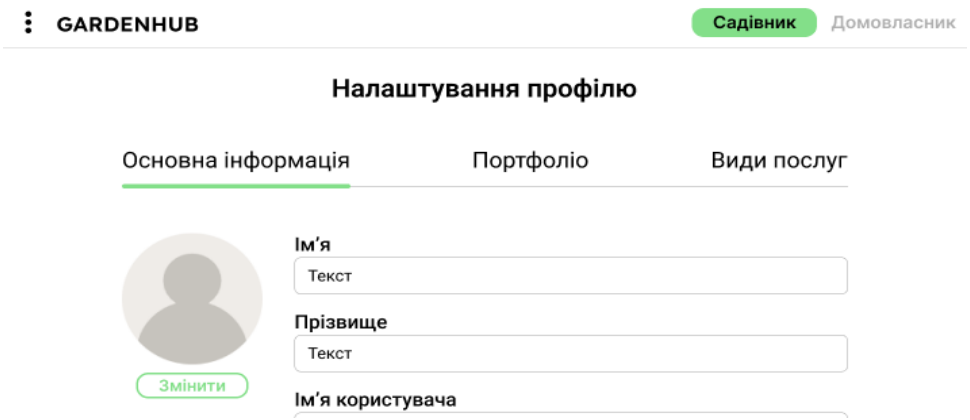


Рисунок 11. Налаштування профілю садівника

● Принцип близькості: "Елементи, що розташовані близько один від одного, сприймаються більш пов’язаними, ніж ті, що знаходяться на деякій відстані".

У нашому дизайні принцип близькості використовується для ефективної організації інформації на сторінці садівника. Основна інформація про нього розташована поруч, утворюючи логічні блоки. Близькість цих елементів вказує на їхню взаємодію та взаємозв’язок. Простір між блоками дозволяє створити чітку візуальну ієрархію та визначити їхню важливість. Такий підхід допомагає користувачам легко здійснювати сканування та знаходити потрібну інформацію, покращуючи загальний вигляд інтерфейсу та сприяючи швидшому досягненню їхніх цілей у взаємодії з платформою (рис. 12).

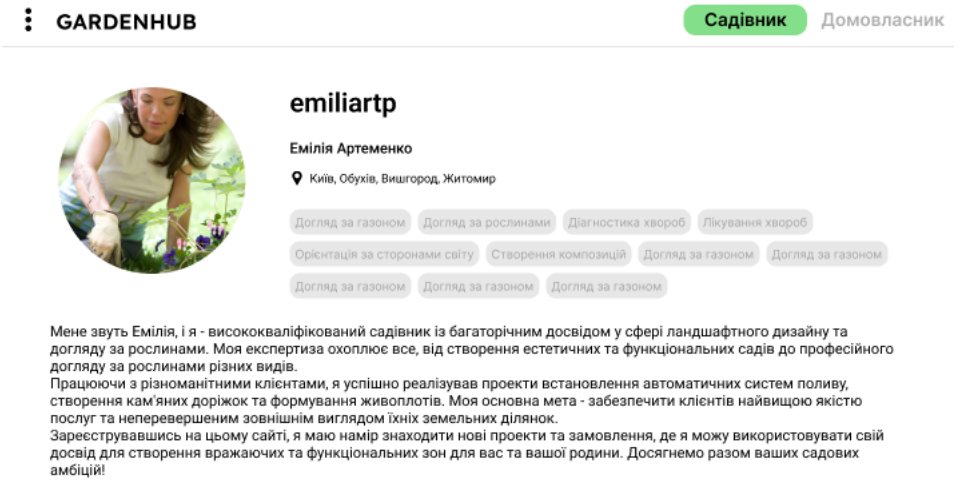


Рисунок 12 – Профіль садівника

ІМПЛЕМЕНТАЦІЯ ЗАСТОСУНКУ

Команда розробила оптимальну модель роботи завдяки комплексному підходу до розробки. Модель будується на припущенні, що кожний юзер може бути сутностями Gardener та Customer, або одночасно і тим, і тим. Очевидним рішенням було розбиття системи на фронтенд та бекенд частини.

Бекенд частина представлена REST API. Із особливостей є наявність BaseCRUDController, який виступає універсальною основою, що поєднує бізнес логіку та роботу з БД. Трьохрівнева структура витримана завдяки наявності EF моделей, репозиторіїв (обгортка над кожною таблицею для CRUD операцій з використанням EF), контролерів (носіїв бізнес-логіки) та фронтенд частини.

Для юзера початковою точкою вважається створення запису UserProfile. При потребі юзер може розширити свою інформацію: коли юзер хоче створити замовлення, він заповнює форму для CustomerProfile; коли юзер хоче виконати замовлення, він заповнює форму для GardenerProfile.

Всі ці сутності підв’язуються one-to-one в сутності UserProfile.

Для юзера в ролі садівника буде доступне створення вже виконаних проєктів, щоб була можливість підгрузити роботи, виконані поза сайтом. Саме портфоліо представляє собою проєкти, які юзер зробив (відображається в БД завдяки полю IsCompleted).

До кожного проєкту є можливість залишити відгук конкретному садівнику (фіксується завдяки unique constraint на поля ProjectId та GardenerProfileId). Ці відгуки сумуються і формують загальну оцінку користувача-садівника.

Також на сайті присутній пошук, який дозволяє швидко орієнтуватися в проєктах за їх назвами/описами.

Впровадження SignalR у нашому проєкті відкрило широкі можливості для покращення якості комунікації між клієнтською та серверною частинами додатка. SignalR - це потужна технологія, що дозволяє створювати реальні чи чатові застосунки з реактивним оновленням стану, яка забезпечує миттєву передачу повідомлень між клієнтами та сервером.

Крім того, SignalR вбудовує в себе механізми автоматичного перепідключення, що забезпечує стабільність з'єднання навіть при тимчасових втратах мережевого зв'язку або зміні умов. Це робить нашу систему більш надійною та готовою до роботи в різних умовах експлуатації.

Завдяки SignalR, наші користувачі отримують миттєві сповіщення про події, які їх цікавлять, незалежно від того, чи це нові повідомлення в чаті або інші важливі події. Це дозволяє їм залишатися завжди в курсі та реагувати на події в реальному часі.

Крім того, за допомогою SignalR ми створили потужну чатову систему, яка забезпечує швидку та плавну комунікацію між користувачами. Відправлення та отримання повідомлень в реальному часі дозволяє нашим користувачам спілкуватися без зайвих затримок та недоліків, підвищуючи ефективність та зручність взаємодії.

Загалом, впровадження SignalR стало важливим кроком у розвитку нашого додатка, забезпечивши йому потужність, ефективність та надійність, що відчутно поліпшило взаємодію з користувачами та рівень задоволення від використання.

ДОКУМЕНТАЦІЯ ПРОЄКТУ

Всі важливі матеріали та документи були організовані за допомогою Trello – універсального інструменту для управління проєктами. Це дозволило нам динамічно реагувати на зміни у вимогах та швидко адаптуватися до розвитку проєкту.

Інструменти. Дошка Trello використовувалась як централізована точка входу для усіх учасників проєкту. Тут зберігалася вся важлива інформація – технічне завдання, документація, дошка Figma із дизайном, аналіз конкурентів та мапа продукту (рис. 13).

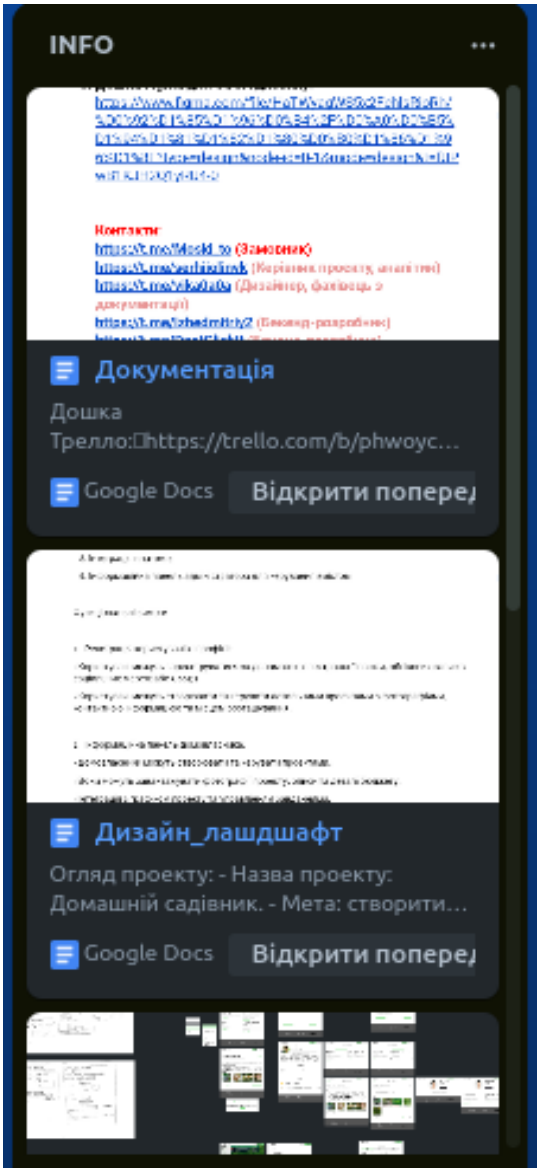


Рисунок 13 – Єдина точка входу на дошці Trello

Види створених документів:

- технічне завдання: опис вимог та функціональності продукту
- документація з інструкціями: надання детальних вказівок та рекомендацій щодо роботи з продуктом
- дошка Figma: вміщує в себе дизайн-макети та прототипи для візуалізації інтерфейсу
- аналіз конкурентів та мапа продукту: важливі матеріали для стратегічного планування та визначення концепції продукту.

Ця організація дозволяла всім учасникам команди легко отримати доступ до необхідної інформації, покращуючи комунікацію та ефективність спільної роботи.

ОПИС ПРОЦЕСУ РОЗРОБКИ

Таблиця 9. Розподіл ролей у команді

Студент	**Роль у команді**
Олійник Серій	керівник проєкту, аналітик, модератор
Дволінська Вікторія	UI/UX дизайнер, фахівець з документації
Ткаченко Ярослав	бекенд-розробник, тестувальник
Прохорчук Дмитро	бекенд-розробник
Таран Владислав	фронтенд-розробник
Сірікова Ольга	фронтенд-розробник
Аксані Іван	бекенд-розробник
Козакевич Роман	фронтенд-розробник
Росновський Ярослав	фронтенд-розробник

Методологія та система управління проєктами. Для організації нашої роботи було вирішено обрати методологію Scrum. Кожен спринт тривав один тиждень, якому передував зідзвон, де обговорювалось, що було зроблено під час минулого спринта, що планується зробити на наступному спринті та які проблеми виникали під час роботи, чи вдалось їх вирішити. Для спілкування всередині команди було створено Telegram-чат, де також і відбувались щотижневі зідзвони.

Необхідною частиною Scrum-методології є канбан-дошка. Її було вирішено створити у додатку Trello (рис. 14), тому що він здався нам найбільш зручним для використання у невеликих проєктах, має зрозумілий інтерфейс та можливості для налаштування під потреби команди. Дошку було поділено на наступні блоки:

- Product Backlog – беклог продукту
- Sprint N Backlog – список задач на N-ий спринт з присвоєними виконавцями
- Sprint N in process – задачі N-го спринта, що знаходяться на етапі виконання
- Sprint N done – виконані задачі N-го спринта
- Next ideas – ідеї для задач наступного спринта
- Info - єдина точка входу

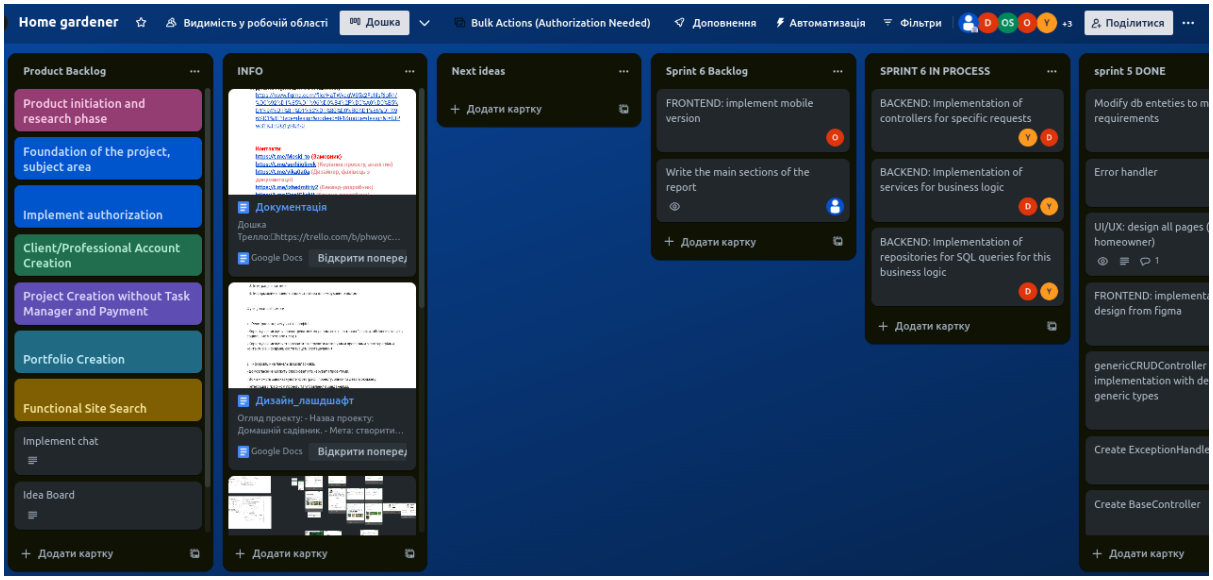


Рисунок 14 – Trello-дошка проєкту

Інструментарій. Під час розробки застосунку використані були наступні технології.

Для організації та планування роботи:

- Google Docs&Sheets – зручний інструмент для ведення та зберігання документації, дозволяє декільком користувачам одночасно працювати над документом.
- Trello – інструмент для планування та трекінгу виконання завдань, канбан-дошка нашого проєкту
- Figma – вебплатформа для дизайну та прототипування, яка дозволяє командам спільно працювати над проєктами, в режимі реального часу
- Git – інструмент спільної роботи над кодом
- Postman – API-платформа, на якій розробники можуть розробляти, створювати, тестувати та повторювати свої API
- Echometer – застосунок для проведення рефлексії
- Miro – цифрова платформа для співпраці, призначена для полегшення віддаленого та розподіленого командного спілкування та управління проєктами
- Swagger UI – інтерактивний інтерфейс, який дозволяє візуалізувати та тестувати RESTful API
- Microsoft Visual Studio – інтегроване середовище розробки (IDE) від Microsoft, яке надає розробникам інструменти для створення, тестування та відлагодження програмного забезпечення для різних платформ
- Visual Studio Code — редактор вихідного коду
- Microsoft SQL Management Studio – система управління базою даних

Технології розробки: .Net, Microsoft SQLServer, TypeScript (Angular), Html, CSS.

Планування спринтів. Щочетверга команда збиралась ввечері для обговорення всіх питань, які наступні задачі кожного з члена команди, які труднощі наразі є, які завдання встигаємо зробили, які ні, також відразу приділяли увагу на те, що рекомендували зробити викладачі, де і які проблеми в нас наразі існували – щоб якомога швидше все виправити (рис. 15).

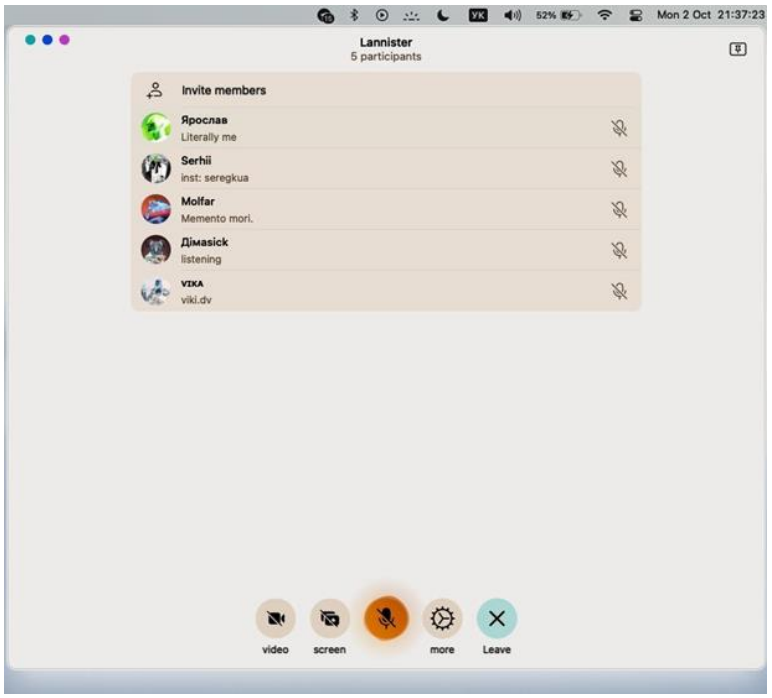


Рисунок 15 – Приклад зустрічей

Рефлексія. Наприкінці цього семестру наша команда провела важливу рефлексію по проєкту, де розглядались позитивні та проблематичні моменти в роботі, а також плани на вдосконалення у наступному семестрі. Ми використовували інтерактивну дошку, де кожен учасник міг висловити свої думки (рис. 16). Застосовуючи метод стікерів, кожен із нас виділяв позитивні моменти та ті аспекти, які потребують уваги. Цей процес дозволив нам зрозуміти, що цінує кожен член команди, виявити спільні проблеми та визначити напрямки покращень. Ми визначили конкретні кроки для усунення помилок та розв'язання проблем, а також визначили цілі на наступний семестр. Рефлексія стала важливим інструментом для нашого колективного розвитку, дозволяючи нам ефективно працювати над покращеннями та подальшим розвитком команди.

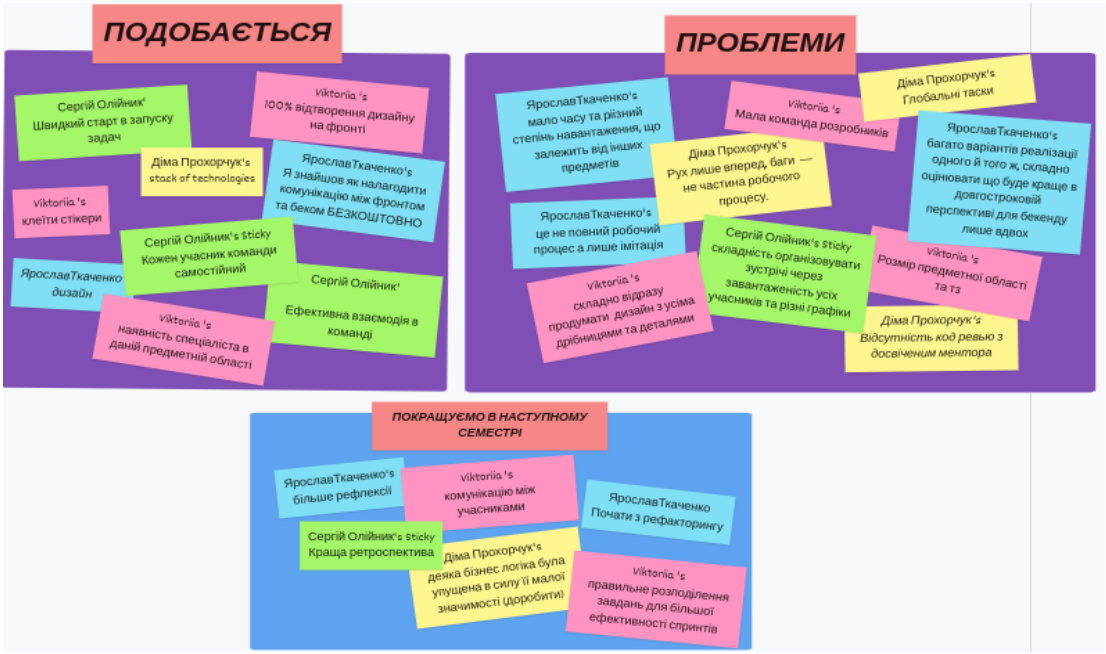


Рисунок 16 – Приклад рефлексії

ВИСНОВКИ

Проєкт "Gardenhub" був ініційований як MVP у першому семестрі, але протягом цього семестру ми значно розширили його функціонал. Зокрема, було додано чати, систему сповіщень та багато інших покращень. Цей процес дозволив нам значно підвищити функціональність та зручність використання платформи для власників будинків та садівників.

Під час цього семестру ми також провели детальний аналіз ринку та визначили стратегічні напрямки розвитку платформи. Робота над проєктом дозволила нам поглибити знання у формальних методах специфікації програм, перевірці якості продукту та використанні технологій .Net, Angular та UI/UX дизайну.

Методологія Scrum дала нам цінний досвід управління проєктом та спільної роботи в команді. Кожен учасник проєкту мав можливість розвиватися в обраній ролі та отримати цінний досвід, який стане важливим активом для подальшої кар'єри в сфері IT.

На заключний етап роботи ми отримали значно розширену версію застосунку, яку можна представити кінцевому споживачу.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Trello Software [Електронний ресурс] – Режим доступу до ресурсу: https://www.atlassian.com/ru/software/trello.
2. Trello [Електронний ресурс] – Режим доступу до ресурсу: https://trello.com/
3. Google Sheets [Електронний ресурс]. – Режим доступу до ресурсу: https://www.google.com/intl/uk_ua/sheets/about/
4. Figma [Електронний ресурс]. – Режим доступу до ресурсу: https://www.figma.com/
5. Echometer [Електронний ресурс]. – Режим доступу до ресурсу: https://echometerapp.com/uk/
6. СКРАМ. ЩО ЦЕ ТАКЕ ТА ЯК ЦИМ КОРИСТУВАТИСЯ [Електронний ресурс]. – Режим доступу до ресурсу:
https://brander.ua/blog/skram-shcho-tse-take-ta-yak-tsym-korystuvatysya
7. Mapus.ua [Електронний ресурс] – Режим доступу до ресурсу: https://mapus.ua/
8. Kabanchik.ua [Електронний ресурс] – Режим доступу до ресурсу: https://kabanchik.ua/
9. Rabotniki.ua [Електронний ресурс] – Режим доступу до ресурсу: https://www.rabotniki.ua/uk/
10. Gigsmart.com [Електронний ресурс] – Режим доступу до ресурсу: https://gigsmart.com/

ЗАСТОСУНОК "MAP OF ACTIVITIES"

Олексій Мацуй, Микола Васильчук, Сергій Лапюк, Олексій Астраханцев, Юрій Підлетейчук, Надія Огійчук

Дана робота присвячена розробці та впровадженню кросплатформного застосунку "Map of Activities ", спрямованого на об'єднання різних типів заходів та подій у реальному часі. Проєкт має на меті створення зручного сервісу, який дозволить користувачам брати активну участь у соціальному житті міста, організовуючи чи беручи участь у заходах. В рамках проєкту було розроблено кросплатформний застосунок, проведено аналіз існуючих рішень, вибрано засоби розробки, прийнято архітектурні рішення та проведено тестування.. Застосунок "Map of Activities " дозволяє користувачам переглядати актуальні події, створювати власні заходи та брати участь у соціальному житті міста, що сприяє підвищенню соціальної активності та покращенню комунікації серед мешканців.

This project focuses on the development and implementation of the cross-platform application "Map of Activities ", aimed at uniting various types of events and activities in real-time into one application. The project aims to create a user-friendly service that fosters active participation in the city's social life by enabling users to organize or join events. Within the project, a cross-platform application was developed, an analysis of existing solutions was conducted, development tools were selected, architectural decisions were made, and testing was carried out. "Map of Activities " allows users to view actual events, create their activities, and participate in the social life of the city, which contributes to increased social activity and improved communication.

КАРТА ПОДІЙ, ОРГАНІЗАЦІЯ ПОДІЙ, РЕАЛЬНИЙ ЧАС, СОЦІАЛЬНЕ ЖИТТЯ МІСТА, ВЕБЗАСТОСУНОК, ASP.NET CORE WEB API, ENTITY FRAMEWORK CORE, MICROSOFT VISUAL STUDIO 2022, MICROSOFT VISUAL STUDIO CODE, C#, QUASAR FRAMEWORK.

MAP OF ACTIVITIES, ORGANISATION OF ACTIVITIES, REAL TIME, SOCIAL LIFE OF THE CITY, WEB APPLICATION, ASP.NET CORE WEB API, ENTITY FRAMEWORK CORE, MICROSOFT VISUAL STUDIO 2022, MICROSOFT VISUAL STUDIO CODE, C#, QUASAR FRAMEWORK.

ВСТУП

Оцінка сучасного стану об’єкта розробки. Мережа Інтернет переповнена різними застосунками, здатними виконувати багато задач. Причому, в умовах масовості та інтенсивності подій, які мають різноплановий характер інформація про їх проведення часто упорядкована хаотично, або є застарілою. Існує безліч сайтів, які висвітлюють лише певні події. ці сайти або застосунки переважно є монофункціональними та розвиваються в одному векторі, висвітлюючи окремий тип подій. Застарілість цих подій унеможливлює участі мешканців у актуальних подіях, а з іншого боку створює перешкоди для організаторів у поширенні подій. Проблемою користувачі є відсутність єдиного джерела, який акумулював би всю інформацію про проведені події в реальному часі з можливістю приєднання до подій. Проблеми користувачів також стосуються нестачі живої комунікації та нових знайомств.

Актуальність проєктної роботи й підстави для її виконання. Обумовлена потребою у швидкому та легкому інструменті, який би об’єднав різні типи заходів та подій у реальному часі, і відповідав би принципам: швидкості та зручності - у сучасному світі, де час – це цінний ресурс, користувачі цінують швидкість та зручність у пошуку

подій. відповідність до реального часу - оперативна подача інформації про події та заходи, стає важливим елементом для актуальності та вчасного реагування. одночасне об’єднання різних типів подій - що дозволить користувачам знаходити та брати участь у заходах за їхніми індивідуальними уподобаннями від спортивних заходів і концертів до культурних подій та подій міського активізму. географічній доступності - швидкий пошук подій та заходів у конкретному місті, стає зручним для мешканців

Мета й завдання проєктної роботи. Головною метою проєкту - є створення зручного сервісу, що дозволяє брати активну участь у соціальному житті міста, організовуючи чи беручи участь у заходах. Для досягнення цієї цілі необхідно розв’язати наступні завдання:

1. Розробити методологію та систему управління проєктом (визначити ролі усіх членів команди).
2. Зробити аналіз існуючих аналогів, зазначивши ряд недоліків та переваг.
3. Провести огляд існуючих рішень, що надають можливості створювати події в реальному часі з можливістю приєднання користувачів до них.
4. Вибрати засоби розробки сервісу, як то: мову програмування, інструменти що полегшать розробку.
5. Прийняти архітектурні рішення щодо структури проєкту, визначити інтерфейс взаємодії користувача із застосунком.
6. Провести аналітику щодо подій міського активізму, які відбуваються наразі.
7. Розробити кросплатформний додаток, додавши аналітику "міського активізму".
8. Провести тестування кросплатформного додатку.

Об’єкт, методи й засоби розроблення. Об'єктом проєктної роботи є розробка та впровадження застосунку "Map of activities ". Цей додаток призначений для об'єднання різних типів заходів та подій у реальному часі, дозволяючи користувачам знаходити та брати участь у різноманітних подіях в їхньому місті. Крім того, застосунок надає можливість користувачам створювати власні події та брати участь у соціальному житті міста.

Предметом проєктної роботи є застосунок "Map of activities ". Розроблений застосунок дозволяє користувачам переглядати та ознайомлювати із актуальними подіями, наприклад, в рідному місті. Більше того, користувачам наданий зручний та інтуїтивно-зрозумілий функціонал для створення власних подій. Методи розроблення включають аналіз існуючих застосунків, проєктування застосунку, розробку логіки та створення інтерфейсу.

Для розробки використовуються інструменти, такі як Microsoft Visual Studio 2022 [1], BitBucket [2], мова програмування C#, фреймворк Quasar [3].

Можливі сфери застосування. Застосунок може бути використаний у вирішенні одночасно декількох аспектів: покращенню соціальної взаємодії - додаток дозволяє людям, які мають спільні інтереси чи хобі, знаходити одне одного та об'єднуватися навколо конкретних тем, заходів та ініціатив. стимулюванні соціальної активності - користування додатком підтримує соціальну активність та зустрічі людей. Спрощенні інформації про проведення подій. Додаток також дозволяє якісно, по новому впливати на міські трансформації, організовуючи та висвітлюючи події міського активізму. Актуальність даного проєкту також обумовлена великою кількістю різноманітних інтересів, які представлені, зручністю вибору події за вподобаннями, можливістю створювати власні події та організовувати заходи, відсутність подібної локальної програми в Україні.

ПОШУК ТА ВАЛІДАЦІЯ ІДЕЇ

З появою нових технологій відкривається безмежний простір для створення різноманітних вебзастосунків та мобільних застосунків. Кожен день на ринку з'являються нові та вдосконалюються наявні системні рішення, що робить вибір складнішим завданням. Вони допомагають покращити продуктивність та ефективність роботи багатьох організацій, що робить їх досить популярними. Огляд наявних систем на ринку є важливим етапом, оскільки він дозволяє виявити переваги та недоліки кожної з них.

Опис пошуку ідеї продукту.

Опис обраної сфери.

Сфера "Entertainment " є високодинамічним та технологічно розвиненим сегментом індустрії, спрямованим на створення та поширення розважального контенту для задоволення різноманітних потреб аудиторії. Основною метою проєктів у цій сфері є розробка та презентація унікального та захоплюючого контенту, спроможного викликати емоційні реакції та відповідати вимогам споживачів.

Опис обговорюваних ідей.

1. Entertainment – Мапа подій. Події біля тебе.

Застосунок дозволяє користувачам створювати геолокаційні точки, пов'язані з певними інтересами або подіями. Ці точки видні на карті у застосунку або в списку. Користувачі можуть планувати події заздалегідь, надаючиможливість іншим користувачам приєднатися або від’єднатись від неї свою зацікавленість. Система також надсилатиме сповіщення про найближчі події. Це дозволяє людям знайти та приєднатися до подій, які відповідають їхнім інтересам, сприяючи соціальній взаємодії та знайомствам.

2. Entertainment - Конструктор настільних ігор.

Застосунок створений для задання ігор, в які можна грати з друзями онлайн. Ігри генеруються на основі шаблонів, а користувачі можуть редагувати дизайн та логіку гри. Крім того, конструктор можна використовувати для друкування ігор і подальшої гри в паперову версію. Це надає можливість гравцям налаштовувати свої ігри, роблячи геймплей більш цікавим і унікальним.

3. EdTech - Помічник абітурієнта.

Застосунок створений для допомоги студентам та абітурієнтам у виборі найкращого варіанту для навчання. Система аналізує університети за інтересами та балом користувача, надаючи рекомендації та можливість консультування із представниками університету. Це допомагає абітурієнтам знаходити оптимальний навчальний шлях відповідно до їхніх прагнень та інтересів.

4. Volunteering - Актуальні волонтерські збори.

Ця платформа створена для зручного волонтерства, де користувачі можуть легко вибрати проєкти для підтримки та отримати підробний звіт про витрати коштів. Застосунок також надає можливість організувати свої власні волонтерські збори. Це сприяє залученню волонтерів та допомагає організаторам зборів легко керувати та відстежувати свої проєкти. Розширення платформи може включати можливість проведення онлайнкурсів або спільних подій для волонтерів, роблячи застосунок більш відкритим та важливим для спільноти.

Огляд наявних на ринку ІТ-продуктів – конкурентів

З появою нових технологій відкривається безмежний простір для створення різноманітних вебзастосунків. Кожен день на ринку з'являються нові та

вдосконалюються наявні системні рішення, що робить вибір вебзастосунків складнішим завданням. Огляд наявних систем на ринку є важливим етапом в процесі вибору вебзастосунків, оскільки він дозволяє виявити переваги та недоліки кожної з них.

Meetup [4] позиціонує себе як платформу для знаходження та приєднання до місцевих груп та подій, відповідно до інтересів користувача. Її ключові функції включають пошук та приєднання до груп, планування подій для групи, взаємодію з іншими учасниками та отримання сповіщень про майбутні заходи. Цільова аудиторія - люди, які бажають знаходити події, що відповідають їхнім інтересам, з акцентом на місцевість. Продукт доступний для користувачів безкоштовно, але організаторам подій пропонуються платні пакети.

Allevents.in [5], натомість, пропонує широкий вибір подій різних категорій, таких як концерти, конференції та фестивалі. Вона дозволяє користувачам шукати, планувати та реєструватися на події через онлайн-платформу. Маючи глобальне охоплення, allevents.in підходить для широкого кола аудиторії. Як і в Meetup, доступ до основних функцій є безкоштовним, але організатори подій можуть використовувати платні пакети для розміщення та просування своїх заходів.

Мапа продукту.

Створення нової платформи розв'яже проблеми нестачі живої комунікації, нових знайомств, складнощів у організації подій та пошуку спільних інтересів для цільової аудиторії віком 12+. Наша платформа буде відрізнятися великим вибором різноманітних інтересів, зручністю вибору подій та можливістю створювати власні точки та організовувати заходи. Ми будемо просуватися серед студентських ком'юніті, використовуючи рекламу в соціальних мережах та спеціалізованих каналах у Тіктоку та тематичних пабліках в Телеграмі. Для отримання прибутку ми плануємо впровадження підписок на просування подій, рекламні банери та привілейовані події на карті. Основні витрати будуть пов'язані з аналізом даних користувачів, оплатою адміністраторів, рекламою та витратами на технічне забезпечення. Ключові метрики успіху включають кількість користувачів, створених точок подій, географічне охоплення та час, проведений у застосунку. Наша неринкова перевага полягає в тому, що в Україні відсутня подібна локальна програма, а також у можливості створення простих або неофіційних подій будь-яким користувачем.

Сильні і слабкі сторони розроблюваного продукту.

SWOT-аналіз обраної ідеї.

Сильні сторони ідеї:

- Пошук і спрощення знайомств за інтересами;
- Гарний спосіб залучити більше людей до події та шукати події, які можуть зацікавити;
- Створення різноманітного інтерактивного контенту та ігор (пошук скарбів).

Слабкі сторони:

- Створення фальшивих точок, зон інтересів недобросовісними користувачами;
- Потрібно залучити модераторів, що перешкоджатимуть створенню фальшивих точок та спаму.

Ризики:

- При малій кількості користувачів, користувачі матимуть малу вибірку подій поруч, що знизить кількість користувачів.

Можливості:

- Популярність застосунку підтримуватиметься бажанням людей знайти місце для відпочинку та знайомства;

- Можливе партнерство з ресторанами, концертними залами, магазинами, що може включати платну рекламу.

ОПИС ЗАСТОСУНКУ

Призначення застосунку.

Застосунок "Map of Activities " розроблений з метою надання ефективного інструменту для організації та участі в різноманітних подіях, що відображаються на мапі та у відповідному списку подій. Основне призначення застосунку полягає в можливості створення, відстеженні та долученні до актуальних заходів різного характеру, таких як коворкінги, турніри із шахів, активізм тощо. Він призначений для широкого кола користувачів, які цікавляться участю в подіях та спільнотною взаємодією. Користувачі можуть планувати події заздалегідь, дозволяючи іншим користувачамприєднатися або відзначити свою зацікавленість.

Метою створення застосунку є забезпечення зручної та інтуїтивно зрозумілої платформи для створення та участі в подіях різного масштабу. Він призначений для вирішення проблеми обмеженої доступності інформації про події та створення зручного середовища для організації події.

Цілі створення застосунку. "Map of Activities " створюється з метою:

- Забезпечити користувачам можливість легко знаходити та долучатися до подій, які їх цікавлять;
- Сприяти залученню людей до участі в різноманітних заходах та спільнотах;
- Забезпечити зручну та інтуїтивно зрозумілу платформу для створення та участі в подіях різного масштабу.

Також застосунок має мати такі функції, зображені на рис. 1 [6].

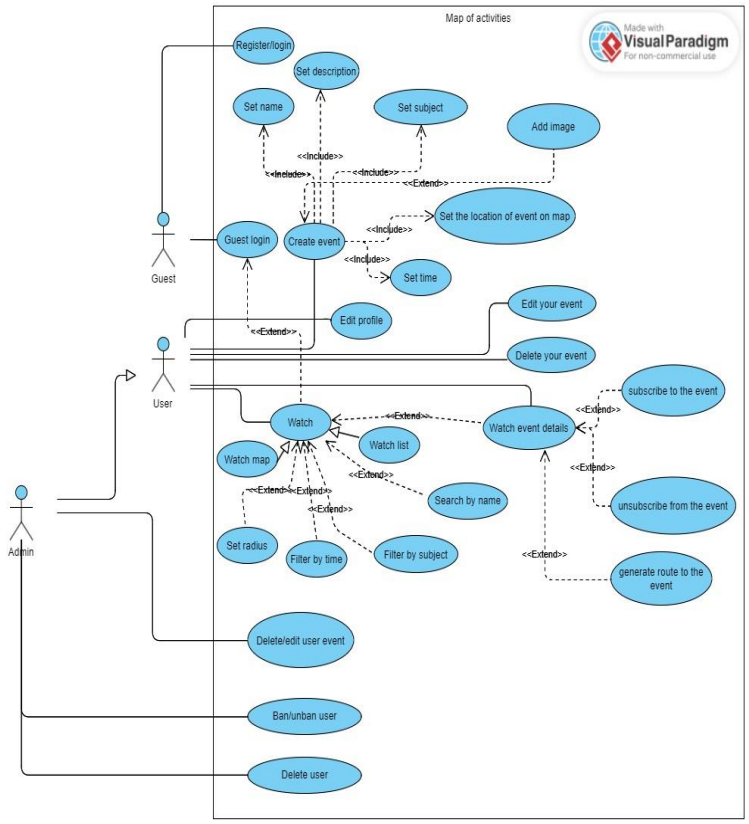


Рисунок 1 – Use-case діаграма застосунку "Map of Activities "

Вимоги до застосунку.

Функціональні вимоги:

- Застосунок повинен забезпечувати можливість створення, редагування та видалення подій;
- Застосунок повинен забезпечувати можливість пошуку подій за типом, часом, місцем проведення та іншими параметрами;
- Застосунок повинен відображати актуальні події у списку та на карті;
- Застосунок повинен забезпечувати можливість приєднання користувачів до подій;
- Застосунок повинен показувати маршрут до події.

Нефункціональні вимоги:

- Застосунок повинен мати простий і інтуїтивно зрозумілий інтерфейс;
- Застосунок повинен бути адаптивним до різних пристроїв;
- Застосунок повинен бути доступним "24/7 ";
- Застосунок повинен бути масштабованим;
- Застосунок повинен бути захищеним від несанкціонованого доступу.

Програмні вимоги:

- Веббраузер, наприклад, Chrome або Firefox;

Крім того, для розробки та розгортання системи можуть знадобитися такі програмні інструменти:

- Віддалена система управління базами даних, наприклад, Neon;
- Вебсервер, наприклад, Apache або Nginx;
- Системи контролю версій, наприклад, BitBucket.

Додаткові вимоги:

- Підключення до Інтернету;
- Доступ до геолокації;
- Доступ до камери.

Вимоги до структури та функціонування.

Застосунок "Map of activities " повинен забезпечувати можливість користувачам знайти події, які їх цікавлять, та приєднатися до них. Також для користувачів має бути наданий зрозумілий функціонал для організації власної події.

В застосунку передбачається виділити наступні функціональні підсистеми:

1) Адміністративна підсистема призначена для управління застосунку, включаючи:

- Додавання, редагування та видалення подій;
- Налаштування застосунку;
- Моніторинг роботи системи.

2) Підсистема користувача призначена для взаємодії користувачів із застосунком, включаючи:

- Реєстрацію та авторизацію користувачів;
- Створення, редагування та видалення події;
- Перегляд на карті актуальних подій;
- Пошук подій;
- Приєднання до подій.

Вимоги до функцій, які виконуються представлено в таблицях 1-2.
Підсистема адміністратора.

Таблиця 1. Перелій функцій, задач що підлягають автоматизації

Функція	**Задача**
Керувати роботою користувачів застосунку	Блокування або розблокування користувачів
	Видалення облікових записів користувачів
	Редагування або видалення подій користувачів
	Опрацювання скарг користувачів на інших користувачів або події
	Перегляд логів дій адміністраторів

Підсистема користувача

Таблиця 2. Перелій функцій, задач що підлягають автоматизації

Функція	**Задача**
Гостьовий вхід	Надання доступу до основних функцій застосунку, окрім створення, редагування, видалення події, приєднання до події та від'єднання від події
Реєстрація	Введення інформації про себе
	Підтвердження пошти
Створення події	Створювання події, вказуючи такі поля: назва, тип, час початку та час кінця (за необхідності), координати, зображення (опціонально), опис (опціонально)
Редагування події	Редагування існуючої події в застосунку із можливістю змін будь-яких із полів, які були вказані при створенні події
Видалення події	Видалення існуючої або запланованої події із застосунку
Пошук події	Забезпечення можливості користувачам шукати події за такими параметрами: тип, час початку та кінця, місце проведення. Результати мають бути показані в списку та на мапі
Перегляд події	Забезпечення можливості користувачам переглядати події у списку або на карті
Перегляд деталей події	Перегляд всіх деталей про подію: автор, локація на карті, назва, тип, час початку та час кінця, зображення, опис
Приєднання до події	Приєднання до події користувачем
Від'єднання від події	Від'єднання від події користувачем
Побудова маршруту	Побудова маршруту до події
Скарга на користувача або подію	Скарга на користувача або подію

Системні вимоги.

Застосунок повинен коректно відображатися в інтернет-браузерах:

- Chrome 98 і вище;
- MS Internet Explorer 11 і вище;
- Mozilla Firefox 85 і вище.

Застосунок повинен працювати на наступних операційних системах:

- Windows 10 і вище;
- Android.

Джерелом даних для застосунку є віддалена база даних Neon. Для підключення до бази даних використовується протокол HTTP. Застосунок повинен забезпечувати безпеку даних користувачів. Для цього необхідно:

- захистити дані користувачів від несанкціонованого доступу, використовуючи HTTPS для шифрування даних, що передаються між користувачем і застосунком.

Додаткові вимоги:

- рекомендована швидкість Інтернет-з'єднання - 10 Мбіт/с.

Опис організації інформаційної бази.

База даних для застосунку "Map of Activities " реалізована у системі управлінням базами даних Neon. Діаграма бази даних зображена на рис. 2.

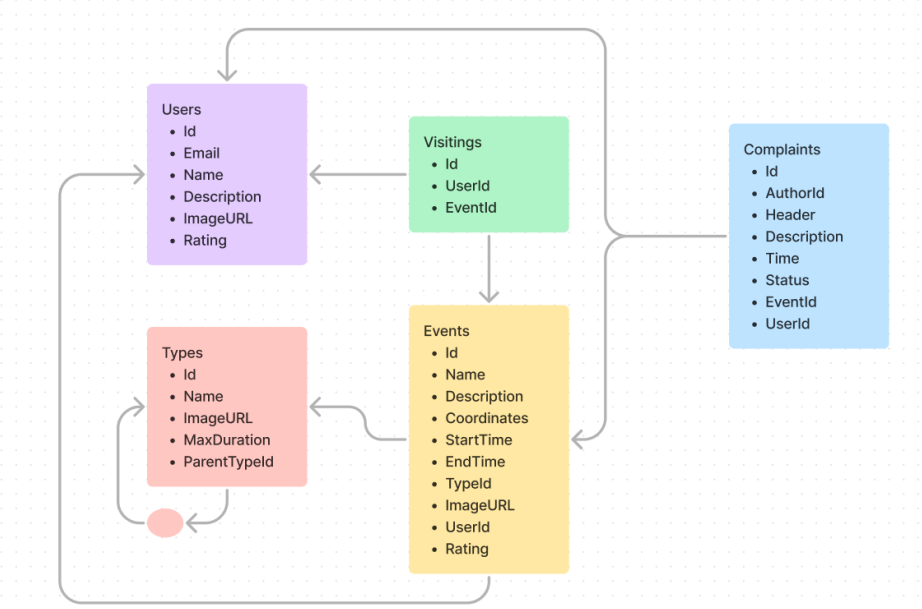


Рисунок **2** – Діаграма бази даних застосунку "Map of Activities "

Перелік таблиць бази даних наведений в таблиці 3.

Таблиця 3. Перелік таблиць бази даних застосунку

Номер	**Таблиця**	**Опис**
1	Events	Таблиця для збереження інформації про події
2	Types	Таблиця для збереження інформації про типи подій
3	Users	Таблиця для збереження інформації про користувачів
4	Visitings	Таблиця для збереження інформації про відвідуваність подій
5	Complaints	Таблиця для збереження інформації про скарги на користувачів та події

ІМПЛЕМЕНТАЦІЯ ЗАСТОСУНКУ

Бекенд імплементація застосунку "Map of activities".

Бекенд імплементація застосунку "Map of activities " - це процес створення та розгортання серверної частини застосунку. Вона включає в себе такі завдання, як:

- розробка бази даних;
- розробка моделей;
- розробка контролерів;
- розробка сервісів.

Створення моделей налаштування та бази даних.

Для створення бази даних в СКБД застосуємо підхід Code First. Спочатку необхідно створити класи моделей відповідно до спроєктованої бази даних.

Перелік класів моделей наведений в таблиці 4.

Таблиця 4. Перелік класів моделей

Номер	Назва	Опис
1	ApplicationIdentityDbContext	Представляє контекст бази даних для сутностей, пов'язаних з ідентифікацією. Він відповідає за керування з'єднанням з базою даних та операціями, пов'язаними з ідентичністю, такими як автентифікація та авторизація користувачів
2	ApplicationUser	Представляє сутність користувача у програмі. Він розширює клас IdentityUser, наданий ASP.NET Core Identity ApplicationUserRoles.cs.
3	ApplicationUserRoles	Зберігає ролі, які можна призначити користувачам
4	Event	Описує сутність "Подія "
5	EventView	Модель для представлення даних події, переданих користувачем
6	ForgotPasswordModel	Модель для забутого пароля
7	LoginModel	Представляє модель для обробки облікових даних користувача. Використовується під час процесу автентифікації
8	MapOfActivitiesAPIContext	Представляє контекст бази даних для програми. Включає властивості DbSet для різних сутностей, що визначають, як вони відображаються у базі даних
9	RefreshTokenModel	Представляє модель для обробки токенів оновлення у системі автентифікації на основі токенів
10	RegisterModel	Представляє модель для реєстрації користувачів, включаючи необхідну інформацію для створення нового облікового запису користувача
11	ResetPasswordModel	Модель для забутого пароля

Номер	Назва	Опис
12	ResponseModel	Являє собою загальну модель для відповідей API. Ймовірно, включає таку інформацію, як статус успіху, повідомлення про помилки або дані
13	TokenModel	Представляє модель для роботи з токенами JWT, включаючи такі властивості, як токени доступу та час закінчення терміну дії
14	Type	Описує сутність "Тип події "
15	User	Описує сутність "Користувач "
16	Visitings	Описує сутність "Відвідування "
17	Complaints	Описує сутність "Скарги "

Для прикладу, код моделі "Користувач " зображений на рис. 3.

```
namespace MapOfActivitiesAPI.Models
{
    public class User
    {
        public int Id { get; set; }
        public string UserId { get; set; }
        public string Email { get; set; }
        public string Name { get; set; }
        public string? Description { get; set; }
        public string? ImageURL { get; set; }

        public List<Event> CreatedEvents { get; set; }
    }
}
```

Рисунок 3 – Код класу "Користувач "

Налаштування контролерів.

Тепер потрібно створити контролери для роботи з об’єктами, нижче в таблиці 5 наведені назви контролерів та їх призначення.

Таблиця 5. Перелік контролерів

Назва	Призначення
AccountController	Для роботи із авторизацією користувачів
EventsController	Для роботи із подіями
ImagesController	Для роботи із картинками
TokensController	Для роботи із токенами для авторизації користувачів
TypesController	Для роботи із типами подій
UsersController	Для роботи із користучами
VisitingsController	Для роботи із відвідуваннями користувачів
ComplaintsController	Для роботи із скаргами на користувачів та події
WebLogsController	Для роботи із логуванням дій адміністратора

Фронтенд імплементація застосунку "Map of activities".

Фронтенд імплементація застосунку "Map of activities " реалізована за допомогою фреймворку Quasar. Quasar - це сучасний фреймворк для веброзробки, який забезпечує швидку та просту розробку сучасних вебзастосунків. Фронтенд імплементація

застосунку "Map of activities " - це процес створення та розгортання клієнтської частини застосунку. Вона включає в себе такі завдання, як:

- створення проєкту Quasar;
- створення сторінок та компонентів;
- відправка запитів до бекенду за допомогою Axios;
- створення єдиного дерева стану.

Імплементація застосунку "Map of activities " під платформу Android

Для імплементації застосунку "Map of activities " під платформу Android необхідно виконати наступні кроки:

1. Створити новий проєкт Quasar CLI.
2. Додати режим Capacitor до проєкту за допомогою команди `npx quasar mode add capacitor`.
3. Згенерувати код застосунку для платформи Android за допомогою команди `npx quasar build -m capacitor -T android -ide`.
4. Імплементувати функції застосунку.

ТЕСТУВАННЯ

Підходи до модульного тестування поділяють на монолітний та покроковий:

Монолітний підхід – спочатку окремо тестуються всі модулі, потім вони об’єднуються в робочу програму для комплексного тестування.

Покроковий підхід – кожен модуль під’єднується до набору ужевідтестованих модулів. Покрокове тестування поділяють на методи:

- Висхідний (тестується лише низові модулі);
- Низхідний (тестуються лише головний модуль);
- Модифікований низхідний (кожен модуль проходить автономне тестування перед підключенням до програми);
- Метод великого стрибка (Кожен модуль тестується автономно. Потім всі модулі інтегруються всі одразу);
- Метод сандвіча (Одночасне низхідне і висхідне тестування);
- Модифікований метод сандвіча (Нижні рівні тестуються строго знизу вгору. Модулі верхніх рівнів спочатку тестуються ізольовано, а потім збираються низхідним методом).

Для вибору коректного для проєкту підходу ми провели їх оцінку за наступними критеріями, що наведені у таблиці 6.

Таблиця 6. Перелік вагових коефіцієнтів

Номер	Ваговий коефіцієнт	Критерій
1	3	Рання збірка
2	3	Раннє отримання працюючого каркасу
3	2	Складність підготовки заглушок
4	2	Складність планування та управління послідовністю тестів
5	1	Можливість тестувати окремі шляхи
6	1	Паралелізм
7	1	доступність інструментів тестування

Оцінимо кожен метод, що наведені у таблиці 7.

Таблиця 7. Оцінка методів

Вага	Критерій	Висхід ний	Низх	Мод низх	Великого стрибка	Сандвіча	Мод сандвіча
3	1	+	+	+	-	+	+
3	2	-	+	+	-	+	+
2	3	-	+	+	+		+
2	4	+	-	-	+	-	-
1	5	+	-	-	+		+
1	6		-		+		+
1	7	+	-	+	+		+
	Всього	2	3	6	-3	6	9

Найбільшу оцінку отримав модифікований метод сандвіча. За даним методом нижні рівні тестуються знизу вгору, а модулі верхніх рівнів тестуються у 2 етапи: спочатку ізольовано, а потім збираються і тестуються низхідним методом.

Модульні тести.

До модулів нижчих рівнів відносяться складові бекенду, а саме методи контроллерів та допоміжних сервісів.

До модулів вищих рівнів відносяться складові фронтенду, а саме модуль навігації та сторінок, компоненти застосунку (поле вводу часу, спадний список для обрання типів та інші), модуль для надсилання API-запитів.

Для тестування нижчих модульних рівнів використовувався swagger та postman.

На рис. 4 наведено приклад 10-ти модульних тестів.

Метод отримання дистанції між точками	passed
Метод фільтрації подій	passed
Операції з подіями	passed
Отримання збережених зображень	passed
Компонента вибору типу	passed
Компонента фільтрації типів	passed
Знаходження геолокації на мапі	failed
Виставлення точки на мапі	passed
Валідація створення події	passed
Прокладання маршруту на мапі	passed
Вхід у систему	passed
Реєстрація в системі	passed
Оновлення токену	passed

Рисунок 4 – Модульні тести

Тести збірки.

Точки зустрічей модулів верхнього та нижнього рівнів відбувались регулярно під час кожного зі спринтів.

Найглобальніша точка зустрічі, у якій поєднувались модулі верхнього та нижнього рівнів, відбулась під час реалізації входу в систему на клієнтському та серверному боці.

На рис. 5 наведено приклад 5-ти тестів збірки

Додавання зображень до події	passed
Додавання зображень до профілю	passed
Вхід через браузер	passed
Оновлення токену в браузері	passed
Автоматичний вхід за збереженими токенами	passed

Рисунок 5 – Тести збірки

ДОКУМЕНТАЦІЯ ПРОЄКТУ

Вимоги до функціоналу.

Документація була написана за допомогою сервісу Confluence. Вимоги до функціоналу:

1. Створення, редагування, видалення подій - зареєстровані користувачі
2. Перегляд подій - всі
3. Долучитись до події - зареєстровані користувачі
4. Залишити коментар до події чи оцінити подію - зареєстровані користувачі
5. Типи (категорії) подій редагуються модератором. При створенні події обирається її тип, який не може мати типів-нащадків.
6. Фільтрація подій відбувається за пошуком по назві та описом, типом (будь-якого рівня), відстанню, часом, тривалістю

На мапі повинна бути генерація маршруту.

Концептуальна модель.

Ролі.

Гість – може спробувати весь важливий функціонал програми: перегляд списку подій, застосовування фільтрів для подій в списку або на мапі, переглядати мапу із наявними подіями. Не може: створювати та приєднуватись до подій, коментувати подію або оцінювати організатора події.

Авторизований користувач - має всі можливості гостя. Тепер може створювати події та видаляти свої події, приєднуватись до існуючих подій, писати коментарі до події.

Адміністратор – має всі можливості користувача, а також може редагувати події інших користувачів із метою обмеження або недопущення поширення ненормативної лексики, блокувати та розблоковувати користувачів, опрацьовувати скарги на події та користувачів.

Інструкція для неавторизованого користувача

Неавторизованого користувача зустріне сторінка, зображена на рис. 6. У користувача є три опції: зареєструватись, увійти в існуючий обліковий запис та увійти як гість.

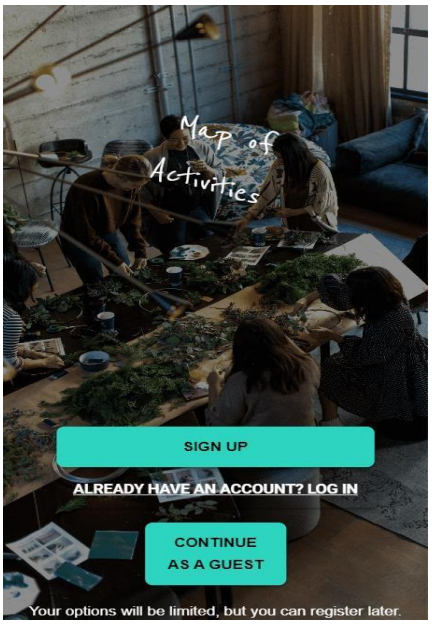


Рисунок 6 – Початкова сторінка

Якщо користувач вирішив зареєструватися, то на сторінці реєстрації (рис. 7), йому необхідно вказати ім’я, електронну пошту, пароль та повторити пароль.

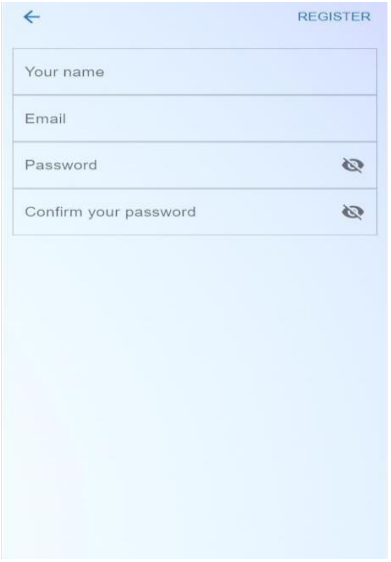


Рисунок 7 – Сторінка реєстрації

Після цього користувач отримає сповіщення (рис. 8) про те, що йому було надіслано лист на вказану електронну адресу для підтвердження реєстрації.

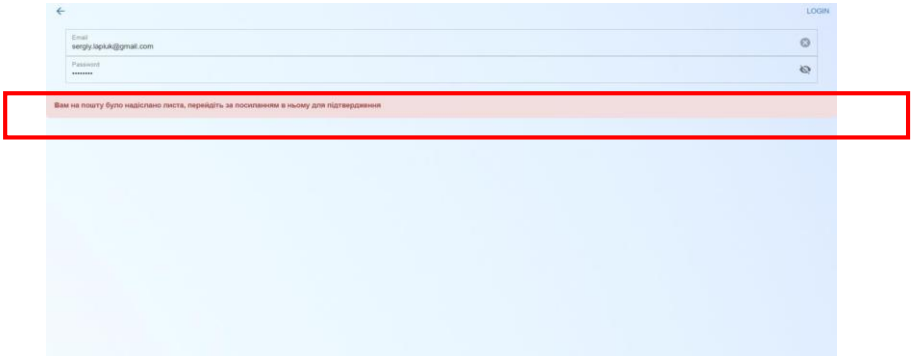


Рисунок 8 – Сповіщення про підтвердежння

Тепер необхідно відкрити отриманий лист (див. рис. 9) та підтвердити реєстрацію, тобто натиснути на кнопку "Підтвердити дію ".

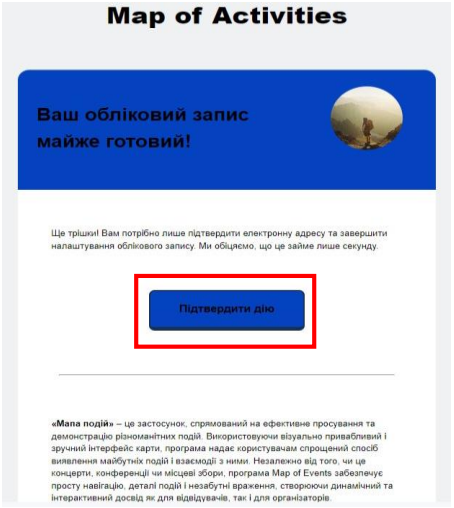


Рисунок 9 – Підтвердження на пошті

Коли користувач натисне на кнопку "Підтвердити дію ", він автоматично перейде на сторінку входу, де буде вказано, що підтвердження реєстрації пройшло успішно (див. рис. 10). Також користувач переходить на цю сторінку, якщо на початковій сторінці вирішить увійти в наявний аккаунт.

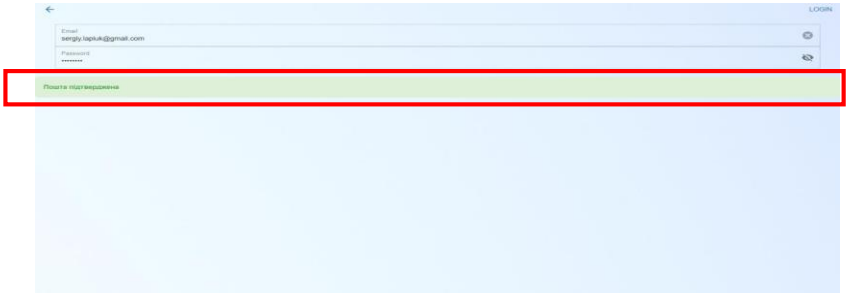


Рисунок 10 – Сторінка після підтвердження пошти

Після натиснення кнопки "LOGIN" користувач увійде у свій обліковий запис. Перевагами зареєстрованих користувачів є те, що вони можуть створювати події, підписуватись на події, також користувачі мають профіль, який вони можуть редагувати.

Після закінчення авторизації користувач потрапляє на сторінку зі списком подій (рис. 11), де є верхня панель для пошуку подій за назвою та нижня панель для навігації між сторінками, яка доступна на кожній сторінці.

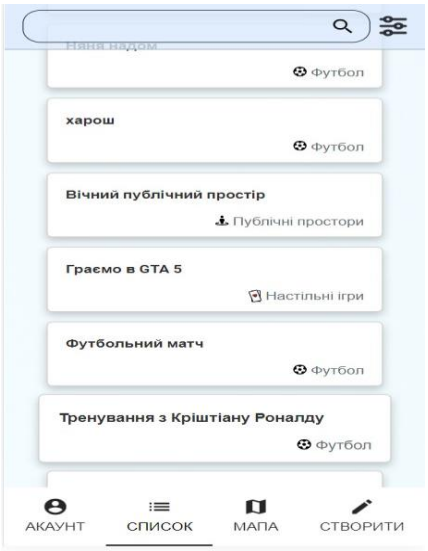


Рисунок 11 – Сторінка зі списком подій

Якщо потрібно відфільтрувати події за часом початку або кінця, за типом події чи відстанню, то можна натиснути на кнопку фільтрації, що у верхній панелі сторінки зі списком подій. Тоді відкриється бічна панель (рис. 12), де можна буде встановити параметри для фільтрації.

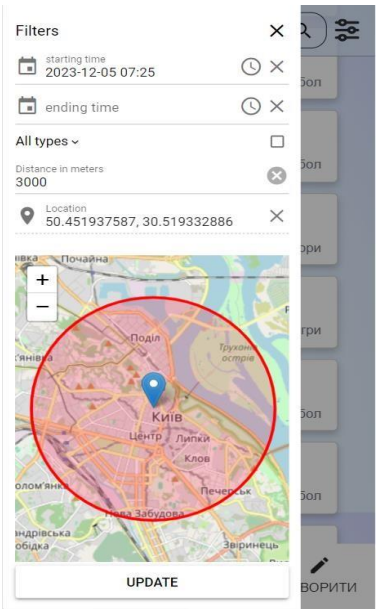


Рисунок 12 – Фільтрація для подій

Також є можливість переглянути всі події на мапі (див. рис. 13). На мапі також є можливість пошуку подій за назвою та фільтрації подій.

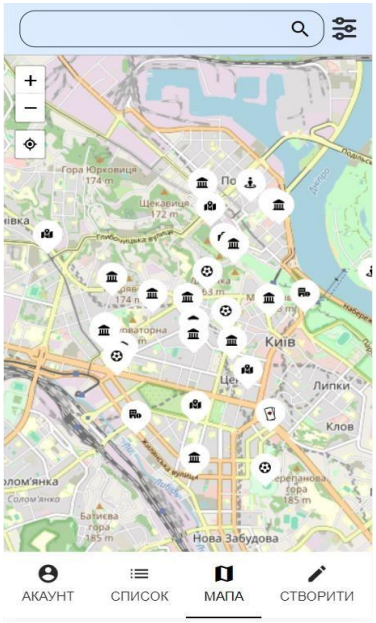


Рисунок 13 – Мапа з подіями

Якщо натиснути на мітку з подією, то відкриється панель (див. рис. 14) з короткою інформацією про подію та з двома кнопка, одна з яких будує маршрут (див. рис. 15) від поточної геолокації до події, а інша здійснює перехід на сторінку події.

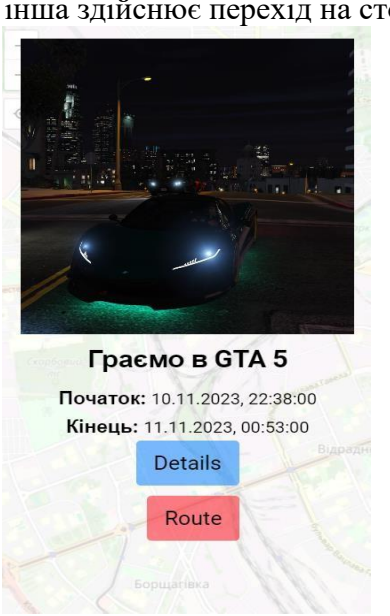


Рисунок 14 – Панель, що відкривається на мапі

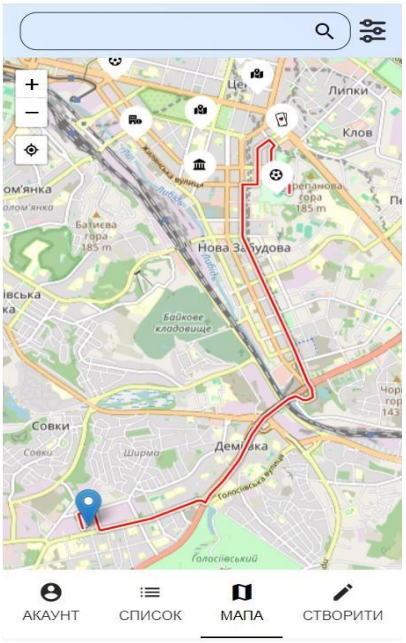


Рисунок 15 – Побудова маршруту на мапі

Якщо на нижній панелі натиснути кнопку "СТВОРИТИ" то користувач перейде на сторінку створення події, що відбувається у два етапи. На першому етапі (рис. 16) необхідно вказати назву події, обрати тип події, час початку та час закінчення, адреса, яка обирається на мапі. На другому етапі (рис. 17) опціонально можна додати зображення події та опис.

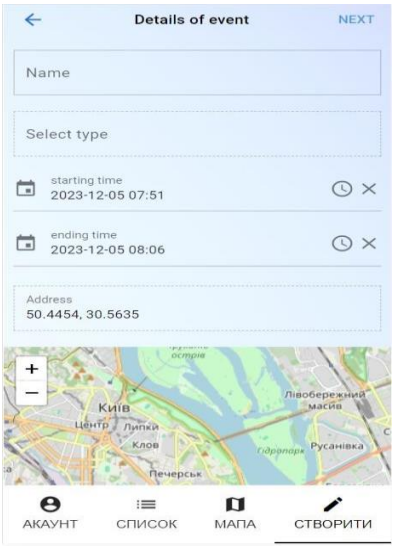


Рисунок 16 – Перший етап створення події

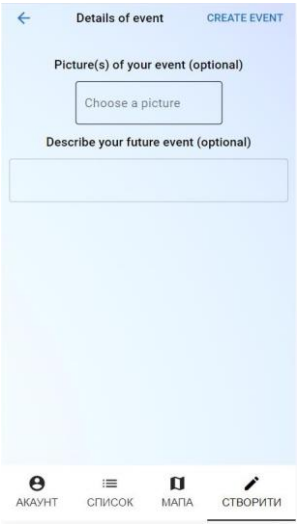


Рисунок 17 – Другий етап створення події

Якщо користувач натисне на нижній панелі кнопку "АКАУНТ", то він перейде на свій профіль (див. рис. 18), де буде аватар профілю, електронна пошта, ім’я користувача, опис профілю та події, які створив користувач або, до яких приєднався та кнопки для редагування та виходу з акаунту.

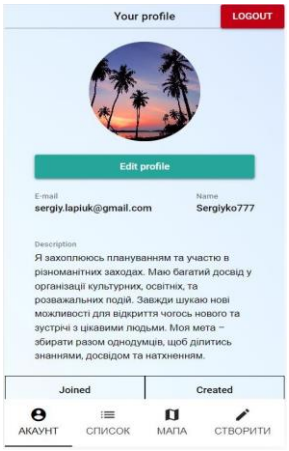


Рисунок 18 – Сторінка користувача

Також, якщо роль користувача "Адміністратор", то йому доступна сторінка управління користувачами, де він може блокувати, розблоковувати, та видаляти користувачів.

ПРОЦЕС РОЗРОБКИ

Розподіл ролей в команді.

Команда складається зі студентів факультету комп'ютерних наук та кібернетики та студентки географічного факультету.

Олексій Мацуй, студент групи ТТП-42, виступає у ролі керівника проєкту. Він відповідає за загальне керівництво проєктом, організацію роботи команди, а також забезпечення дотримання термінів та якості роботи. Окрім адміністративних обов'язків, Олексій також активно брав участь у технічній стороні проєкту. Він розробив зокрема систему фільтрації подій, та також працював над інтеграцією бекенду з фронтендом.

Олексій також відіграв важливу роль у підтримці команди, допомагаючи вирішувати технічні проблеми. Його знання та досвід були дуже корисними для команди. Крім цього, студент також виконував обов'язки фахівця з документації та тестувальника.

Сергій Лапюк, студент групи ТТП-42, відіграє ключову роль у розробці фронтенду та бекенду проєкту. Він відповідальний за створення важливих елементів інтерфейсу користувача, включаючи сторінки списку подій, сторінку профілю користувача та функціонал редагування профілю. Сергій також працював над інтеграцією зображень у бекенд-систему, забезпечуючи їх ефективне зберігання, оптимізацію та вивід на фронтенд. На додаток до своїх технічних завдань, Сергій також виконував роль модератора. Він допомагав у координації робочих процесів та комунікації всередині команди, гарантуючи плавність робочих процесів та ефективну взаємодію між учасниками проєкту.

Олексій Астраханцев, студент групи ТТП-42, є ключовою фігурою у розробці бекенду проєкту. Він забезпечує стабільність та ефективність серверної частини. Олексій виступив головним бекенд розробником, взявши на себе відповідальність за розробку більшості методів роботи з основною базою даних та базою даних для авторизації. Це включало створення ефективних, безпечних та масштабованих рішень для управління даними, які є критично важливими для функціонування вебзастосунку та забезпечення безпеки інформації користувачів. Крім основних обов'язків, Олексій також активно допомагав іншим учасникам команди у вирішенні технічних проблем, особливо тих, що пов'язані з бекенд-розробкою.

Микола Васильчук, студент групи ТТП-42, виконує значну роль у розробці фронтенду та бекенду проєкту, з особливим акцентом на роботу над фронтенд-частиною. Микола брав на себе відповідальність за розробку декількох критично важливих компонентів. Він працював над сторінкою створення події, де користувачі можуть додавати нові події до системи. Також він розробив сторінку авторизації, забезпечуючи безпечний та ефективний доступ користувачів до платформи. Крім того, Микола був відповідальний за сторінку управління користувачами, де можливе адміністрування профілів. Робота Миколи включала також інтеграцію з бекенд-сервісами.

Юрій Підлетейчук, студент групи ТТП-42, відіграє значну роль у розвитку як бекенду, так і фронтенду проєкту, при цьому зосереджуючись переважно на розробці фронтенду. Юрій був відповідальний за розробку важливої функціональної частини вебзастосунку – сторінки з інтерактивною мапою. На цій сторінці відображаються різні події, що відбуваються у місті, з можливістю перегляду деталей та прокладання маршруту до місця проведення події. Крім мапи, Юрій також працював над розробкою сторінки події, де користувачі можуть приєднатися до події, отримувати детальну інформацію про конкретні події, включаючи опис, час, місце проведення та інші важливі деталі.

Надія Огійчук, студентка географічного факультету, відіграє важливу роль у зборі та аналізі інформації, пов'язаної з точками міського активізму. Надія відповідає за надання актуальної та точної інформації про події міського активізму. Це включає збір детальної інформації про кожну подію, а також пошук та відбір відповідних зображень, які використовуються для ілюстрації цих подій на сторінках. Крім збору інформації, Надія також допомагала у процесі вибору іконок для різних типів подій. Це включало пошук і відбір іконок, які точно відображають характер кожної події та допомагають користувачам швидко ідентифікувати тип події на інтерфейсі вебзастосунку.

Методологія та інструментарій розробки.

У процесі розробки нашого проєкту ми обрали Scrum як основну методологію роботи. Scrum – це гнучкий фреймворк, який підтримує команди у вирішенні складних завдань, дозволяючи їм ефективно адаптуватися до змін та створювати цінність.

Scrum базується на декількох ключових принципах:

1. Власник продукту (Product Owner) визначає та організує робочі завдання у беклозі продукту (Product Backlog) для вирішення комплексних проблем;
2. Scrum команда (Scrum Team) працює над вибраною частиною беклогу, перетворюючи її у цінний інкремент продукту протягом обмеженого часового вікна, званого спринтом (Sprint);
3. По завершенню кожного спринту, Scrum команда та зацікавлені сторони аналізують виконану роботу та вносять корективи у плани наступного спринту;
4. Повторюваність процесу є ключовою, дозволяючи команді постійно покращувати процес роботи та результати.

Важливою характеристикою Scrum є його гнучкість та адаптивність. Фреймворк навмисно залишений неповним, надаючи командам простір для адаптації та використання власного досвіду і знань. Scrum сприяє самоорганізації та колективному інтелекту, фокусуючись на взаємодії та комунікації між членами команди, а не на жорстких інструкціях. У рамках Scrum ми можемо інтегрувати різноманітні процеси, техніки та методи, які найкраще підходять для конкретного проєкту. Scrum виявляє ефективність наявних управлінських підходів, дозволяючи команді виявляти та усувати недоліки, що значно підвищує продуктивність та якість роботи [7].

У нашому проєкті для розробки прототипу інтерфейсу ми використовували Figma. Це сучасний інструмент дизайну, який дозволяє командам легко співпрацювати над створенням, прототипуванням та ітерацією дизайну інтерфейсу. Також за допомогою Figma ми створювали багато схем, зокрема схему бази даних.

Для розподілу команд наша команда використовувала сервіс Jira Software. Це загальна платформа для управління проєктами, орієнтована на команди розробників програмного забезпечення. Вона використовується для відстеження помилок, додавання нових функцій і призначення та виконання завдань. Jira має багато функцій керування продуктами, які корисні під час створення, виконання та завершення проєкту. Використання Jira також дозволяє командам налаштувати платформу відповідно до своїх потреб. Можна використовувати основні функції Jira, а також додавати інші функції за допомогою інтеграції [8]. На рис. 19 можна побачити дошку одного з наших спринтів.

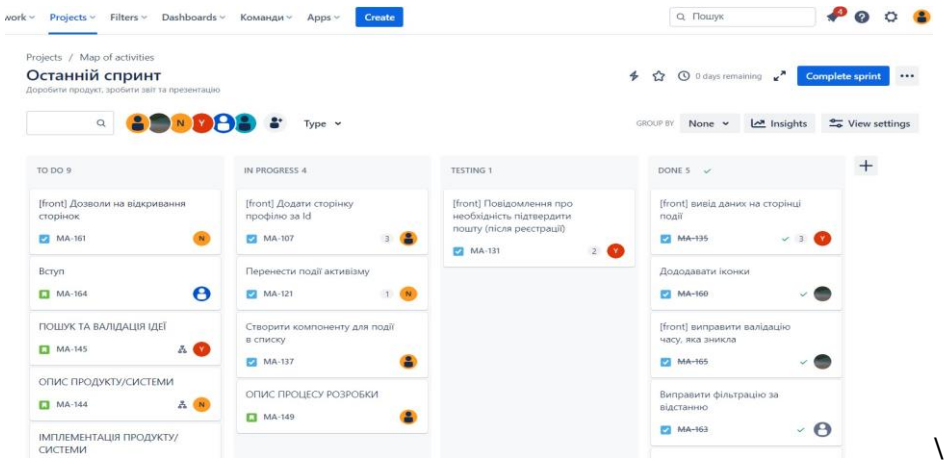


Рисунок 19 – Дошка спринту в Jira

У нашому проєкті для ведення документації та здійснення ретроспектив спринтів ми використовували Confluence. Це потужний інструмент для спільної роботи над документацією, розроблений Atlassian, який дозволяє командам ефективно організовувати та управляти документацією проєкту.

Confluence дозволяє зберігати всю проєктну документацію в одному централізованому місці, забезпечуючи легкий доступ усім членам команди. Платформа підтримує спільне редагування документів, коментування та обмін знаннями, що сприяє глибшому взаєморозумінню та ефективній командній роботі.

Цей інструмент був особливо корисним для проведення ретроспектив спринтів, дозволяючи команді збирати зворотній зв'язок, оцінювати процеси та планувати покращення на майбутнє.

Використання Confluence значно підвищило ефективність управління проєктною документацією (див. рис. 20) та сприяло більшій злагодженості та організованості робочого процесу в нашій команді.

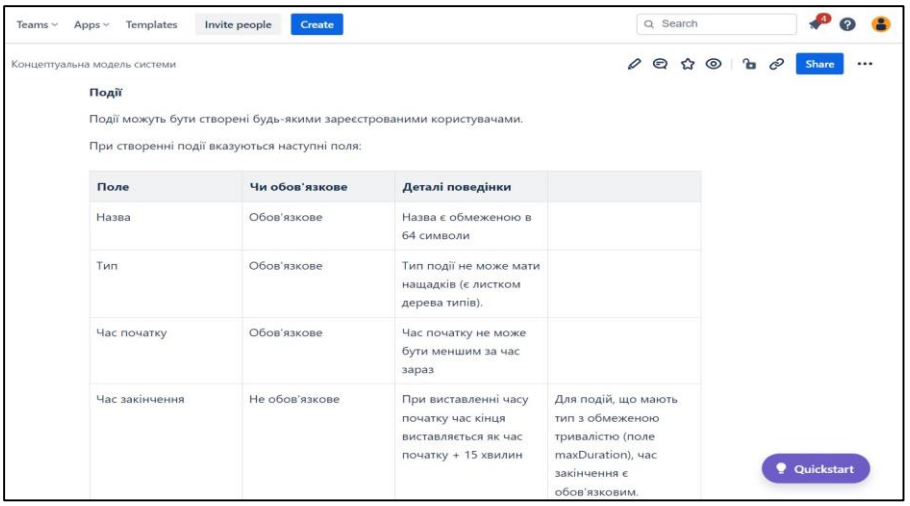


Рисунок 20 – Документація в Confluence

Наша команда використовувала Bitbucket як основну платформу для управління кодом, зокрема для розробки як фронтенду, так і бекенду проєкту. Bitbucket – це вебсервіс для управління проєктами та спільної роботи над кодом, розроблений компанією Atlassian. Ця платформа надає командам інструменти для ефективної роботи з репозиторіями коду, зокрема на базі системи контролю версій Git.

Великим плюсом використання Bitbucket стало те, що платформа легко інтегрується з іншими продуктами Atlassian, такими як Jira та Confluence, що дозволяє забезпечити плавний робочий процес. Bitbucket підтримує систему Pull Requests, яка дозволяє проводити код-рев'ю, сприяючи підвищенню якості коду та співпраці в команді.

На рис. 21 зображено приклад використання Bitbucket нашою командою для управління кодом.

Використання Bitbucket в нашому проєкті дозволило нам ефективно управляти кодом, спрощуючи співпрацю та забезпечуючи якість розробки.

Мови програмування та база даних.

Для написання даної роботи була використана мова програмування – C#.

Наразі C# займає п’яту позицію за популярністю серед мов програмування [9].

C# – це об’єктно-орієнтована мова програмування. Вона була створена в період з 1998 по 2002 рік командою інженерів Microsoft під керівництвом Андреса Хейлсберга та Скотта Вільтаумота.

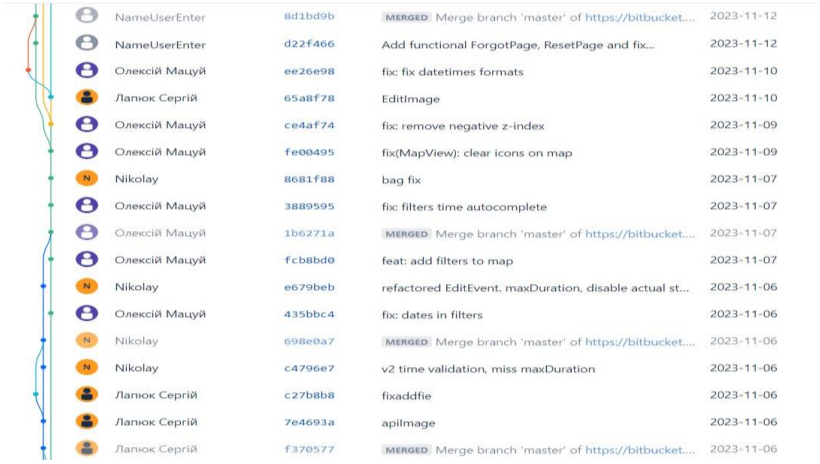


Рисунок 21 – Коміти учасників команди в Bitbucket

Мова входить в сім’ю С-подібних мов. Синтаксис наближений до Java та C++. Її особливості:

- статична типізація;
- підтримується поліморфізм;
- підтримується перевантаження операторів;
- доступна делегація, атрибути, дії, узагальнені типи та анононімні функції.

Розробка Microsoft багато особливостей успадкувала від інших мов програмування. При цьому творці нової мови виключили із свого проєкту багато практик та специфікацій, які вважалися проблемними.

JavaScript – це кросплатформна, об'єктно-орієнтована скриптова мова, яка є невеликою. Всередині середовища виконання JavaScript може бути пов'язаний з об'єктами даного середовища і надавати програмний контроль над ними.

JavaScript включає стандартну бібліотеку об'єктів, наприклад Array, Date і Math, а також базовий набір мовних елементів, наприклад, оператори та керуючі конструкції [10].

У рамках групового проєкту наша команда використовувала Neon, що є передовим серверним рішенням для баз даних, яке базується на PostgreSQL і надає можливості, які є привабливими для нашої команди. Neon пропонує такі функції, як автоматичне масштабування, розгалуження, безмежне зберігання тощо. Це робить його дуже привабливим для розробників, які шукають гнучкість та інноваційні рішення в управлінні базами даних. У нашому випадку команда віддала перевагу Neon, бо він надає можливість зручно працювати з віддаленою базою даних.

Огляд використаних технологій.

Технології Entity Framework Core, ASP.Net Core Web API.

Entity Framework Core (EF Core) є об'єктно-орієнтованою, легковаговою і розширюваною технологією від компанії Microsoft для доступу до даних. EF Core є ORM-інструментом (object-relational mapping – відображення даних на реальні об'єкти). Тобто EF Core дозволяє працювати з базами даних, але є більш високий рівень абстракції:

EF Core дозволяє абстрагуватися від самої бази даних та її таблиць та працювати з даними незалежно від типу сховища [11].

Entity Framework Core підтримує багато різних систем баз даних. Таким чином, ми можемо через EF Core працювати з будь-яким СКБД, якщо для неї є потрібний провайдер.

ASP.NET Core Web API - це фреймворк для створення RESTful вебсервісів. Він дозволяє розробникам створювати вебсервіси, які відповідають архітектурі REST, яка є стандартом для розробки вебсервісів.

Огляд Quasar Framework

Quasar обраний для розробки вебзастосунку з кількох причин. По-перше, це надійний фреймворк, який дозволяє створювати додатки для різних платформ, включаючи десктопи, мобільні пристрої та браузери. Крім того, Quasar відомий своєю високою швидкістю та продуктивністю, що робить його відмінним вибором для створення ефективних додатків. Нарешті, він є розширюваним, що дозволяє легко розширити функціональність додатку у майбутньому [12].

Технології Leaflet, OpenStreetMap

Вебзастосунок, який розробляла наша команда, використовує мапу для відображення розташування користувачів, створення та відображення точок з подіями, які створюють користувачі, а також створення маршрутів від поточної геопозиції до обраної події. Для реалізації цих завдань ми використовували бібліотеку Leaflet JavaScript і базу даних OpenStreetMap.

Leaflet – це безкоштовна, відкрита бібліотека JavaScript для створення інтерактивних карт. Вона має простий у використанні API, який дозволяє швидко та легко створювати карти [13].

OpenStreetMap - це безкоштовна, відкрита база даних мап, створена спільнотою. Вона містить точні та актуальні дані про географічні об'єкти по всьому світу.

Планування спринтів та рефлексія

Перед тим, як планувати наступний спринт, наша команда завжди проводила ретроспективу, під час цього ми використовували Confuence (див. рис. 22). На цих зустрічах ми аналізували наші попередні досягнення та визначали, що варто зберегти, що варто почати робити та що слід припинити. Також ми розглядали весь спринт в цілому, оцінюючи, що вдалося зробити і що залишилося недоробленим. Якщо виникали випадки, коли деякі завдання не були виконані, ми вивчали причини цього і намагалися уникнути схожих ситуацій у майбутньому.

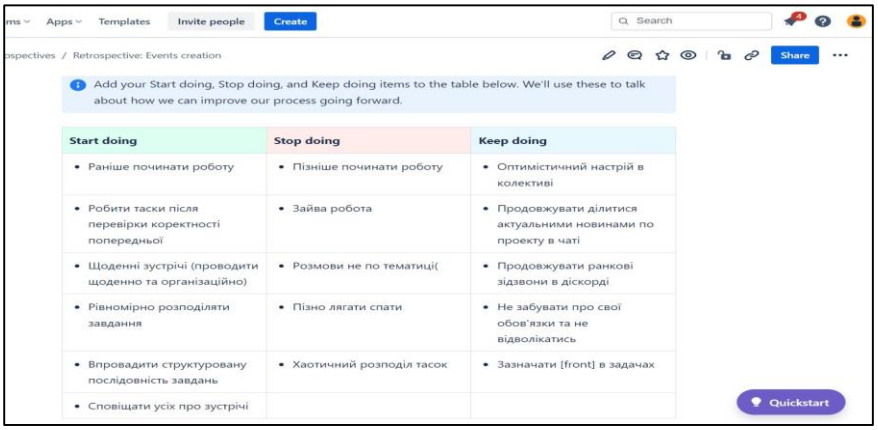


Рисунок 22 – Ретроспектива в Confluence

Після завершення ретроспективи ми приступали до планування наступного тижневого спринту. Зазвичай це відбувалося по вівторках після демонстрації результатів минулого спринту викладачам з використанням Discord [14] або Google Meet (див. рис. 23) та Jira для розподілу завдань (див. рис. 24).

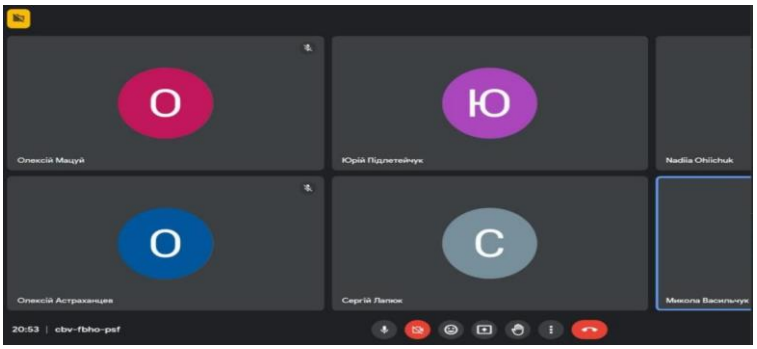


Рисунок 23 – командна зустріч в Google Meet

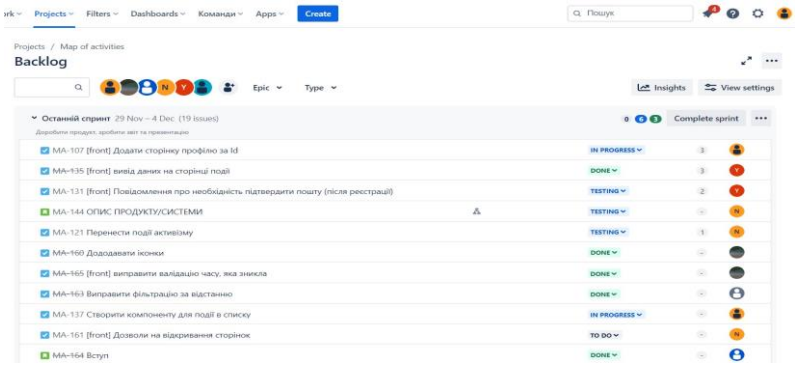


Рисунок 24 – розподіл завдань за допомогою Jira

Загалом наша команда працювала дуже злагоджено, атмосфера в команді була хорошою. Ми проводили щоденні онлайн-зустрічі, що покращувало якість та швидкість розробки. Під час цих зустрічей ми з’ясовували питання, допомагали один одному, коли виникали технічні проблеми. Зрозуміло, що як і під час будь-якого робочого процесу у нас інколи виникали суперечки. Але їх кількість була мінімальна і вирішувались вони досить швидко. Всі учасники команди працювали старанно та виконували поставлені їм завдання під час планування спринту.

Загалом ми виконали план, у результаті отримали повнофункціональний застосунок. Отриманий досвід надзвичайно цінний і слугує джерелом навчання та збагачення наших навичок у розробці програмного забезпечення. Ми плануємо продовжити підтримку та розвиток застосунку.

ВИСНОВКИ

Аналіз подібних застосунків та відсутності аналогу українського відповідника показав потребу у створенні власного додатку, який буде легко відображати та створювати різні події з відображенням їх у реальному часі та можливістю приєднання.

"Map of activities " - є інструментом який швидко та легко об’єднує різні типи заходів та подій у реальному часі. Цей додаток надає можливість одночасно потенційним користувачам та організаторам подібних заходів оперативно подавати інформацію про події та заходи, а користувачам швидко та зручно знаходити та брати участь у заходах за

їхніми індивідуальними уподобаннями від спортивних заходів і концертів до культурних подій та подій міського активізму.

Крім того, він забезпечує авторизацію користувачів. Авторизовані користувачі можуть створювати нові події, які будуть відображатися на мапі подій. Поєднуючи у собі різні типи подій застосунок "Map of activities " може знайти застосування для покращення соціальної взаємодії та стимулюванні соціальної активності.

Процес розробки також дозволив закріпити знання з проєктування баз даних, створення контролерів, розробки інтерфейсу та дизайну, представлень та моделей. У процесі розробки додатку "Map of activities" ми покращили навички співпраці та командної роботи. Використання Jira для планування спринтів і дотримання методології Scrum допомогли ефективно організовувати робочий процес, що сприяло успішній командній роботі.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Visual Studio 2022 [Електронний ресурс] – Режим доступу до ресурсу: https://visualstudio.microsoft.com/ru/
2. Bitbucket [Електронний ресурс] – Режим доступу до ресурсу: https://bitbucket.org/map-of-activities/workspace/overview/
3. Quasar [Електронний ресурс] – Режим доступу до ресурсу: https://quasar.dev/
4. https://www.meetup.com/login/ [Електронний ресурс] – Режим доступу до ресурсу: https://www.https://www.meetup.com/login/.com/login/.
5. Allevents [Електронний ресурс] – Режим доступу до ресурсу: https://allevents.in/.
6. VisualParadigm [Електронний ресурс] Режим доступу до ресурсу: https://online.visual-paradigm.com/
7. Швабер К. Посібник зі Скраму / К. Швабер, Д. Сазерленд.
8. What is Jira? [Електронний ресурс] – Режим доступу до ресурсу: https://airfocus.com/glossary/what-is-jira/.
9. Tiobe [Електронний ресурс] – Режим доступу до ресурсу: https://www.tiobe.com/tiobe-index/.
10. MDN Web Docs [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org.
11. Документація Microsoft [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/uk-ua/ef/core/.
12. What is Quasar? [Електронний ресурс] – Режим доступу до ресурсу: https://quasar.dev/introduction-to-quasar/.
13. Leaflet [Електронний ресурс] – Режим доступу до ресурсу: https://leafletjs.com/index.html.
14. Discord [Електронний ресурс] – Режим доступу до ресурсу: https://discord.com/.

ІНФОРМАЦІЙНИЙ ХАБ ДЛЯ ВІЙСЬКОВИХ ТА ПОСТРАЖДАЛИХ ВІД ВІЙНИ

Дмитро Алексенко, Аліна Беденко, Владислав Бурлака, Світлана Гнатюк, Юлія Недавня, Юлія Соломаха, Владислав Спотар, Наталія Філімончук

Дана робота висвітлює процес реалізації мобільного застосунку "WelcomeHome", створеного для спрощення процесу адаптації військовослужбовців до цивільного життя після повернення з війни. Застосунок включає такі функціональні модулі: психологічна підтримка, реабілітація, соціальні виплати, підтримка спільноти та вакансії.

Реалізація проєкту була здійснена за підтримки військових, фахівців у галузі психології, соціальної роботи та IT. Зокрема, в рамках проєкту було розроблено інтуїтивно зрозумілий інтерфейс, який допомагає користувачам легко переміщатися по розділах застосунку, забезпечуючи легкий доступ до необхідної інформації та ресурсів. Особлива увага була приділена інтеграції з зовнішніми базами даних для актуалізації вакансій.

Процес розробки застосунку "WelcomeHome" був поділений на два семестри. В першому семестрі команда зосередилася на аналізі потреб цільової аудиторії, зборі інформації та створенні мінімально життєздатного продукту (MVP) з базовими функціями. Другий семестр присвячений покращенню застосунку та розробці нових функціональних можливостей.

This work discusses the development and implementation of the "WelcomeHome" mobile application, specifically designed to ease the transition of military personnel back into civilian life after their service. The application is structured around several key functional modules: psychological support, rehabilitation services, guidance on social benefits, community support initiatives, and resources for job opportunities.

The project was analyzed with the collaboration of military experts and experts in psychology, social work, and information technology. A significant aspect of this collaboration was the creation of an intuitive user interface. This interface allows users to navigate seamlessly through the application, ensuring they have quick and straightforward access to essential services and information. Moreover, the application was designed to integrate with external databases, which keeps the listings of job vacancies and educational programs up to date, further aiding users in their reintegration efforts.

The development process was strategically divided into two semesters to optimize outcomes. During the first semester, the primary focus was on conducting a thorough analysis of the target audience's needs. This phase also included extensive information gathering which informed the creation of a Minimum Viable Product (MVP). This MVP featured basic functionalities that were essential for initial user interaction and feedback collection. The second semester was geared towards refining the application based on feedback received from the MVP. During this period, the team concentrated on enhancing existing features and developing new functionalities to better serve the users' needs and improve the overall user experience.

АДАПТАЦІЯ ВІЙСЬКОВОСЛУЖБОВЦІВ, МОБІЛЬНИЙ ЗАСТОСУНОК, СОЦІАЛЬНІ ВИПЛАТИ, МІНІМАЛЬНО ЖИТТЄЗДАТНИЙ ПРОДУКТ, РОЗРОБКА ФУНКЦІОНАЛУ, ЦІЛЬОВА АУДИТОРІЯ.

ADAPTATION OF MILITARY SERVANTS, MOBILE APPLICATION, SOCIAL PAYMENTS, MINIMUM VIABLE PRODUCT, FUNCTIONAL DEVELOPMENT, TARGET AUDIENCE.

ВСТУП

У сучасному світі адаптація військовослужбовців до цивільного життя після повернення з війни є важливим соціальним викликом. Повернення до мирного життя вимагає комплексного підходу та використання спеціалізованих інструментів, здатних задовольнити різноманітні потреби військових. У відповідь на цю потребу, було розроблено мобільний застосунок "WelcomeHome", що має на меті полегшити процес адаптації військових, надаючи доступ до необхідних ресурсів та підтримки.

Актуальність теми полягає у зростаючій потребі в інтегрованих рішеннях, які можуть сприяти психологічній стабілізації, соціальній адаптації та професійній реінтеграції військовослужбовців. Розробка такого проєкту дозволяє забезпечити цілісний підхід до вирішення цих важливих соціальних питань, водночас сприяючи науковому та технологічному розвитку у галузі мобільних додатків.

Методи дослідження включають аналіз потреб цільової аудиторії, проєктування інтуїтивно зрозумілого інтерфейсу та інтеграцію з зовнішніми базами даних. Також було використано ітеративний підхід Scrum для гнучкого управління проєктом, що дозволяло оперативно вносити зміни та вдосконалення в процесі розробки.

Основною ціллю проєкту є створення ефективного інструменту, який допоможе військовим успішно повернутися до цивільного життя. Завдання проєкту охоплюють розробку ключових модулів застосунку, що стосуються психологічної підтримки, реабілітації, оформлення соціальних виплат, пошуку роботи та підтримки від спільноти. Крім прикладних завдань, професійні цілі включають закріплення знань та поглиблення компетенцій у сферах веброзробки та мобільних технологій через практичний досвід із використанням таких інструментів як SQL Server, ASP.NET Core 6, Entity Framework у бекенді та React Native у фронтенді.

ПОШУК ТА ВАЛІДАЦІЯ ІДЕЇ

Опис пошуку ідеї продукту

Під час аналізу ідей для створення продукту було досліджено актуальні проблеми українського суспільства, що виникають у повсякденному житті осіб, які зіштовхуються з цими викликами. Актуальні проблеми були виявлені у погіршенні рівня економіки держави та зниженні рівня і якості життя населення, а також у соціальних проблемах, які виникли або загострилися внаслідок війни чи повномасштабного вторгнення Росії проти України.

Серед виявлених проблем можна виокремити:

- Інфляція, зростання цін, зменшення кількості робочих місць, падіння економічної спроможності держави.
- Лікування поранених цивільних, які отримали травми внаслідок військових дій. Також, виявлено не завжди коректне ставлення суспільства до осіб з видимими пошкодженнями, відсутністю деяких кінцівок або протезами.
- Надмірне навантаження на волонтерів, нестача ресурсів (певних навичок, фінансів, мотивації, нестача людей у волонтерському секторі), велика відповідальність та ризики для життя, що спричиняють виснаження та вигорання.
- Реабілітація демобілізованих військовослужбовців та їхнє повернення до цивільного життя. Складність пошуку інформації про реабілітаційні центри, центри надання психологічної допомоги. Соціальна та економічна адаптація особливо актуальна для військових з отриманими фізичними та психологічними травмами, несумісними з попередньою діяльністю. Психологічні та емоційні виклики після військової служби.

- Надзвичайно довгий, складний і незрозумілий процес оформлення соціальних виплат.

З часом було зрозуміло, що відсутність єдиної бази з корисною інформацією для демобілізованих військовослужбовців та людей, які постраждали від війни, є дуже актуальною проблемою, і в нас є пропозиція щодо її вирішення. Для того, щоб спростити адаптацію та зробити її максимально "безболісною", було вирішено створити централізовану інформаційну платформу, щоб зібрати всю необхідну інформацію в одному місці.

Продуктом, розробленим для вирішення цієї проблеми, став мобільний застосунок для демобілізованих військовослужбовців та людей, які постраждали від війни – "WelcomeHome". У цьому додатку зібрана необхідна інформація про реабілітаційні центри, служби психологічної підтримки, інформація про процес оформлення соціальних виплат, перелік груп психологічної підтримки та благодійних заходів, а також перелік доступних вакансій для військових.

Огляд наявних на ринку IT-продуктів – конкурентів

Наразі єдиним дещо схожим додатком на "WelcomeHome" є застосунок "База" від "Вільного Вибору [1].

Застосунок спрямований допомогти ветеранам, родичам або близьким військових та цивільним стабілізувати психологічний стан. Також можна отримати інформацію про основні види психологічної самодопомоги та поставити питання фахівцям. Застосунок **"База"** має такі підрозділи:

- Інструменти. Тут можна дізнатися, як працювати з емоціями та ділитися власним досвідом. Наприклад, що робити після контузії;
- Базове. У розділи йдеться про фізичні та психологічні стани, з якими можуть стикатися ветерани або ветеранки;
- Стоп кнопка. Тут можна отримати допомогу, якщо "накрило";
- Обране. В цей розділ можна додавати усю корисну інформацію та вправи, які ви знайшли у застосунку. Вони будуть доступні навіть в режимі офлайн;
- Контакти. Цей розділ для тих, хто прагне поспілкуватися із фахівцями, поставити питання або додати контакти (свої або інші, корисні).

"Тут можна почитати про тривогу, наслідки легкої черепно-мозкової травми, хронічний біль, поганий сон, депресію, нав’язливі думки та знайти інструменти, як працювати з цим", – йдеться в описі.

Вхід у застосунок здійснюється як через власний акаунт, так і без реєстрації. Користувачі можуть обирати теми та голоси, які звучатимуть у вправах (є чоловічий та жіночий). Також є вбудований є трекер настрою, можливість записувати свої думки та стани, ставити нагадування.

Щоб завантажити застосунок необхідно перейти в App Store для власників Apple та в Google Play для власників Android.

Мапа продукту

Застосунок був розроблений для вирішення наступних проблем:

Психологічні та емоційні виклики. Після військової служби військові та постраждалі від війни часто стикаються з серйозними психологічними проблемами, такими як ПТСР, депресія та тривога. Ці стани можуть перешкоджати їм повертатися до нормального життя і спричиняти загрозу їхньому фізичному та емоційному благополуччю.

Соціальна та економічна адаптація. Після війни важко знайти роботу та забезпечити собі та родині достатній рівень життя. Це особливо актуально для

військових з отриманими фізичними та психологічними травмами, несумісними з попередньою діяльністю.

Інформаційна обізнаність. Наразі немає жодного єдиного інформаційного порталу зі списком реабілітаційних центрів, що надають безкоштовне або не дороге лікування та встановлення протезів українським військовим. Зазвичай вся інформація передається із рук в руки через знайомих волонтерів і більшість людей навіть не мають уявлення про допомогу, яку вони можуть отримати не тільки в Україні, а і за кордоном.

Щодо сегментів клієнтів, то застосунок перш за все орієнтований на:

- Військових: це основна цільова аудиторія. Даний застосунок може надавати їм необхідну підтримку та інформацію для подолання психологічних, фізичних та соціальних перешкод.
- Постраждалих від війни: цей сегмент включає людей з фізичними та психологічними травмами через досвід війни.
- Роботодавців: підприємства, які можуть бути зацікавлені в наймі військових або постраждалих від війни, можуть публікувати доступні вакансії.
- Волонтерів: саме вони можуть наповнювати застосунок інформацією, тобто додавати нові психологічні служби, реабілітаційні центри, створювати вакансії, додавати соціальні виплати із кроками для їх отримання.

Унікальною ціннісною пропозицією цього застосунку є його комплексний підхід до підтримки військових та постраждалих від війни у процесі повернення до нормального життя. На даний момент у цій конкретній ніші конкурентів взагалі немає. Це робить цей застосунок першим на ринку і надає йому перевагу в розробці та наборі користувачів.

Розуміючи всю глибину та складність питання адаптації до цивільного життя після війни, до процесу вдосконалення функціональних можливостей застосунку було долучено головних бенефіціарів додатку - військовослужбовців, та разом були напрацьовані виважені рішення. Відтак, застосунок "WelcomeHome" міститиме такий функціонал:

- Блок із психологічною підтримкою.
- Блок із інформацією про реабілітаційні центри по всьому світу з інтерактивною мапою та контактами.
- Блок із інформацією про соціальні виплати.
- Блок із психологічними службами, благодійними заходами та групами психологічної підтримки.
- Блок із інформацією щодо роботи та/або можливостей освоєння нової професії.

Задля охоплення якнайбільшого кола аудиторії, було передбачено поширення інформації про застосунок через такі канали комунікації:

Соціальні мережі. Запуск рекламних кампаній та створення активних сторінок та спільнот на платформах, таких як Facebook, Instagram, Telegram канали тощо.

Поширення інформації про застосунок серед волонтерів. Мобілізація та підтримка активних волонтерів для поширення інформації про застосунок серед військових та постраждалих від війни.

Благодійні фонди та організації. Партнерство та співпраця з благодійними фондами та організаціями, які вже допомагають військовим та постраждалим від війни.

Державне партнерство. Звернення до урядових структур та програм для отримання підтримки та фінансування для розробки та рекламної кампанії.

З огляду на ідею та соціальне значення розробленого застосунку, передбачається залучення фінансування для його розробки та функціонування через такі джерела фінансування: **державне** (уряди та міністерства оборони можуть виділяти кошти на проєкти та ініціативи, що спрямовані на підтримку військових та постраждалих від

війни. Застосунок може звертатися до державних програм фінансування для забезпечення стабільного фінансування та розвитку) та **грантове** (організації громадського сектору, фонди та благодійні організації можуть надавати гранти на проєкти, спрямовані на підтримку ветеранів та військових). Застосунок може звертатися до цих джерел фінансування для забезпечення розробки, підтримки та поширення.

Важливо відзначити, що всі витрати, необхідні для реалізації даної ідеї, можуть бути поділені на дві групи: змінні та постійні витрати. До змінних витрат належать: дослідження та розробка; забезпечення безпеки та конфіденційності; маркетинг та просування. До постійних витрат відносяться: інфраструктура та хостинг; підтримка та обслуговування користувачів; гонорари та зарплати для персоналу.

Для вимірювання ефективності бізнесу, пов'язаного з застосунком "WelcomeHome", планується застосування наступних метрик:

1. Кількість активних користувачів: Вимірювання щоденної та місячної активності користувачів дозволить оцінити загальний зацікавленість та залученість у застосунок.
2. Взаємодія з контентом: Аналіз кількості переглядів, завантажень та взаємодій з ресурсами застосунку, включаючи використання інформаційних матеріалів та участь у психологічних групах.
3. Конверсія та утримання користувачів: Відстеження кількості нових реєстрацій порівняно з кількістю користувачів, які залишаються активними протягом певного періоду.
4. Задоволеність користувачів: Регулярне збирання відгуків через опитування та аналіз відгуків для оцінки задоволеності користувачів послугами застосунку.
5. Кількість розміщених вакансій та реакція на них: Вимірювання ефективності блоку з вакансіями через аналіз кількості розміщених вакансій та відповідей на них.

Унікальність застосунку "WelcomeHome" визначається наступними особливостями:

Інтеграція всебічних послуг: Застосунок може стати єдиним на ринку, який інтегрує всі необхідні сервіси для військових та постраждалих від війни в одному місці, включно з реабілітацією, психологічною підтримкою, інформацією про соціальні виплати та можливості працевлаштування.

Комплексний підхід до психологічної підтримки: Застосунок не тільки надає базову інформацію, а й забезпечує глибоку психологічну підтримку через доступ до групових та індивідуальних сесій, що дозволяє користувачам ефективно працювати над відновленням.

Співпраця з волонтерськими та державними організаціями: Тісна співпраця з організаціями та державними інституціями забезпечує надійне джерело інформації та ресурсів, що сприяє забезпеченню високої якості наданих послуг.

Адаптованість до потреб користувачів: Розробка застосунку проводиться з безпосередньою участю його кінцевих користувачів — військовослужбовців, що забезпечує максимальну адаптацію функціоналу під їхні реальні потреби та вимоги.

Ці унікальні особливості надають "WelcomeHome" вирішальну перевагу на ринку, дозволяючи застосунку надавати більш цілісний і дієвий підхід до реінтеграції військовослужбовців і постраждалих від війни у цивільне життя.

SWOT-аналіз

У таблиці 1 наведений SWOT-аналіз застосунку Welcome Home.

Таблиця 1. SWOT-аналіз застосунку.

Strengths	**Weaknesses**
- Соціальний проєкт, що здатний допомогти військовим - Складність проєкту	- Небезпека зберігання даних про військових в одній базі
Opportunities	**Threats**
- Отримання держ фінансування як GovTech проєкт - Інтеграція з застосунком Дія - Можливість допомогти великій кількості людей повернутися до нормального життя після війни - Інтеграція з Djinni, Coursera, LinkedIn, Mono	- Небезпека бути зламаними російськими кібер-атаками - Небезпека витоку особистих даних військових

Огляд використаних технологій. Для розробки мобільного додатку "WelcomeHome" було обрано низку технологій, що дозволяють оптимізувати роботу, забезпечити високу продуктивність, і забезпечити безпеку та надійність системи.

ASP.NET Core Web API вибрано через його високу продуктивність і гнучкість [2]. Це середовище розробки від Microsoft дозволяє створювати потужні вебзастосунки та вебсервіси, які легко масштабуються та інтегруються з різноманітними системами управління базами даних і іншими технологіями. Використання ASP.NET Core дозволяє забезпечити високу продуктивність серверної частини додатку, а також безпечне з'єднання з клієнтською частиною.

React Native було обрано для розробки клієнтської частини додатку завдяки його ефективності в побудові інтерактивних користувацьких інтерфейсів. Ця JavaScript-бібліотека дозволяє швидко відображати зміни даних без потреби перезавантаження сторінки, що є критично важливим для забезпечення гарного досвіду користувача.

Redux використовується разом з React для керування станом додатку. Це дає можливість легко управляти станом, який може бути спільним між різними компонентами без необхідності пропускати властивості через всю ієрархію компонентів, що забезпечує більш організовану та ефективну структуру коду [4].

SQL Server використовується як система управління базами даних через її надійність, швидкість і можливості інтеграції з іншими продуктами Microsoft, що робить її ідеальним вибором для корпоративних додатків. SQL Server забезпечує потужні інструменти для управління даними, які критично необхідні для забезпечення цілісності та безпеки інформації у такому важливому застосунку, як "WelcomeHome".

NUnit застосовується для тестування .NET додатків, забезпечуючи можливість ефективного написання та виконання тестів. Цей фреймворк допомагає перевіряти правильність роботи програмного коду, що є необхідним для забезпечення високої якості та надійності функціональності продукту.

ОПИС ЗАСТОСУНКУ

Призначення застосунку. Мобільний застосунок "WelcomeHome" призначений для спрощення процесу адаптації військових до нормального життя після повернення з війни. Застосунок являє собою інформаційний портал, що стосується таких аспектів:

1.Психологічна підтримка. В додатку є розділ "Психологічної підтримки " із списком наявних психологічних служб, групами підтримки, а також списком благодійних заходів.

2.Реабілітація. Додаток має надавати можливість переглядати контактну інформацію та розташування на карті реабілітаційних центрів.

3.Соціальні виплати. Це стосується надання чіткої, зрозумілої покрокової інструкції щодо оформлення соціальних виплат та пільг для військових, а в майбутньому - створення ефективного інструменту для подання необхідних документів через додаток.

4.Підтримка. Створення платформи для організації зустрічей та подій, спрямованих на позитивну соціальну взаємодію військових між собою.

5.Навчання і вакансії. Наразі у вебзастосунку є можливість публікації та перегляду вакансій. У майбутньому - представлення інформації про безкоштовне навчання для військових з метою надання можливості освоїти нову професію.

Цілі створення застосунку. Застосунок було створено з метою досягнення низки важливих цілей, серед яких:

1.Розробити додаток з інтуїтивно зрозумілим інтерфейсом та привабливим дизайном, який буде комфортним для використання.

2.Надати чітку та легку покрокову інструкцію щодо оформлення соціальних виплат.

3.Створення централізованої інформаційної платформи: надання можливості отримувати всю необхідну інформацію в одному місці.

4.Проінформувати військових про можливості отримання різноманітної допомоги, що доступні для них не тільки в Україні, а і по всьому світу.

5.Максимальне спрощення процесу адаптації військових до цивільного життя після перебування на фронті.

6.Відзначити, що невід'ємний внесок військових у майбутнє країни є надзвичайно важливим, і що ми не лишаємо їх без уваги, а цінуємо та поважаємо, а наші зусилля спрямовані на те, щоб допомогти їм повернутися у суспільне життя.

На основі визначених цілей були розроблені діаграми використання Use-case (рис. 1-3), які ілюструють ключові взаємодії з різними суб'єктами та системою в цілому. Ці діаграми допомагають визначити та візуалізувати важливі аспекти функціональності системи та служать основою для подальшого проєктування та розробки програмного застосунку.

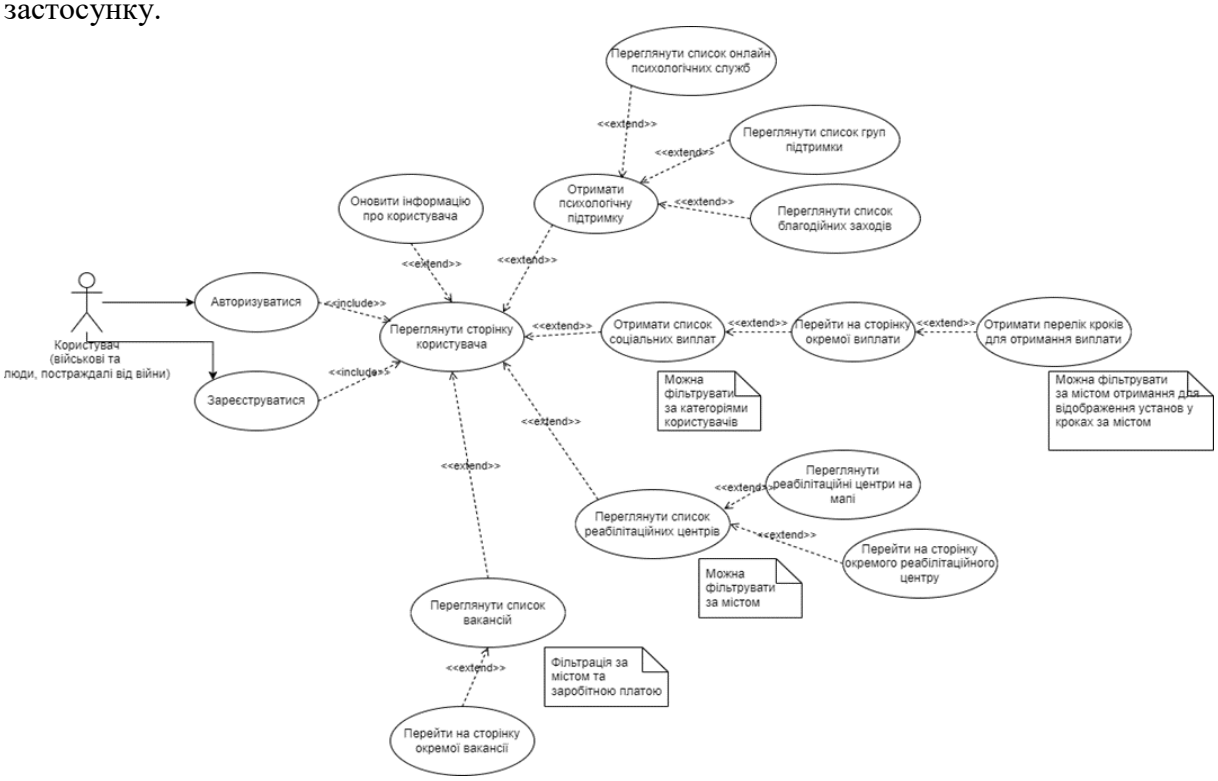


Рисунок 1 – Діаграма взаємодії для користувача

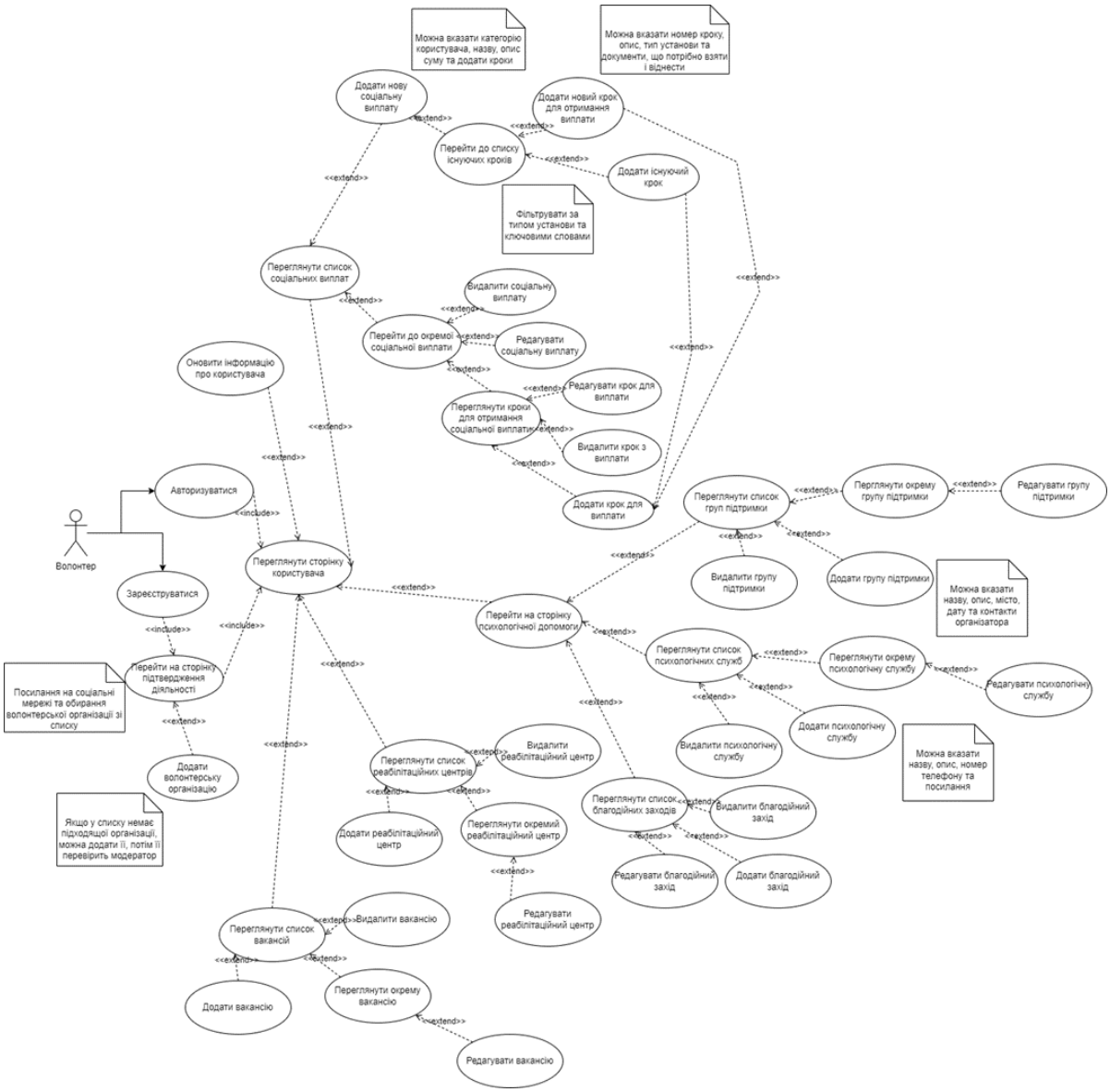


Рисунок 2 – Діаграма взаємодії для волонтера

Вимоги до застосунку

Вимоги в цілому

Мобільний застосунок має надавати можливість користувачам зареєструватися, заповнивши форму необхідними обліковими даними, а також виконувати автентифікацію та авторизацію на основі ролей. Також мають бути створені такі сторінки, як:

1.Сторінка реєстрації та авторизації;

2.Сторінка для перегляду списку психологічних служб, груп підтримки та благодійних заходів.

3.Сторінка для перегляду списку реабілітаційних центрів з окремою сторінкою їхнього розташування на карті, а також наявна можливість пошуку за фільтром.

4.Сторінка для перегляду списку доступних соціальних виплат з можливостю фільтрувати за категорією користувачів. При натисканні на соціальну виплату має з’явитися покрокова інструкція для її оформлення.

5.Сторінка для перегляду доступних вакансій.

Вимоги до структури та функціонування застосунку

В системі передбачається виділити наступні функціональні підсистеми:

- підсистема користувача, призначена для потенційний користувачів (військові та постраждалі від війни);
- підсистема волонтера, призначена для користування волонтерами;
- адміністративна, призначена для використання модератором;

До застосунку висувалися наступні функціональні вимоги:

- Застосунок має забезпечувати автентифікацію користувачів в системі з використанням електронної пошти та пароля.
- Застосунок повинен забезпечувати авторизацію з використанням двох ролей (користувач, волонтер) та надавати доступ до відповідних ресурсів.
- Для реєстрації нового користувача необхідно ввести повне ім’я, номер телефону, електронну пошту та пароль.
- Для реєстрації волонтера необхідно ввести додаткову інформацію для підтвердження діяльності, а саме посилання на соціальні мережі та вибір зі списку волонтерських організацій. Якщо волонтерської організації немає, волонтер має право створити нову волонтерську організацію.
- Після реєстрації волонтер має пройти верифікацію модератором, який перевірить всі введені дані.

Також висувалися наступні вимоги до інтерфейсу:

- Всі функції та розділи додатку повинні бути легко доступні та зрозумілі для користувача.
- Інтерфейс повинен мати привабливий та сучасний дизайн, що стимулює користувача до використання додатку.
- Для розділів, що містять інформацію про реабілітаційні центри, соціальні виплати чи навчальні ресурси, повинна бути реалізована можливість швидкого пошуку та фільтрації за ключовими параметрами.

В системі передбачається виділити наступні функціональні підсистеми:

-підсистема користувача, призначена для потенційних користувачів (військові та постраждалі від війни);

-підсистема волонтера, призначена для користування волонтерами.

Підсистема користувача

У таблиці 2 подано перелік функцій та задач, що підлягають автоматизації для підсистеми користувача.

Таблиця 2. Вимоги до підсистеми користувача

Функція	**Задача**
Реєстрація	Введення ім’я та прізвища.
	Введення електронної пошти.
	Введення номеру телефону.
	Введення паролю.
Робота з додатком	Переглянути сторінку користувача
	Оновити інформацію користувача
	Виключно перегляд доступної інформації (списки психологічних служб, груп підтримки, благодійних заходів, соціальних виплат, реабілітаційних центрів, вакансій).

Підсистема волонтера

Перелік функцій та задач, що підлягають автоматизації для підсистеми волонтера зображено у таблиці 3.

Таблиця 3. Вимоги до підсистеми волонтера

Функція	**Задача**
Реєстрація	Введення ім’я та прізвища.
	Введення електронної пошти.
	Введення номеру телефону.
	Введення інформації про соціальні мережі.
	Вибір назви волонтерської організації із випадного списку.
	Створення нової волонтерської організації.
	Введення паролю.
Робота з додатком	Перегляд інформації.
	Переглянути сторінку користувача.
	Оновити сторінку користувача.
	Додавання, редагування, видалення психологічних служб.
	Додавання, редагування, видалення благодійних зустрічей.
	Додавання, редагування, видалення груп підтримки.
	Додавання, редагування, видалення реабілітаційних центрів.
	Додавання, редагування видалення соціальних виплат та кроків для їх отримання.
	Додавання, редагування, видалення вакансій.

Опис організації інформаційної бази

Для роботи з проєктом було обрано SQL Server 19. Результуюча діаграма БД (рис. 4, 5) виглядає наступним чином:

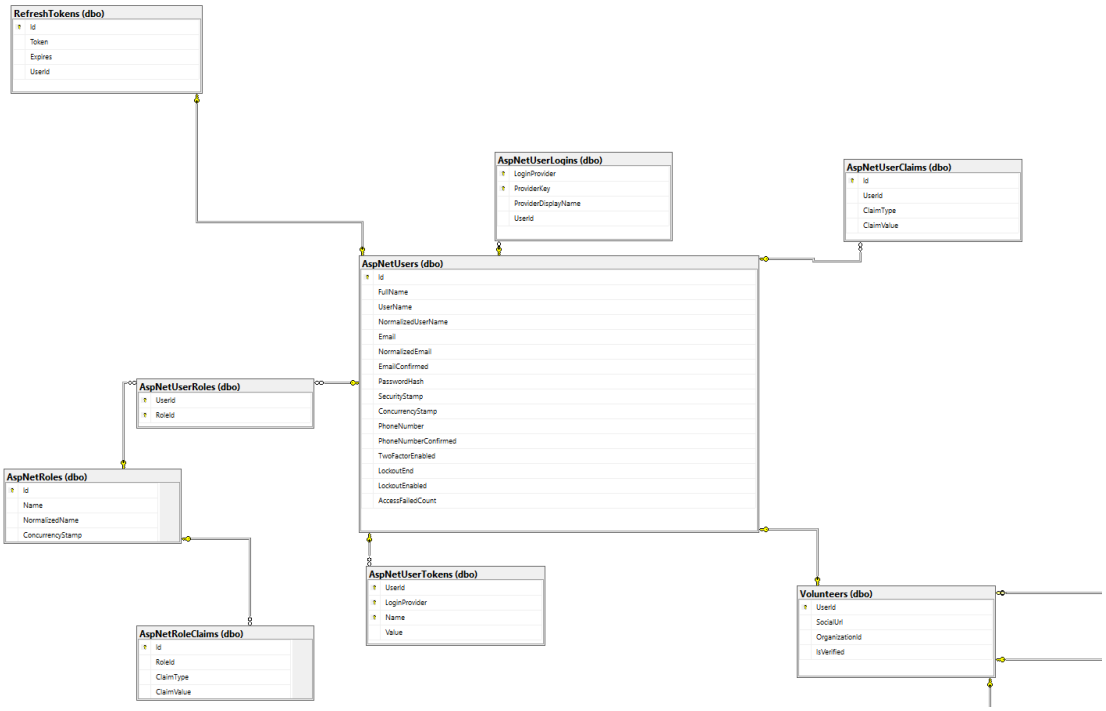


Рисунок 3 – Діаграма бази даних частина 1

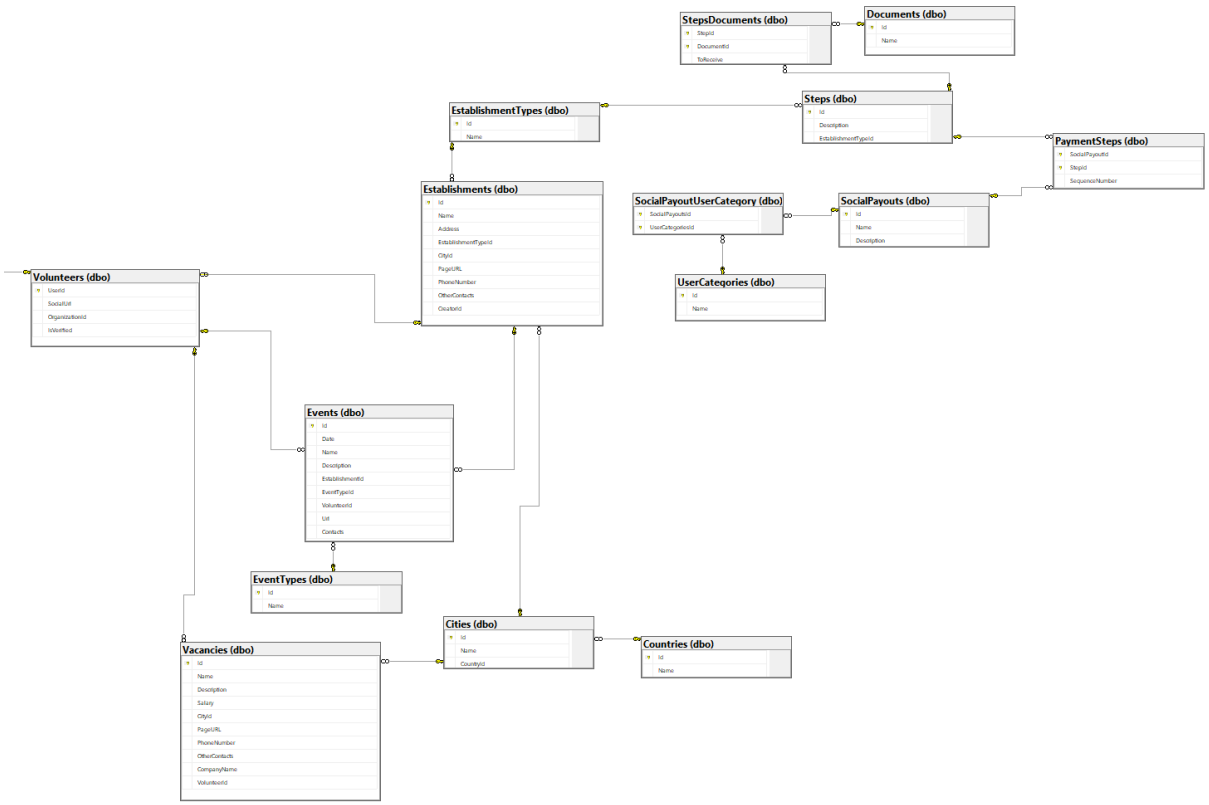


Рисунок 4 – Діаграма бази даних частина 2

Таблиця 4. Перелік таблиць бази даних

Номер	**Таблиця**	**Опис**
1	RefreshTokens	Таблиця токенів оновлення для авторизації
2	Volunteers	Таблиця волонтерів
3	Countries	Таблиця країн
4	Cities	Таблиця міст
5	Establishments	Таблиця установ
6	EstablishmentTypes	Таблиця типів установ
7	Events	Таблиця заходів
8	EventTypes	Таблиця типів заходів
9	Vacancies	Таблиця вакансій
10	SocialPayouts	Таблиця соціальних виплат
11	UserCategories	Таблиця категорій користувачів
12	SocialPayoutUserCategory	Проміжна таблиця виплати/категорії
13	Steps	Таблиця кроків для отримання соціальних виплат
14	PaymentSteps	Проміжна таблиця виплати/кроки
15	Documents	Таблиця документів
16	StepsDocuments	Проміжна таблиця кроки/документи

Опис таблиць

Таблиці AspNetRoles, AspNetUsers, AspNetUserRoles, AspNetRoleClaims, AspNetUserTokens, AspNetUserClaims та AspNetUserLogins були створені автоматично

технологією ASP.NET Core Identity, яка використовувалась для реалізації функціоналу авторизації та автентифікації (таблиці 5-20).

Таблиця 5. Опис таблиці RefreshTokens

Атрибут	**Тип**	**Опис**
Id	bigint	Ідентифікатор
Token	nvarchar(max)	Токен
Expires	datetime2	Дата закінчення терміну дії
UserId	bigint	Ідентифікатор користувача

Таблиця 6. - Опис таблиці Volunteers

Атрибут	**Тип**	**Опис**
UserId	bigint	Ідентифікатор користувача
SocialUrl	nvarchar(max)	Посилання на сторінки соціальних мереж волонтера
OrganizationId	bigint	Ідентифікатор установи
IsVerified	bit	Чи верифікований волонтер

Таблиця 7. - Опис таблиці Countries

Атрибут	**Тип**	**Опис**
Id	bigint	Ідентифікатор
Name	nvarchar(max)	Назва країни

Таблиця 8. - Опис таблиці Cities

Атрибут	**Тип**	**Опис**
Id	bigint	Ідентифікатор
Name	nvarchar(max)	Назва міста
CountryId	bigint	Ідентифікатор країни

Таблиця 9. - Опис таблиці Establishments

Атрибут	**Тип**	**Опис**
Id	bigint	Ідентифікатор
Name	nvarchar(max)	Назва установи
Address	nvarchar(max)	Адреса
EstablishmentTypeId	bigint	Ідентифікатор типу установи
CityId	bigint	Ідентифікатор міста
PageURL	nvarchar(max)	Посилання на сторінку установи
PhoneNumber	nvarchar(max)	Номер телефону
OtherContacts	nvarchar(max)	Інші контакти
CreatorId	bigint	Ідентифікатор волонтера

Таблиця 10. - Опис таблиці EstablishmentTypes

Атрибут	**Тип**	**Опис**
Id	bigint	Ідентифікатор
Name	nvarchar(max)	Назва

Таблиця 11. - Опис таблиці Events

Атрибут	**Тип**	**Опис**
Id	bigint	Ідентифікатор
Date	datetime2	Дата заходу
Name	nvarchar(max)	Назва заходу
Description	nvarchar(max)	Опис заходу
EstablishmentId	bigint	Ідентифікатор установи
EventTypeId	bigint	Ідентифікатор типу заходу
VolunteerId	bigint	Ідентифікатор волонтера
Url	nvarchar(max)	Посилання на захід
Contacts	nvarchar(max)	Контакти

Таблиця 12. - Опис таблиці EventTypes

Атрибут	**Тип**	**Опис**
Id	bigint	Ідентифікатор
Name	nvarchar(max)	Назва

Таблиця 13. - Опис таблиці Vacancies

Атрибут	**Тип**	**Опис**
Id	bigint	Ідентифікатор
Name	nvarchar(max)	Назва вакансії
Description	nvarchar(max)	Опис вакансії
Salary	real	Сума зарплатні
CityId	bigint	Ідентифікатор міста
PageURL	nvarchar(max)	Посилання на сторінку вакансії
PhoneNumber	nvarchar(max)	Номер телефону
OtherContacts	nvarchar(max)	Інші контакти
CompanyName	nvarchar(max)	Назва компанії
VolunteerId	bigint	Ідентифікатор волонтера

Таблиця 14. - Опис таблиці SocialPayouts

Атрибут	**Тип**	**Опис**
Id	bigint	Ідентифікатор
Name	nvarchar(max)	Назва виплати
Description	nvarchar(max)	Опис виплати
Amount	float	Сума виплати

Таблиця 15. - Опис таблиці UserCategories

Атрибут	**Тип**	**Опис**
Id	bigint	Ідентифікатор
Name	nvarchar(max)	Назва категорії

Таблиця 16. - Опис таблиці SocialPayoutUserCategory

Атрибут	**Тип**	**Опис**
UserCategoriesId	bigint	Ідентифікатор категорії
SocialPayoutsId	bigint	Ідентифікатор виплати

Таблиця 17. - Опис таблиці Steps

Атрибут	**Тип**	**Опис**
Id	bigint	Ідентифікатор
Description	nvarchar(max)	Опис кроку
EstablishmentTypeId	bigint	Ідентифікатор типу установи

Таблиця 18. - Опис таблиці PaymentSteps

Атрибут	**Тип**	**Опис**
SocialPayoutId	bigint	Ідентифікатор виплати
StepId	bigint	Ідентифікатор кроку
SequenceNumber	bigint	Порядковий номер кроку

Таблиця 19. - Опис таблиці Documents

Атрибут	**Тип**	**Опис**
Id	bigint	Ідентифікатор
Name	nvarchar(max)	Назва документу

Таблиця 20. - Опис таблиці StepsDocuments

Атрибут	**Тип**	**Опис**
DocumenttId	bigint	Ідентифікатор документу
StepId	bigint	Ідентифікатор кроку
ToReceive	bit	Вказує отримати чи подати документи

UI/UX-дизайн продукту

Можна виділити наступні закони та принципи UX-дизайну, які були використані при його створенні:

1.*Закон Фіттса (Fitts's Law)*. Цей закон стверджує, що час, необхідний для переміщення до цілі (наприклад, кнопки або іншого інтерактивного елемента), є функцією відстані до цілі та розміру цілі. У дизайні важливі інтерактивні елементи, наприклад, кнопки "Зареєструватись" та "Додати інформацію", мають великий розмір та поміщені у зручних для досягнення місцях, знижуючи зусилля користувача для взаємодії.

2.*Принцип ясності (Clarity)*: Ясність є одним із основних принципів UX-дизайну, оскільки допомагає користувачам розуміти, як користуватися продуктом. На сторінках чітко виділені розділи та кнопки, текст читабельний, а інтерфейс не перевантажений, що дозволяє користувачу легко орієнтуватись.

3.*Закон проксимітету (Proximity)*: Елементи, що мають відношення один до одного, розміщені поруч, що сприяє логічному згрупуванню інформації. Наприклад,

поля для введення інформації у формі створення кроку групуються разом, допомагаючи користувачу сприймати їх як єдине ціле.

4.*Принцип доступності (Accessibility):* Цей принцип забезпечує, що продукт може бути використаний якомога більшою кількістю людей. Великі кнопки та чіткий текст на скріншотах роблять інтерфейс доступнішим для осіб з різними здібностями.

5.*Принцип консистентності (Consistency):* Цей принцип допомагає користувачам швидше вчитися та ефективніше користуватися продуктом. Всі сторінки мають єдиний стиль інтерфейсу, який дозволяє користувачам інтуїтивно розуміти, як взаємодіяти з різними частинами додатку.

6.*Закон навантаження (Cognitive Load)*: Додаток не перевантажує користувача зайвою інформацією, тим самим знижуючи когнітивне навантаження. Інформація подається порціями, що робить процес взаємодії менш стресовим.

Ці закони та принципи впливають на поведінку користувача, забезпечуючи позитивний досвід взаємодії з продуктом, зменшуючи фрустрацію та збільшуючи задоволення від використання додатку. Чим краще продукт відповідає цим принципам, тим більша ймовірність того, що користувачі будуть його часто використовувати та рекомендувати іншим.

ІМПЛЕМЕНТАЦІЯ ПРОДУКТУ

Розробка Backend частини

Логіка застосунку написана на мові C# з використанням фреймворку ASP.NET Core 6. Було реалізовано трирівневу архітектуру та розділено програму на дві бібліотеки класів і один проєкт ASP.NET Core 6.

Структура проєкту

WelcomeHome.DAL - це бібліотека класів, яка відповідає за роботу з базою даних - MSSQL. У цьому проєкті було використано Entity Framework Core та впроваджено шаблони проєктування: Репозиторій та Unit of Work. Бібліотека класів містить моделі для створення таблиць, репозиторії, Unit of Work та DbContext.

WelcomeHome.Services - це бібліотека класів, яка є прошарком бізнес-логіки. Вона використовує бібліотеку класів WelcomeHome.DAL та містить DTO класи для представлення інформації, отриманої з бази даних або від користувача. Для співставлення DTO класів до моделей і навпаки використано бібліотеку AutoMapper. Основними у даній бібліотеці класів є сервіси. У них описана вся бізнес-логіка застосунку.

WelcomeHome.Web - проєкт ASP.NET Core 6. Зберігає в собі контролери, що відповідають за обробку HTTP-запитів, які надходять до сервера від користувача, класи для обробки кожного HTTP-запита (Middleware) та файл appsettings.json, в якому зберігається вся інформація потрібна для роботи застосунку.

Процес автентифікації та авторизації

У даному проєкті для імплементації автентифікації, авторизації та ідентифікації користувача були застосовані такі інструменти: ASP.NET Core Identity та JSON Web Token (JWT). Використання UserManager спрощує процес реєстрації, забезпечуючи безпеку зберігання конфіденційної інформації, зокрема, за допомогою хешування паролів.

ASP.NET Core Identity, який є частиною відкритої вебструктури для .NET, надає зручний фреймворк для управління користувачами, їхньою аутентифікацією та авторизацією [3].

JSON Web Token (JWT) виступає як механізм обміну авторизаційною інформацією між сторонами у вигляді токенів. Використання JWT для генерації Access Token та Refresh Token в реалізації автентифікації дозволяє ефективно підтверджувати особистість користувача та управляти його доступом до ресурсів системи.

При авторизації система отримує токен автентифікованого користувача та на основі інформації, отриманої з JWT токена (ролей та claim), визначає чи відповідає дана інформація визначеним в цій системі правилам доступу до даного ресурсу (policy).

Обробка виняткових ситуацій

Для забезпечення більш змістовних повідомлень про помилки та їх обробки у більш стандартизований спосіб було використано бібліотеку EntityFramework.Exceptions.Common. У бібліотеці визначено набір винятків, які можуть використовуватись для обробки загальних помилок, які виникають при роботі з Entity Framework (прошарок роботи з даними).

Для обробки виключень отриманих з рівня роботи з даними та їх обгортки в більш інформативні виключення, на рівні бізнес логіки було створено нові типи виключень: BusinessException та RecordNotFoundException. Відловлювання та створення нових виключень реалізовані на основі використання патернів проєктування "Медіатор" та "Ланцюг відповідальностей". Відловлювання відповідного типу виключення отриманого з бази даних - відповідальність класів, що наслідуються від абстрактного класу DataExceptionHandler (вони і будуть створювати "Ланцюг відповідальностей"). Клас ExceptionHandlerMediator налаштовує та запускає ланцюг.

На рівні репрезентації (проєкт ASP.NET Core Web API) для обробки виключень відповідних типів до інуючого в ASP.NET CORE переліку методів обробки HTTP-запитів (middleware) було створено клас, який містить у собі метод, що при обробці кожного надісланого HTTP-запиту, відловлює виключення рівня бізнес логіки та повертає HTTP-відповідь з відповідним кодом помилки.

Розробка Frontend частини

Застосунок був розроблений з використанням мови програмування JavaScript та фреймворку Expo для мобільних додатків на React Native. Основними інструментами є webpack для збірки проєкту, Babel для транспіляції коду, Node.js версії 16 для високої продуктивності та сумісності, а також TypeScript 4.8 для строгих типів.

У проєкті використовуються різноманітні бібліотеки, такі як Jest для тестування, Axios для HTTP-запитів, Formik для форм, react-native-keychain для безпечного зберігання даних, styled-components для стилізації та eslint для контролю якості коду.

Структура коду організована за парадигмою React, розділена на компоненти, екрани, навігацію, сервіси, зберігання стану, теми стилів та утиліти для чіткої відповідальності та полегшення розвитку.

Архітектура проєкту

Верхній рівень структури включає директорію screens, яка містить компоненти екранів користувацького інтерфейсу. Кожен підкаталог (Account, Help, Payments, Work, auth, Centers) в межах screens представляє функціональний модуль в застосунку. Наприклад:

- Account містить компоненти та логіку, пов’язані з управлінням користувацького облікового запису.
- Help містить інформацію допомоги чи підтримки для користувачів.
- Payments відповідає за обробку та вивід інформації про соціальні виплати.
- Work містить компоненти, пов’язані з вакансіями для користувачів.

- Auth включає екрани та логіку автентифікації користувачів.
- Centers пов’язаний з функціональністю закладів та установ, інформація про які також виводиться у застосунку.

Директорія services містить служби, що обробляють бізнес-логіку та забезпечують інтеграцію з зовнішніми API. Файли api-client.ts та auth.ts містять відповідно логіку для HTTP-запитів до вебсервісів та обробку аутентифікації користувачів.

Взаємодія із сервером

Взаємодія з серверною частиною реалізована через специфічний програмний прошарок, зазвичай позначений як 'service layer'. Цей рівень відповідає за організацію та виконання HTTP-запитів до сервера, таких як GET, POST, PUT та DELETE. Він містить набір шаблонів для цих запитів, а також визначає основний об'єкт ендпоїнтів (endpoints), який містить в собі усі необхідні адреси сервера для звернення.

Крім того, в цьому прошарку присутній middleware, який відіграє ключову роль у механізмі обробки відповідей сервера. Особлива увага приділяється відловлюванню статусу 401, що є критично важливим для ефективної роботи з системами, заснованими на JSON Web Tokens (JWT). Таке рішення дозволяє ефективно управляти сесіями користувачів та забезпечувати безпеку даних.

У процесі роботи вебзастосунку, на відповідні моменти часу всередині компонентів інтерфейсу, ініціюються потрібні виклики до цього сервісного прошарку. Це дозволяє забезпечити належний рівень абстракції та модульності, що сприяє підвищенню ефективності розробки та зменшенню складності коду.

ТЕСТУВАННЯ

Методи та підходи до тестування

При розробці тестів використовується Покроковий (Step-by-Step) підхід. Цей підхід розбиває тестування на окремі кроки, де кожен крок перевіряє певну функціональність або частину системи. Це дозволяє більш детально та систематично тестувати окремі компоненти чи функції програми.Основною причиною вибору цього підходу є чітке розділення проєкту на функціональні модулі, такі як Репозиторії, Сервіси, Контроллери та скрипти для БД. Вони розглядаються як окремі фрагменти програми, які відповідають за певну частину додатку.

Із методів тестування була обрана стратегія "модифікованого низхідного тестування". Оскільки програма має чітку ієрархію взаємодії між рівнями - репозиторій -> сервіс -> контролер, це спрощує можливість протестувати всі складові системи, розпочинаючи з верхнього рівня. Спочатку тестується головний модуль, а потім модулі, які викликаються з нього, і так далі до найнижчих рівнів. Основним недоліком, з яким довелося зіткнутися, стала потреба у написанні складних заглушок для найнижчих модулів. Це приводить до використання принципу мокування об'єктів, що, хоча частково полегшує цю проблему, не вирішує її повністю.

Модульні тести

На початковому етапі модульного тестування для аналізу було обрано сервіси, а саме сервіси волонтерів та документів. Вони були вибрані, оскільки не залежали від інших моделей, тому їх було тестувати найлегше. Для сервісу волонтерів класу VolunteerServiceTests було написано 8 тестів:

- GetAsync_WhenVolunteerExists_ReturnsVolunteerOutDTO()
- GetAsync_WhenVolunteerDoesNotExist_ThrowsRecordNotFoundException():

- GetAll_ReturnsAllVolunteersMappedToVolunteerOutDTO()
- AddAsync_WhenVolunteerAdded_SuccessfulAddition()
- AddAsync_WhenExceptionThrown_ThrowsException()
- UpdateAsync_WhenVolunteerUpdated_CheckUpdateAndPropertyName()
- DeleteAsync_WhenVolunteerDeleted_DeleteMethodCalledOnceWitCorrectId()
- GetCount_ReturnsTotalVolunteerCount()

Для сервісу волонтерів класу DocumentServiceTests було написано 9 тестів:

- GetAsync_WhenDocumentExists_ReturnsDocumentOutDTO()
- GetAsync_WhenDocumentDoesNotExist_ThrowsRecordNotFoundException()
- GetAll_ReturnsAllDocumentsMappedToDocumentOutDTO()
- GetByStepNeeded_ReturnsDocumentsFilteredStepIdAndToReceiveFlagFalse()
- GetByStepReceived_ReturnDocumentsFilteredStepIdAndToReceiveFlagTrue()
- AddAsync_WhenDocumentAdded_AddMethodCallsOnceAndMatches()
- AddAsync_WhenExceptionThrown_ThrowsException()
- UpdateAsync_WhenDocumentUpdated_CheckUpdateAndPropertyName()
- DeleteAsync_WhenDocumentDeleted_DeleteCalledOnceWithCorrectId()

Інтеграційні тести

В рамках цього пункту було проведене тестування, яке перевіряє взаємодію двох основних систем в програмі: програмного забезпечення, яке розробляється, та бази даних, з якою воно взаємодіє. Такі тести переконуються, що обидві системи працюють правильно, коли взаємодіють між собою.

Були протестовані методи репозиторія країн, а саме класу CountryRepository. Всього для прикладу вдалося розробити 4 тестових методи, оскільки вони доволі складні у налаштуванні та у подальшій підтримці. Основні тестові методи класу CountryRepositoryTests:

- GetAll_ReturnsAllCountriesWithCities()
- AddAsync_AddsNewCountry()
- DeleteAsync_DeletesExistingCountry()
- UpdateAsync_UpdatesExistingCountry()

ДОКУМЕНТАЦІЯ ПРОЄКТУ

Обрана модель та інструменти ведення документації

Під час розробки документації була використана модель "перекидаємо через стіну". Основною причиною вибору цієї моделі є документування вже робочого продукту, оскільки проєкт на початковому етапі розробки та часто змінюється, немає сенсу "документувати 2 рази". Як наслідок документація по продукту буде завжди актуальною. Мінусом цього підходу, який було відчутно - є сповільнення розробки документації, коли розробляється ключовий функціонал. Даний підхід до документування вписується в роботу невеликих груп та стартапів.

Основним інструментом для ведення документації було обрано Confluence через його широкий набір інструментів, призначених для створення якісної та візуально зрозумілої документації. Ця платформа надає можливість легко створювати графіки, таблиці та презентації, що робить процес документування більш ефективним та доступним. Confluence також має можливість додавати розширення, які значно

розширюють його можливості. Один з яскравих прикладів таких розширень - "Content Formatting Macros", яке дозволяє створювати вкладки. Ця функція відсутня в стандартному наборі інструментів Confluence.

Ще одним ілюстративним прикладом є можливість відображення профілі користувачів у системі за допомогою тієї ж самої надбудови за допомогою функції "User Profile". Ця функція дозволяє користувачам швидко переглядати фотографію особи, основну інформацію про неї та починати взаємодію через просте натискання на їх профіль (рисунок 5).

Учасник	Роль учасника
 Юлія Соломаха	Керівник проєкту, бекенд, фронтенд розробник
 Alina Bedenko	Бекенд, продукт дизайнер
 Vladyslav Spotar v.spotar@knu.ua	Тестувальник, фахівець з документації
 Vladyslav Burlaka	Тестувальник, фахівець з документації
 Dmytro Aleksenko	Фронтенд розробник
 Julia Nedavnia	Модератор, бекенд розробник
 Nataliia Filimonchuk	Аналітик
 Світлана Гнатюк	Аналітик

Рисунок 5 – Таблиця учасників команди

ОПИС ПРОЦЕСУ РОЗРОБКИ

Опис команди та ролей в команді

Команда складається з 8 людей. Студенти ФКНК 4-го курсу: Спотар Владислав, Алексенко Дмитро, Беденко Аліна, Бурлака Владислав, Соломаха Юлія, Недавня Юлія; студенти географічного факультету: Гнатюк Світлана, Філімончук Наталія. Розподіл ролей відбувався з урахуванням власних побажань. В в команді були такі ролі: фронтенд розробник, бекенд розробник, дизайнер, скрам-майстер, керівник проєкту, тестувальник, фахівець із документації, а також аналітик. Ознайомитись із зонами відповідальності кожного члена команди можна за допомогою рисунку 5. Варто відмітити, що наведений вище розподіл на рис. 5 є більше формальним, оскільки в реальному процесі створення продукту зона відповідальності часто змінювалась і кожен мав змогу спробувати себе у різних ролях.

Опис обраної методології та інструментарію

Був обраний наступний інструментарій організації процесу розробки: Jira, Google Meet та Telegram. Для роботи над проєктом була використана методологія Scrum, яку було обрано з ряду причин:

- Прозорість: дозволяє всім членам команди бути в курсі того, що відбувається в проєкті та як просувається розробка. Scrum-дошка в Jira та ретроспективні зустрічі дозволяють відслідковувати прогрес та виявляти проблеми.

- Результативність: дозволяє зосередитись на розв'язанні найважливіших проблем та завдань та забезпечує їх ефективне виконання в рамках відведеного часу. Це дозволяє швидко реалізувати корисні функції та мінімізувати витрати.
- Контроль: дозволяє чітко визначити ролі та обов'язки кожного члена команди та забезпечити їх ефективну співпрацю. Також Scrum дозволяє відслідковувати прогрес розробки та вчасно виявляти проблеми та затримки.

Для забезпечення ефективної комунікації та співпраці в команді, на основі опитування членів команди, було вирішено проводити зустрічі в Google Meet три рази на тиждень та ретроспективи наприкінці кожного спринту. Ретроспектива допомагала оцінити результати попереднього спринту та виявити проблеми та покращення для майбутніх спринтів.

Для організації та відстеження процесу розробки була створена Scrum-дошка в сервісі Jira. Це дозволило відслідковувати стан різних завдань, оцінювати їх складність та контролювати терміни виконання.

Основна комунікація в команді проводилась через месенджер Telegram, що дозволяло швидко та зручно обговорювати питання та ділитись інформацією.

Обрана методологія Scrum та засоби, що були використані для її реалізації, дозволили забезпечити ефективну комунікацію в команді, організувати процес розробки та в рамках відведеного часу успішно реалізувати поставлені завдання.

Планування спринтів

На початку роботи над проєктом командою було вирішено, що щопонеділка буде проводитись зустріч, на якій будуть обговорюватись задачі на поточний спринт. Також було проведено зустріч для планування спринтів на поточний семестр, а результати планування були записані в файл і додані до Confluence, що дозволило всім членам команди бути в курсі стану робіт та прогресу розробки [5].

Для демонстрації планування спринтів було використано Scrum дошку, на якій видно список завдань, що було потрібно виконати на кожен спринт та яка частина роботи належить кожному члену команди. Також за допомогою Scrum дошки відслідковувався хід виконання завдань та оцінювався стан робіт на кінець спринту. Демонстрація планування спринтів допомагала всім членам команди розуміти, які завдання повинні бути виконані та який прогрес був досягнутий на даний момент (рис. 6).

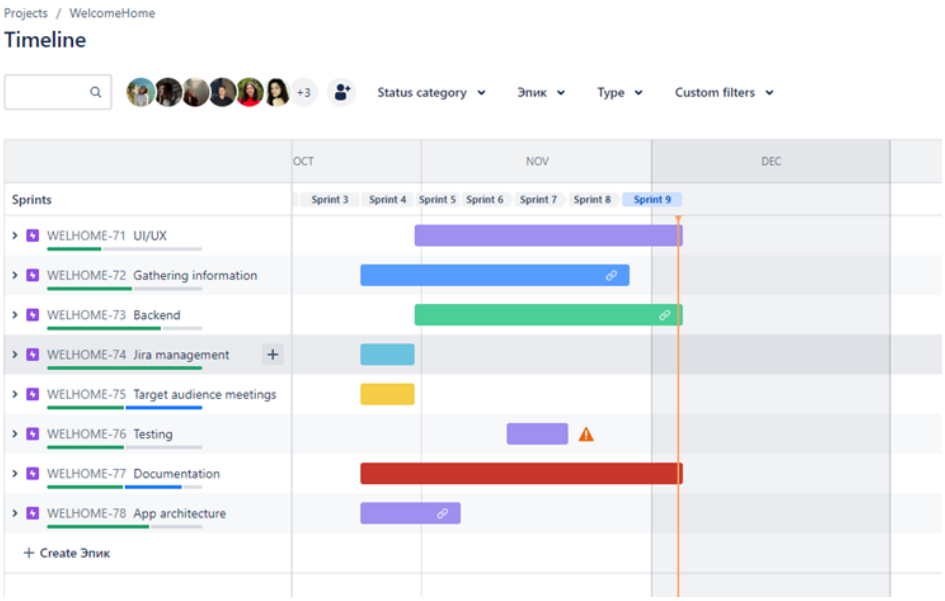


Рисунок 6 – Скрам-дошка

Для ефективного ведення процесу розробки програмного забезпечення потрібно було розподіляти велику задачу розробки додатку на менші і більш конкретні задачі, щоб розділити роботу між учасниками команди та більш точно відслідковувати прогрес роботи. У даному проєкті використовувалась така стратегія та для цього було використано механізм Еріс в Jira.

Еріс є потужним інструментом для організації масштабних проєктів та великих задач. У проєкті були означені основні складові програми, які були розкриті у вигляді епіків. Кожен з епіків описував конкретну частину програми.

Рефлексія

Рефлексія є важливим етапом для кожної команди, яка працює за Scrum методологією. Це час, коли кожен член команди має можливість проаналізувати, що вдалося і що не вдалося зробити, а також, як можна поліпшити роботу команди.

В командному проєкті ретроспективи проводилися наприкінці кожного спринту, що дозволяло команді аналізувати роботу проведену протягом поточного спринту та складати план для наступного. Були випадки, коли деякі задачі не були виконані протягом поточного спринту за рядом причин та переносилися на наступний. Однак, кожен раз проводився аналіз проблеми та знаходилися можливі варіанти її вирішення, за допомогою комунікації між членами команди та звернення по допомогу до спеціалістів.

Приклад використання ретроспективи під час розробки програмного продукту може бути проілюстрований наступним випадком. У рамках одного зі спринтів команді не вдалося реалізувати задачу щодо організації віддаленої бази даних, внаслідок відсутності безкоштовних ресурсів для хостингу. Однак, протягом однієї з ретроспектив було знайдено необхідні рішення шляхом розподілу завдання між учасниками команди, взаємодії та обміну інформацією, а також зверненню за підтримкою до викладачів та інших курсів кафедри. Отриманий досвід дозволив команді бути готовою до вирішення подібних проблем у майбутньому, а також знайти більш ефективні та продуктивні рішення.

Чому навчилися

Протягом роботи в командному проєкті команда навчилася безлічі корисних навичок та кращих практик.

- Керування проєктами: на практичному досвіді команда навчилася різним методам керування проєктом.
- Комунікація: кожен учасник команди був в курсі прогресу кожного з членів команди (за допомогою дошки в Jira), а при виникненні проблем кожен міг звернутися за допомогою до колег в чаті Telegram.
- Рефлексія та вдосконалення: Scrum методологія навчила команду проводити регулярну ретроспективу, під час якої команда знаходила переваги та недоліки своєї роботи та шукала способи поліпшення.
- Робота в команді: було покращено soft-skills. Кожен з членів команди відчував відповідальність за інших, докладав зусилля та знаходив компроміси.
- Управління часом: Scrum методологія навчила команду краще контролювати свій час та визначати пріоритети.

Під час співпраці в командному проєкті неодмінно виникали різноманітні проблеми та складнощі, однак саме завдяки їх наявності учасники отримували можливість виявляти недоліки та недосконалості у своїй роботі та шукали ефективні способи їх вирішення.

ВИСНОВКИ

В результаті, на даному етапі отримали продукт у вигляді мобільного застосунку під назвою "WelcomeHome". Ця назва більше говорить про те, що ми поважаємо і цінуємо військовослужбовців та їх внесок у майбутнє нашої держави, а наші зусилля спрямовані на те, щоб спростити військовим процес адаптації до цивільного життя після перебування на фронті.

Фактично, наші прямі бенефіціари зможуть знайти у цьому застосунку необхідну та корисну інформацію про можливості отримання психологічної підтримки, організації спільних зустрічей з однодумцями, перелік центрів для реабілітації після поранення, покрокові інструкції для отримання соціальних виплат, а також детальну інформацію про вакансії для працевлаштування поранених, або військових, які мають інвалідність тощо.

Маємо відзначити про те, що дана розробка не є повністю завершеною, та у майбутньому її буде доповнено та доопрацьовано. Ми впевнені, що цей продукт буде корисним у користуванні її кінцевими бенефіціарами, та допоможе або значно спростить повернення військовослужбовців додому.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Мобільний застосунок "База" [Електронний ресурс]. Вільний вибір. Режим доступу до ресурсу:
2. https://www.vvybir.org.ua/мобільний-застосунок-база/
3. ASP.NET Core Documentation [Електронний ресурс]. Microsoft. Режим доступу до ресурсу:
4. https://learn.microsoft.com/en-gb/aspnet/core/?view=aspnetcore-8.0
5. ASP.NET Core Identity Documentation [Електронний ресурс]. Microsoft. Режим доступу до ресурсу:
6. https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-8.0&tabs=visual-studio
7. Redux Documentation [Електронний ресурс]. Microsoft. Режим доступу до ресурсу: https://redux.js.org/
8. План розвитку продукту "Welcome Home" [Електронний ресурс]. Команда Stark. Режим доступу до ресурсу: https://welcomehomeua.atlassian.net/wiki/spaces/PD/pages/11239510/MVP

МОБІЛЬНИЙ ЗАСТОСУНОК ТОВАРІВ ПРОДУКТОВИХ МЕРЕЖ

Володимир Шафран, Аліна Д'ячек, Дмитро Макаров, Святослав Романків

Робота присвячена створенню мобільного застосунку для оптимізації процесу вибору та покупки продуктів харчування в сучасних умовах високої конкуренції та швидких темпів життя. Приділено увагу дослідженню існуючих рішень, аналізу їхніх переваг та недоліків, та розробці власного застосунку з урахуванням найкращих практик і сучасних технологій. В звіті описано методи розробки, використані інструменти, такі як Ruby on Rails та Flutter, та представлено аналіз ринку конкурентів. Важливу роль відіграє SWOT-аналіз, що допомагає виявити сильні та слабкі сторони продукту, його можливості та ризики. Робота також включає опис системних та програмних вимог, структури бази даних, процесів вебскрапінгу для збору даних, а також інтерфейсу та користувацького досвіду. Важливим аспектом є розгляд методології Agile і впливу Scrum на процес розробки. Ключова мета проєкту — створити надійний та ефективний інструмент, який забезпечує користувачам комфорт та ефективність при виборі продуктів.

This work focuses on developing a mobile application designed to streamline the process of selecting and purchasing food products amidst intense market competition and fast-paced lifestyles. Through a comprehensive analysis of existing solutions, the project highlights both their strengths and weaknesses, leading to the development of a custom application that incorporates best practices and cutting-edge technology. Using Ruby on Rails and Flutter, the team employed robust development methods and conducted a thorough market analysis including a SWOT analysis to gauge the product's viability, strengths, opportunities, risks, and weaknesses. Key objectives include creating a reliable and effective tool that enhances user experience in product selection by leveraging real-time data scraping for price and product information from various stores. This application aims to provide users with an integrated and intuitive interface that facilitates informed purchasing decisions, promoting efficiency and user comfort. The document details the system requirements, database structure, web scraping processes, user interface, and user experience enhancements. It also explores Agile methodology's role in the development process and the impact of Scrum practices on project management. Overall, the project delivers a sophisticated platform that responds to contemporary consumer needs, blending functionality with ease of use to improve everyday shopping experiences.

МОБІЛЬНИЙ ЗАСТОСУНОК, ОПТИМІЗАЦІЯ ПОКУПОК, RUBY ON RAILS, FLUTTER, SWOT-АНАЛІЗ, ВЕБСКРАПІНГ, ІНТЕРФЕЙС, ПРОДУКТИ

MOBILE APPLICATION, PURCHASE OPTIMIZATION, RUBY ON RAILS, FLUTHER, SWOT ANALYSIS, WEB SCRAPING, USER INTERFACE, PRODUCTS

ВСТУП

Актуальність роботи та підстави для її виконання. Споживачі в сучасному світі зіштовхуються з проблемою надмірного вибору та необхідністю оптимізації покупок. За останні роки, в умовах швидких темпів життя та змін у споживчих уподобаннях, виникла потреба в зручному інструменті для вибору продуктів харчування. Така необхідність обумовлена конкуренцією на ринку, активним розвитком технологій, а також ростом інтересу до здорового харчування. В цьому контексті розробка застосунку відповідає актуальним вимогам ринку.

Мета й завдання роботи. Метою роботи є створення мобільного застосунку, який надасть споживачам зручний та інтегрований інструмент для вибору продуктів харчування. Для досягнення цієї мети поставлено такі завдання:

– проаналізувати існуючі мобільні застосунки, визначити їхні переваги та недоліки;

– розробити інтуїтивно зрозумілий інтерфейс та функціонал, що охоплюватиме усі аспекти вибору продуктів;

– забезпечити можливість отримання актуальної інформації про ціни та асортимент з різних магазинів;

– набути практичних навичок у розробці, впровадженні мобільних застосунків і роботі в команді;

– поглибити знання у сфері UX/UI дизайну.

Об'єкт розробки. Об'єктом розробки є процес вибору та покупки продуктів харчування.

Методи розробки. Аналіз наявних на ринку продуктів, аналіз та розробка програмного продукту.

Інструменти розробки. Безкоштовний та вільно поширюваний текстовий ректор Visual Studio Code, мови програмування Dart та Ruby.

Огляд наявних конкуруючих на ринку IT-продуктів

SuperMarkets [1]

Зрозумілий інтерфейс, можна шукати акційні товари по категоріях, по магазинах, сортувати за ціною та брендами. Даний застосунок є найбільш привабливим серед всіх конкурентів. Спеціалізується виключно на продуктах харчування.

Переваги:

– можна на карті зазначити своє місцезнаходження для того, щоб були відображені всі магазини поряд;

– гарний варіант відображення цін для інтуїтивного сприйняття;

– відображена інформація про Nutri-Score (система маркування харчової цінності, яка наноситься за бажанням виробника на лицьову сторону упаковки товару), метою чого є допомогти покупцеві зрозуміти харчову цінність та користь продукту;

– графічне відображення історії змін ціни на товар.

Недоліки:

– незручно обирати своє місто, оскільки в списку випадають й міста, й області, а також не всі крупні міста доступні;

– можливо, накручені відгуки клієнтів про товар.

Costless [2]

Ключова перевага: можливість створення списків продуктів. Також у застосунок можна завантажити всі дисконтні картки та використовувати їх для покупок. Додаток надає бонуси за здійснені покупки, які можна обмінювати на сертифікати магазинів або поповнити власний мобільний рахунок.

Переваги:

– можна додавати свої дисконтні карти;

– створений список покупок можна переслати, наприклад, в Телеграм;

– наявні готові шаблони продуктів: для діабетиків, при гастриті, для сніданку, продукти в дорогу тощо.

Недоліки:

– немає зручної розбивки товарів на категорії;

– відображаються абсолютно всі знижки в супермаркетах, проте в такому випадку викидає купу дрібних товарів (наприклад, канцелярія), які користувачу потрібно довго листати;

– сортувати товари можна лише за ціною.

Love Sales [3]

В застосунку можна переглядати каталоги акційних товарів в супермаркетах, додавати в обрані цікаві для себе та слідкувати за датами дії акційних пропозицій.

Переваги:

– пошук знижок по окремих магазинах чи торгових мережах;

– можна налаштувати застосунок: приховати нецікаві для себе категорії товарів чи магазини;

Недоліки:

– відображені лише каталоги товарів, неможливо знайти конкретний товар.

Сильні і слабкі сторони розроблюваного продукту

SWOT-аналіз обраної моделі:

– сильні сторони: гнучкість ідеї, велика цільова аудиторія (тому продукт буде мати попит), мала конкуренція;

– можливості: розширення функціоналу, залучення рекламодавців та сторонніх сервісів (пр. Glovo);

– слабкі сторони: необхідність регулярної актуалізації, недостатня кількість учасників команди, відсутність інвестицій;

– ризики: зміни на зовнішніх сайтах магазинів, правові аспекти на доступ.

Мапа продукту

На рисунку 1 зображено мапу продукту, де висвітлено концепцію продукту по основним темам.

Рисунок 1 – Мапа продукту

Огляд використаних технологій

Ruby [15] — це динамічна, об'єктно-орієнтована мова програмування, яка була розроблена в Японії в 1990-х роках Юкихіро Мацумото (Мац). Вона володіє простим і зрозумілим синтаксисом, який забезпечує легку читабельність коду. Ruby визначається

своєю гнучкістю, експресивністю та здатністю до підтримки багатьох парадигм програмування, включаючи процедурне, функціональне та метапрограмування [4].

Ось деякі ключові риси мови програмування Ruby:

- синтаксис;
- об'єктно-орієнтованість;
- динамічна типізація;
- збирання сміття;
- метапрограмування;
- велика стандартна бібліотека.

Ruby застосовується для розробки різноманітних видів програмного забезпечення, від вебзастосунків до настільних програм і скриптів. Вона є популярним вибором для веброзробки завдяки фреймворкам. Також Ruby знаходить застосування в автоматизації завдань, науковому моделюванні, обробці текстів і багатьох інших областях.

Homebrew був розроблений на Ruby і використовує Ruby для управління пакетами та збереження логіки програми. Користувачі можуть використовувати команди Homebrew, написані на Ruby, щоб швидко встановлювати та оновлювати пакети з великого репозиторію програм, який підтримується Homebrew [5].

Homebrew є прикладом потужного і популярного проєкту, де Ruby використовується для розробки важливих інструментів і допомагає спростити життя розробників, що працюють на платформі macOS.

Ruby on Rails, або просто Rails, є високорівневим вебфреймворком, розробленим мовою програмування Ruby. Він створений з метою забезпечити зручність розробки вебзастосунків шляхом надання конвенцій та стандартів, які дозволяють програмістам писати менше коду, зменшувати повторюваність та прискорювати процес розробки [6].

Rails використовує концепцію Model-View-Controller (MVC), що дозволяє розділити логіку програми, що пов'язана з базою даних (модель), логіку відображення і взаємодії з користувачем (представлення) та логіку обробки запитів та координації дій (контролер). Цей підхід сприяє структурованому та організованому розробці застосунків. Також є популярним вибором для розробки бекенд частини як API.

Rails має багато вбудованих функціональних можливостей, що полегшують розробку. Основні риси фреймворку включають:

- конвенція над конфігурацією;
- ActiveRecord;
- RESTful маршрутизація;
- автоматизоване тестування;
- велика екосистема.

Ruby on Rails широко використовується для швидкої розробки вебзастосунків, включаючи соціальні мережі, електронну комерцію, платформи з керування вмістом та багато іншого. Його популярність зумовлена швидкістю розробки, читабельністю коду та зручними інструментами, що він надає [7].

Dart - це сучасна, об'єктно-орієнтована мова програмування, яка була розроблена компанією Google. Вона створена для розробки високоефективних, масштабованих та переносних додатків. Основна сфера застосування Dart - це розробка клієнтських додатків, зокрема мобільних та вебзастосунків [8].

Основні риси мови Dart:

- сучасність і ефективність;
- об'єктно-орієнтованість;
- статична типізація;
- переносність;

– мережева безпека;
– платформа Flutter;
– зручна розробка інтерфейсу користувача.

Flutter - це відкритий крос-платформний фреймворк для створення мобільних, веб та десктоп-застосунків з використанням мови програмування Dart. Розроблений компанією Google, Flutter надає можливість швидкої розробки та створення красивого та високопродуктивного інтерфейсу користувача. Основні особливості Flutter включають [9]:

– гаряча перезавантаження (Hot Reload);
– відмінна візуалізація;
– компоненти (Widgets);
– крос-платформенність;
– доступ до платформенних сервісів;
– широка спільнота та підтримка;
– висока продуктивність.

Flutter набуває популярності в розробці мобільних та вебзастосунків завдяки своїм перевагам у швидкості розробки, гарному вигляду та мультиплатформності.

ОПИС ПРОДУКТУ

Призначення продукту

Застосунок створюється з метою забезпечення ефективного збору, обробки та аналізу інформації, пов'язаної з процесом вибору та покупки продуктів. Основні аспекти призначення системи включають:

1. Застосунок автоматизовано збирає різноманітну інформацію, пов'язану із споживачами та продуктами. Це включає дані про ціни, акції, властивості товарів, а також персональні вподобання та історію користувачів.
2. Застосунок забезпечує автоматизовану обробку зібраної інформації. Це включає сортування продуктів за різними параметрами, фільтрацію за вказаними у користувача критеріями та агрегацію даних для отримання загальних статистичних показників.
3. Однією з можливих функцій застосунку є можливість глибокого аналізу даних. Вона допомагає користувачам виявляти та розуміти патерни в їхньому споживацькому поведінці, а також надає рекомендації щодо оптимальних виборів продуктів.

Таким чином, застосунок розроблено для полегшення процесу вибору та покупки продуктів, надаючи користувачам зручний та інтуїтивно зрозумілий інструмент для взаємодії з інформацією та прийняття обґрунтованих рішень у сфері шопінгу.

Основні події в системі з боку користувача позначені на рис. 2.

Цілі створення продукту:

– забезпечити швидкий та ефективний збір та обробку інформації;
– забезпечити широкий функціонал для аналізу різних аспектів;
– створити інтуїтивно зрозумілий і зручний інтерфейс для користувачів;
– забезпечити стабільну та надійну роботу системи в усіх умовах;
– забезпечити можливість розширення функціоналу та роботи з більшим обсягом даних.

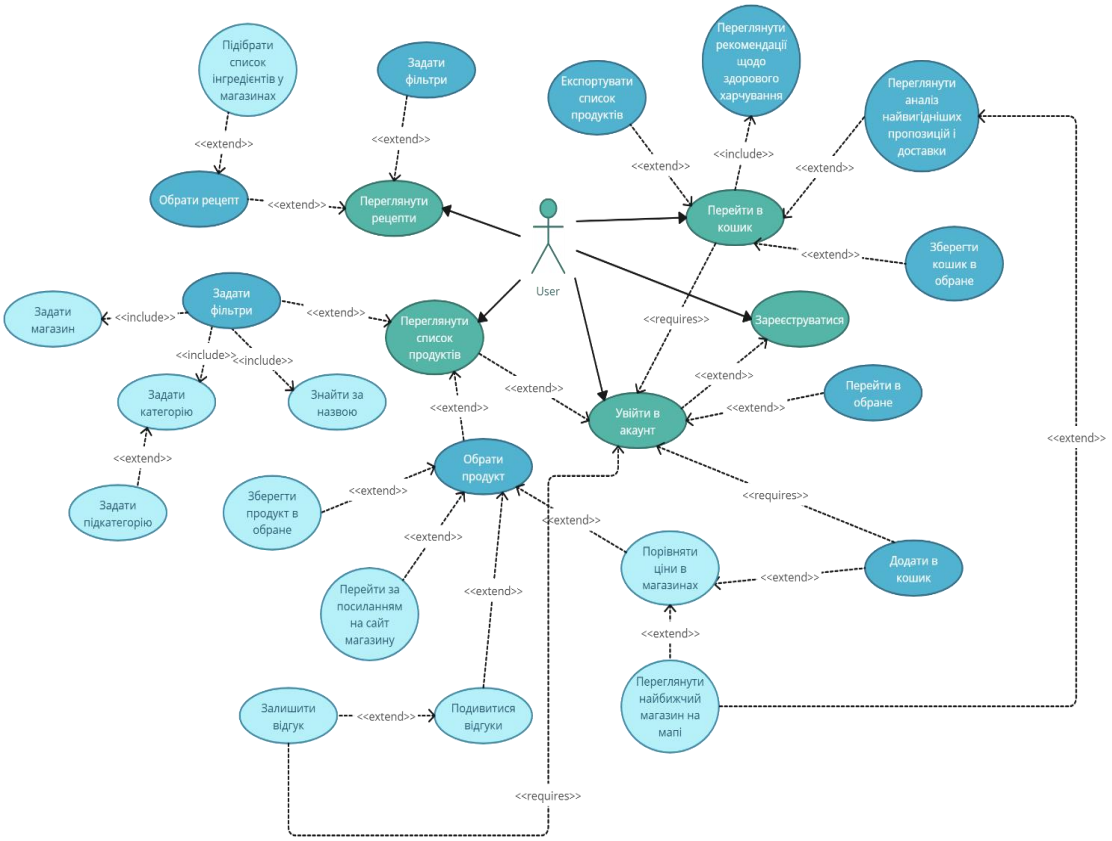


Рисунок 2 – Use-case діаграма

Вимоги до продукту

В продукті передбачаються наступні функціональні вимоги:

– можливість реєстрації та авторизації користувачів;

– створення та збереження персональних профілів користувачів;

– можливість здійснення пошуку продуктів за різними параметрами (ціна, бренд, акції тощо);

– фільтрація результатів пошуку;

– відображення детальної інформації про продукти, включаючи ціни, акції, характеристики;

– перегляд рейтингів та відгуків інших користувачів;

– надання персоналізованих рекомендацій на основі історії покупок та вподобань.

Нефункціональні вимоги:

– інтуїтивно зрозумілий та привабливий інтерфейс для забезпечення зручного використання користувачами;

– мінімальні часові затримки при виконанні запитів та завантаженні інформації;

– захист персональних даних користувачів;

– механізм авторизації;

– здатність системи ефективно працювати при збільшенні обсягу даних та кількості користувачів;

– забезпечення доступності сервісу для користувачів у будь-який час;

– сумісність системи з різними пристроями та операційними системами;

– наявність документації для адміністраторів та користувачів.

Вимоги до структури та функціонування продукту

Для крос-платформенного мобільного застосунку, що використовує Flutter для фронтенду та Ruby on Rails для бекенду, ось деякі загальні вимоги:

Функціональні вимоги:

1. **Формування списку покупок:** додавання/видалення продуктів у список покупок і можливість вказати кількість та інші параметри продуктів.
2. **Пошук та фільтрація:** пошук продуктів за назвою, категорією тощо та фільтрація за різними параметрами (ціна, бренд, категорія).
3. **Інтеграція з бекендом:** взаємодія з бекендом через API для отримання даних про продукти та авторизація та безпека взаємодії між мобільним застосунком та сервером.
4. **Перенаправлення на сайт супермаркету: м**ожливість перенаправлення користувача на сайт супермаркету для покупки обраних товарів і забезпечення безпеки та відслідковування перенаправлень.
5. **Відображення за категоріями:** забезпечте можливість перегляду продуктів за їхніми категоріями та можливість швидко фільтрувати та відображати тільки ті продукти, які відносяться до певної категорії.
6. **Можливість додавання нових категорій:** якщо є потреба в розширенні асортименту продуктів, можливість додавання нових категорій.

Нефункціональні вимоги:

1. **Вигляд та інтерфейс:** привабливий та зручний інтерфейс користувача, відповідний стандартам матеріального дизайну (Material Design) для Flutter, гарантія єдності дизайну на різних платформах.
2. **Продуктивність та Швидкодія:** оптимізація швидкодії та продуктивності мобільного застосунку за допомогою Flutter та ефективних запитів до бекенду.
3. **Масштабованість та Розширюваність:** забезпечення можливості масштабування мобільного застосунку для росту користувачів, легкість розширення функціональності за допомогою нових модулів чи власних плагінів.
4. **Ефективність бази даних:** забезпечте оптимізовану структуру бази даних для швидкого виконання запитів та отримання продуктів з різних категорій.
5. **Швидкість завантаження списку продуктів:** забезпечте швидке завантаження списку продуктів для користувачів, зокрема тих, які переглядають продукти певної категорії.
6. **Коректність синхронізації даних:** у випадку оновлення категорій або додавання нових, гарантувати коректну синхронізацію цих даних між мобільним застосунком та бекендом.
7. **Безпека:** захист персональної інформації користувачів та інших конфіденційних даних і використання безпечних протоколів комунікації між мобільним застосунком та бекендом.
8. **Тестування:** забезпечення високого рівня тестування, включаючи модульне тестування для Flutter та тестування API для Ruby on Rails.
9. **Крос-платформенність:** гарантія працездатності на різних мобільних платформах (Android та iOS).

Важливо також враховувати зміни та оновлення вимог протягом розвитку продукту.

Системні вимоги. Операційна система: Застосунок був створений для роботи на ОС Android 13.0 та iOS 17.0. Мінімальною версією ОС для роботи застосунка є Android 5.0 або iOS 11.0.

Інші: Для коректної роботи застосунок повинен мати доступ до мережі Інтернет.

Логічна структура бази даних

Використання PostgreSQL [14] в ролі системи управління базами даних (СУБД) визначалося його потужністю, надійністю та розширюваністю, що є ключовими аспектами при вирішенні завдань проєкту.

Застосування дозволило ефективно зберігати та управляти даними про продукти, списки покупок та інші важливі компоненти системи. Система гарантує надійність операцій з базою даних, а також підтримує високий рівень безпеки.

Однією з важливих переваг є його здатність до розширення, що сприяє масштабованості системи. Це дозволяє ефективно обробляти збільшення обсягу даних та витримувати велику кількість транзакцій.

Під час розробки API та взаємодії з PostgreSQL, було враховано особливості роботи з реляційними структурами даних, а також забезпечено оптимізацію запитів для підвищення продуктивності взаємодії між бекендом та базою даних (таб. 1-7).

Таблиця 1. Перелік таблиць БД

Номер	**Таблиця**	**Опис**
1	shops	Збереження назви магазинів
2	catalogs	Збереження назви каталогів
3	categories	Збереження необхідної кількості полів
4	products	Збереження інформації про продукти
5	endpoints	Збереження адреси точки входу для скраперів
6	images	Збереження картинок з поліморфним зв’язком

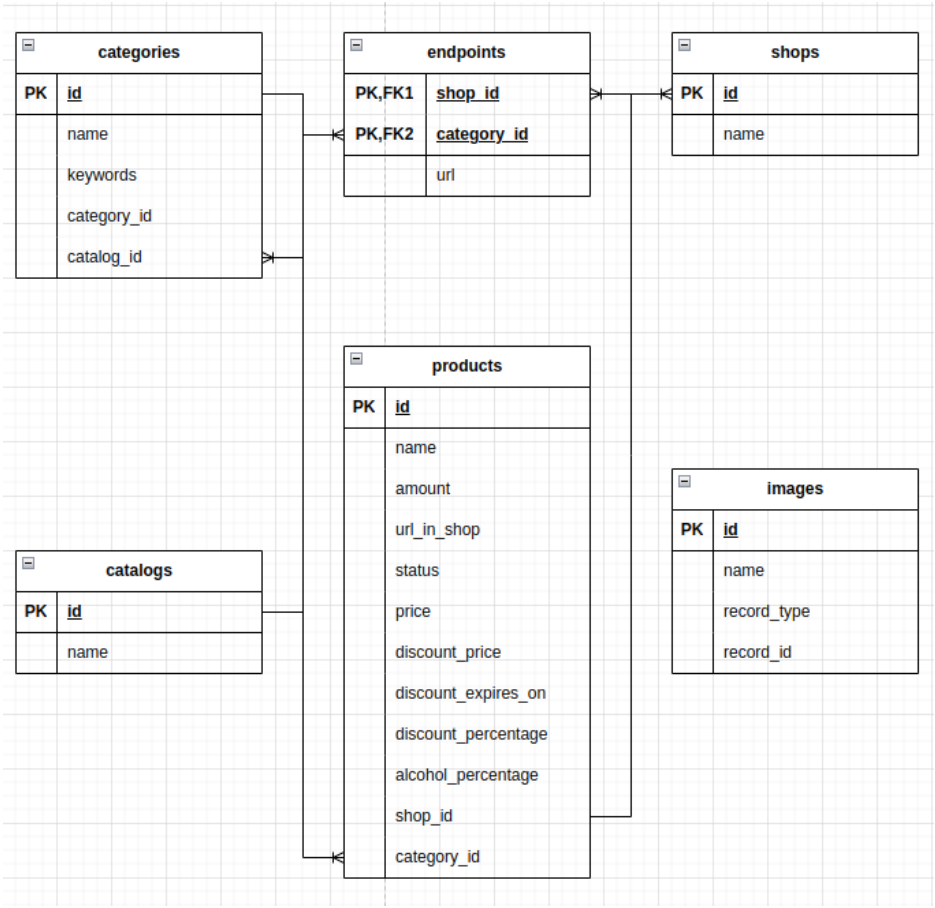


Рисунок 3 – Схема бази даних

Опис таблиць

Таблиця 2 — Таблиця shops

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор
name	varchar	Назва магазину

Таблиця 3 — Таблиця catalogs

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор
name	varchar	Назва каталогу

Таблиця 4 — Таблиця categories

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор
name	varchar	Назва категорії/підкатегорії
category_id	integer	Поле для опціональної підкатегорії
catalog_id	integer	Кожна категорії відноситься до каталогу
keywords	text	Ключові слова для опціональної категоризації

Таблиця 5 — Таблиця endpoints

Атрибут	**Тип**	**Опис**
shop_id	integer	Ідентифікатор магазину
category_id	integer	Ідентифікатор категорії
url	varchar	Адреса для точки входу скраперу

Таблиця 6 — Таблиця images

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор
name	varchar	Назва картинки
record_type	varchar	Тип об’єкта, для реалізації поліморфності
record_id	integer	Ідентифікатор об’єкта

Таблиця 7 — Таблиця products

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор
name	varchar	Назва продукту
amount	varchar	Вміст продукту
url_in_shop	varchar	Посилання на продукт в магазині
status	integer	Enum статус продукту
price	integer	Зберігаємо ціну в копійках для зручності
discount_price	integer	Ціна зі знижкою
discount_expires_on	date	Дата закінчення знижки
discount_percentage	integer	Відсоток знижки
alcohol_percentage	real	Кількість спирту
shop_id	integer	Ідентифікатор магазину
category_id	integer	Ідентифікатор категорії

UI/UX-дизайн продукту

У сучасному цифровому світі, розробка програмного забезпечення, мобільних застосунків, вебсайтів та інших цифрових продуктів є необхідною умовою для конкурентоспроможності бізнесу. У цьому контексті UX (взаємодія користувача) і UI (інтерфейс користувача) грають ключову роль у створенні продуктів, які задовольняють потреби та очікування користувачів [10].

Гарний UX/UI дозволяє створити приємний та зручний досвід користувача. Чіткість, зрозумілість та ефективність інтерфейсу дозволяють користувачам легко орієнтуватися та досягати своїх цілей. Задоволені користувачі частіше повертаються до продукту, рекомендують його своїм знайомим та залишають позитивні відгуки.

Зручний UX/UI спрощує взаємодію з продуктом і допомагає користувачам швидше досягати своїх цілей. Інтуїтивно зрозумілі інтерфейси зменшують кількість помилок, пов'язаних з неправильним використанням продукту, і допомагають зекономити час та зусилля. Ефективність взаємодії з продуктом сприяє підвищенню продуктивності користувачів і забезпечує їх задоволення.

UX/UI впливають на сприйняття продукту користувачами та можуть сприяти побудові позитивного бренду. Гарний дизайн створює довіру, збільшує лояльність та створює позитивні асоціації з брендом. Якщо користувачі задоволені взаємодією з продуктом, вони стають більш лояльними до бренду і його продуктів, що сприяє розвитку довгострокових відносин з клієнтами.

Розробка UI/UX дизайну застосунку застосунку відбувалася з використанням наступних принципів:

1. **Функціональний мінімалізм.**

Застосунок не повинен перевантажувати користувача зайвими діями, не потрібно відволікати користувача від головної мети.

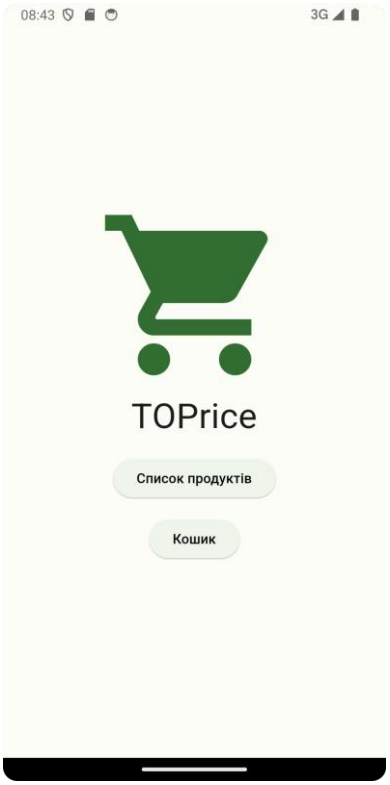


Рисунок 4 – Головна сторінка

2. **Чіткий порядок та ієрархія.**

У кожен момент часу використання застосунку користувачем на екрані повинен знаходитися лише найнеобхідніших контент. Навігація між екранами повинна бути чіткою та зрозумілою. Прикладом у застосунку є чітка та зрозуміла навігація між списком продуктів та конкретним продуктом.

3. **Використання жестів.**

Правильне використання жестів дозволяє покращити UX використання застосунку та зробити взаємодію з ним набагато більш зручною та природньою. Користувачі звикли до використання жестів у сучасних мобільних застосунках.

Прикладом у застосунку є використання свайпінгу та натискання для навігації та використання скролінгу для переміщення по списку продуктів.

4. **Адаптивність.**

Забезпечення оптимальної роботи застосунку на різних пристроях та розмірах екранів.

UI застосунку розроблявся з розумінням того, що конфігурація екрану та пристроїв може бути різною, прикладом цього є правильне відображення елементів інтерфейсу незалежно від орієнтації екрану:

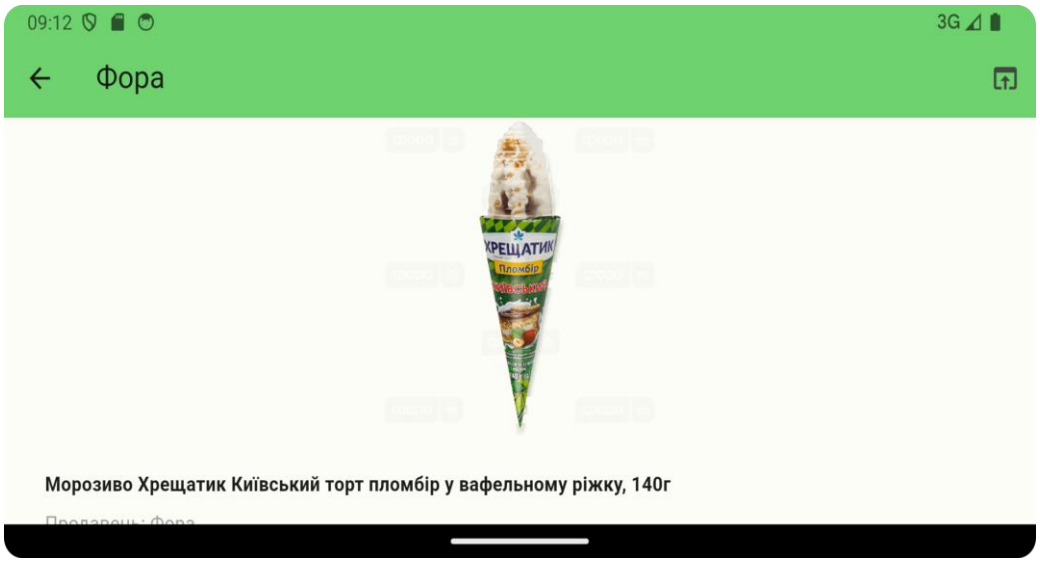


Рисунок 5 – Продукт в горизонтальній орієнтації

5. **Консистентність.**

Збереження однакового стилю та поведінки на всіх етапах користувацької взаємодії з застосунком.

Увесь інтерфейс застосунку було реалізовано у одному стилі та кольоровій схемі, що надає застосунку гарний та професійний вигляд, спрощує концентрацію уваги користувача під час його використання.

ІМПЛЕМЕНТАЦІЯ ПРОДУКТУ

Постановка завдання

Метою проєктної роботи є створення застосунку з товарами продуктових мереж. Для досягнення цієї мети використаємо фреймворк Ruby on Rails та бібліотеку Watir, а також Flutter. Розроблене рішення повинно включати в себе:

- зручний та інтуїтивний інтерфейс;
- адаптивний дизайн;

– фільтр по категоріям;
– пошук по назві;
– відображення актуальних товарів;
– можливість перейти на офіційну сторінку магазину;
– можливість використовувати списки продуктів.

Серверна частина

Скрапінг — це процес автоматичного отримання даних з вебсторінок. Скрапери використовуються для вилучення даних з різних джерел, таких як вебсайти, блоги, соціальні медіа і багато іншого. Скрапери зазвичай аналізують HTML-код вебсторінок, знаходять потрібні елементи і парсингом отримують інформацію з них [11].

Основна ідея вебскрапінгу полягає в тому, щоб отримати доступ до інформації, яка відображається на вебсторінках, і використовувати її для різних цілей, таких як аналіз даних, моніторинг цін, отримання новин або будь-якого іншого вмісту з вебсайту.

Watir може використовуватись для реалізації скраперів, оскільки вона надає можливість взаємодіяти з вебелементами на сторінках. За допомогою Watir можна автоматично переходити по сторінках, заповнювати форми, натискати кнопки і отримувати дані з вебелементів. Це робить Watir потужним інструментом для створення скраперів.

Однак, при реалізації скрапера з використанням Watir важливо дотримуватись правил і обмежень, які встановлюються власниками вебсайтів. Необхідно розуміти та дотримуватись політики конфіденційності, правил використання API або вказівок, якщо вони існують. Також важливо пам'ятати про можливість зміни розміщення елементів на вебсторінці, що може призвести до неправильної роботи скрапера [12].

Першим кроком є аналіз HTML-коду вебсторінки, з якої ми будемо отримувати дані. Можна використовувати інструмент для розгляду джерела сторінки, такий як розширення браузера для розробників або інструменти для інспекції HTML. Це допоможе зрозуміти структуру сторінки та знайти потрібні елементи, які ми бажаємо скрапити.

Оскільки потрібно акційні продукти в продуктових мережах, нас будуть цікавити такі дані про товар: назва, картинка, посилання на сторінку в магазині, ціна, знижка, ціна зі знижкою, дата закінчення знижки та опціонально кількість або вага. На кожну мережу буде свій скрапер зі своїми особливостями.

Основні проблеми які виникали під час скрапінгу:

Складність клієнтської частини. Деякі вебсайти використовують JavaScript і AJAX для завантаження контенту динамічно. Це може ускладнити скрапінг, оскільки потрібно виконати JavaScript і обробити AJAX-запити, щоб отримати повний вміст сторінки. Або ускладнена верстка, для якої потрібно знайти підхід.

Блокування. Якщо робити занадто багато запитів до вебсайту або порушуєте їхні обмеження, ваша IP-адреса може бути блокована. Це може призвести до недоступності даних або обмеження. Тому потрібно встановлювати обмеження при парсингу, наприклад для завантаження однієї картинки надавати декілька секунд.

Зміна розмітки. Вебсайти можуть часто змінювати свою розмітку, що може призвести до того, що ваш скрапер перестане працювати. Якщо потрібні елементи, переміщаються або змінюються, доведеться адаптувати скрапер до нової розмітки.

Формат даних. На кожному сайті може бути свій формат даних, наприклад дата закінчення акції. Потрібно приводити до одного формату. Також нерідко відсутність відсотку знижки, який можна легко порахувати на етапі парсингу. Та кількість або вага продукту може бути присутня в назві самого продукту або окремо.

Основне завдання скрапера, парсити всі акційні продукти з необхідними даними про них. На виході з кожного скраперу мають бути дані в одному форматі. Підготувати

для зберігання їх у зручному форматі, наприклад, у базі даних. Скрапери можуть запускатися в нічний час для оновлення даних.

Побудова API

Після того як скрапери зібрали та підготували необхідні дані про продукти вони вже відсортовані по категоріям, а також компаніям яким вони належать.

Періодично продукти можуть знову з'являтись в акційних пропозиціях і при парсингу існуючого продукту в базі даних нам потрібно лише оновити ціни, дату та знижку відповідного товару. Ми можемо пропускати момент парсингу картинки, що значно прискорить майбутнє оновлення бази даних.

Пошук реалізовано за допомогою регулярних виразів. Отримуючи від користувача параметри знаходяться бажані продукти відповідно до запиту.

В ході проєктування архітектури API було визнано важливість застосування RESTful принципів для досягнення простоти та ефективності взаємодії. Використання стандартних HTTP методів, таких як GET, POST, PUT, та DELETE, дозволило реалізувати зручні та легкі в управлінні ендпоінти для роботи з ресурсами використовуючи JSON.

Однією з ключових складових була забезпечена безпека взаємодії за допомогою механізмів авторизації та використання протоколу HTTPS для шифрування переданих даних. Введення системи версіонування API стало важливим елементом для підтримки сумісності та зручного впровадження змін.

Подальшим кроком було створення RESTful ресурсів, кожен із яких мав унікальний URI та ефективно обробляється через відповідні ендпоінти. Обробка помилок була реалізована через повернення консистентних HTTP статус-кодів та зрозумілих повідомлень про помилки.

У системі буде реалізована можливість пагінації та сортування для зручності користувачів, що дозволяло обмежити кількість повернутих записів та отримувати відсортовану інформацію. Використання кешування та локалізації дозволило підвищити швидкодію та адаптувати інтерфейс для різних мов.

Важливою частиною процесу буде також створення детальної документації API, яка включала опис ресурсів та приклади використання. Тестування буде займати центральне місце для забезпечення стабільності та високої надійності API. Ці кроки були вжиті для забезпечення ефективної взаємодії між мобільним застосунком та бекендом, що сприяє успішній розробці та впровадженню продукту.

Клієнтська частина

В процесі імплементації було:

– Спроєктовано запланований інтерфейс застосунку.
– Створено початковий проєкт Flutter.

Вигляд загальної структури проєкту:

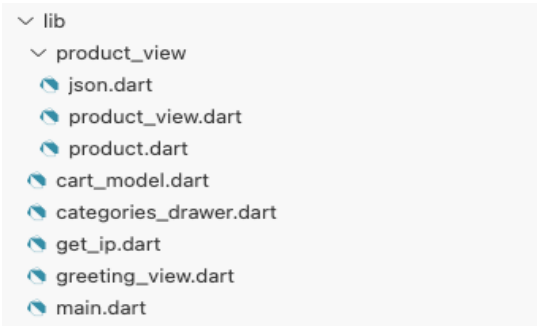


Рисунок 6 – Структури проєкту

– Налаштовано середовище розробки.
– Розроблено необхідні компоненти інтерфейсу.

Приклад імплементації віджету для відображення списку продуктів:

```
class ProductsListView extends StatelessWidget {
  final List<ProductItem> productsList;

  const ProductsListView({super.key, required this.productsList});

  @override
  Widget build(BuildContext context) {
    return ListView.separated(
        itemCount: productsList.length,
        separatorBuilder: (context, index) => Divider(
          height: 3, color: Theme.of(context).colorScheme.primary
        ), // Divider
        itemBuilder: (context, index) => productsList.elementAt(index),
    ); // ListView.separated
  }
}
```

Рисунок 7 – Імплементації віджету

Результуючий інтерфейс списку продуктів:

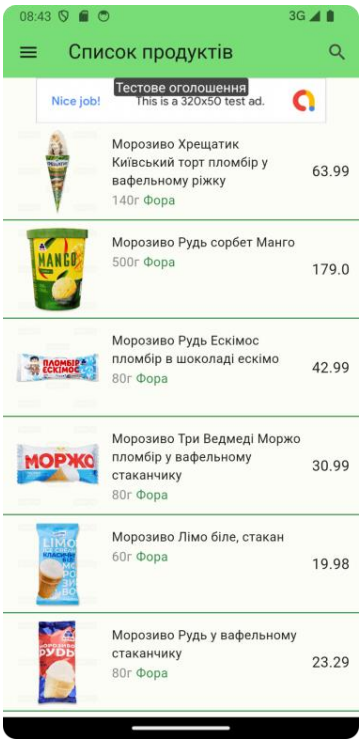


Рисунок 8 — Список продуктів

– Реалізовано взаємодію з бекендом за допомогою обговоренного API.

Приклад реалізації методу, що відповідає за отримання списку продуктів із вебсервера:

```dart
Future<List<ProductItem>> fetchProducts() async {
  final response = await http
      .get(Uri.parse('http://${getAPIAddress()}/api/v1/products/'));

  if (response.statusCode == 200) {
    // If the server did return a 200 OK response,
    // then parse the JSON.
    final products = jsonDecode(response.body) as List<dynamic>;
    return products.map((e) => ProductItem.fromJson(e)).toList();
  } else {
    // If the server did not return a 200 OK response,
    // then throw an exception.
    throw Exception('Failed to load products');
  }
}
```

Рисунок 9 – Реалізація ендпоінту

– Реалізовано навігацію між сторінками застосунку.

Приклад реалізації переходу на сторінку конкретного продукту:

```dart
onTap: () {
Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => ProductViewPage(title: shop, productID: id)));
},
```

Рисунок 10 – Реалізація навігації

ДОКУМЕНТАЦІЯ ПРОЄКТУ

Модель та інструменти ведення документації

Обрана модель ведення документації базується на Agile методології розробки, зокрема, на Scrum підході. Це означає, що документація створюється паралельно з розробкою продукту та визначається потребами команди на кожному етапі розробки.

Для ведення документації використовуються такі інструменти:

– Jira для керування завданнями, а також для планування роботи на різних етапах проєкту;

– Confluence для розміщення спільних для всієї команди документів, протоколів зустрічей тощо.

Види створених документів:

– точка входу в документацію, розміщена в Confluence;

– завдання та етапи: опис завдань у Jira, специфікації етапів розробки;

– планування спринтів і спринт-ретроспективи;

– технічна документація (схема бази даних, інструкція користувача тощо)

Інструкція користувача

При відкритті застосунку завантажується стартова сторінка:

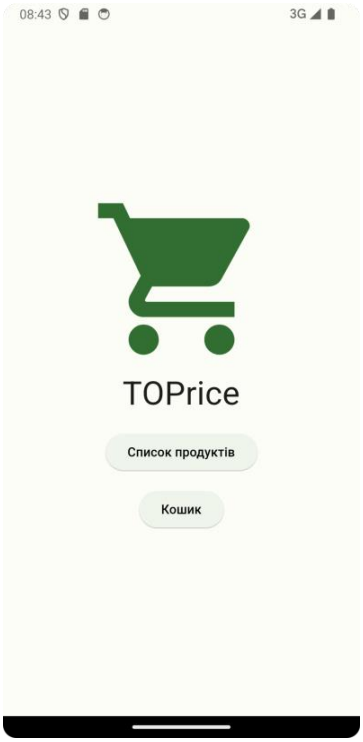


Рисунок 11 – Стартова сторінка

Далі, є можливість перейти до списку наявних продуктів або переглянути кошик раніше обраних продуктів(рис. 12 та рис. 13 відповідно).

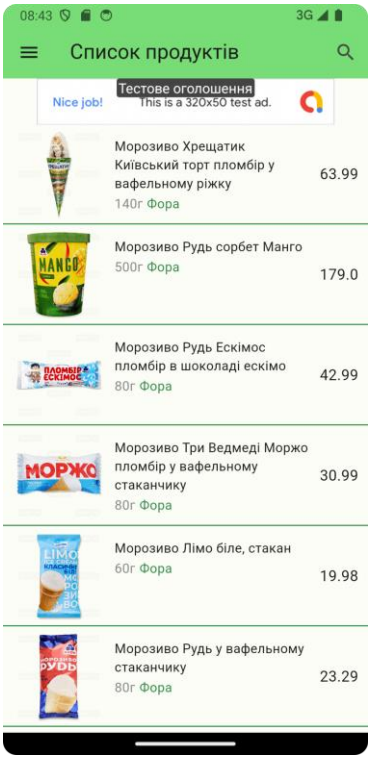


Рисунок 12 – Список продуктів

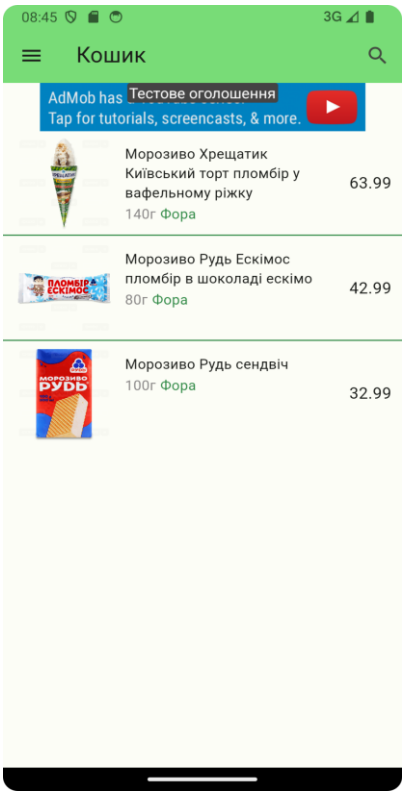


Рисунок 13 – Кошик продуктів

Щоб додати або прибрати товар з кошика, необхідно перейти клацнувши на один з продуктів в списку та натиснути кнопку "Додати в кошик" або "Видалити з кошика", а також перейти на сторінку продукти в магазині (рис. 14).

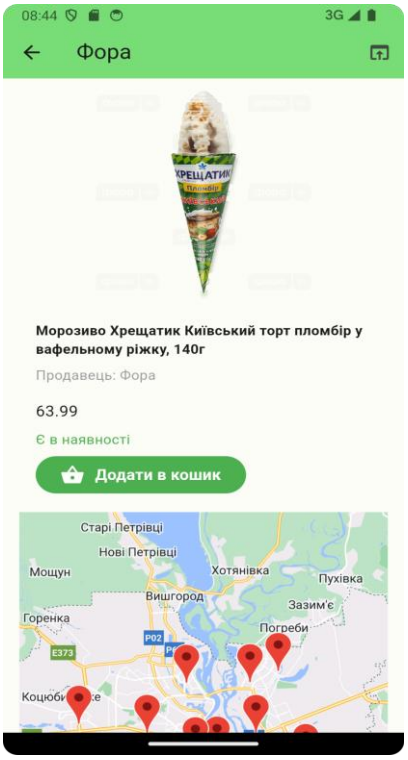


Рисунок 14 – Сторінка продукту

Також в наявності пошук та категорії товарів (рис. 15).

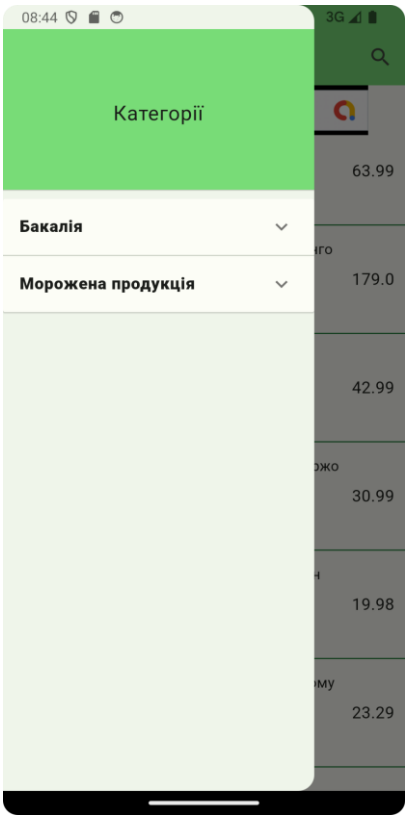


Рисунок 15 – Категорії

ОПИС ПРОЦЕСУ РОЗРОБКИ

Опис команди та ролей у команді

Команда розробки застосунку включає в себе студентів Київського національного університету імені Тараса Шевченка Факультету комп'ютерних наук та кібернетики, та Географічного факультету. Склад команди представлено в таблиці 5.1.

Таблиця 8 – Склад команди

Учасник	**Роль у команді**
Шафран Володимир	керівник проєкту, back-end розробник
Д’ячек Аліна	модератор, аналітик, фахівець з документації
Макаров Дмитро	мобільний розробник
Романків Святослав	мобільний розробник

Опис обраної методології, інструментарію, обґрунтування вибору

У контексті розгляду методології Scrum та її ключового елементу - спринтів, виявлено численні переваги, що роблять цей підхід дуже привабливим для команд розробників та розробників програмного забезпечення.

На перший погляд, гнучкість та адаптивність Scrum виділяють його серед інших методологій розробки. Спрінти, як короткі ітерації, дозволяють швидко реагувати на зміни в вимогах чи ринкових умовах.

Однією з ключових переваг є висока прозорість в роботі, забезпечена регулярними спринт-ретроспективами, що дозволяють команді бачити та аналізувати свою

продуктивність. Також, демонстрація функціоналу на кінці кожного спрінта забезпечує залучення замовника та його активний внесок в процес розробки.

Постійне вдосконалення є іншою важливою характеристикою Scrum, забезпечуючи регулярні ретроспективи для аналізу та покращення робочих процесів. Цей підхід допомагає ефективно управляти ризиками, виявляючи проблеми на ранніх етапах розробки.

Крім того, Scrum сприяє власній організації та мотивації команди, дозволяючи їй самостійно визначати обсяг роботи протягом спрінта. Це підтримує високий рівень відповідальності та ефективності.

Також, визначені та фіксовані часові рамки спринтів полегшують планування та управління часом. Це важливо для забезпечення стабільного та ефективного процесу розробки.

У методології Scrum передбачено регулярні ревізії, що дозволяють легко вносити зміни в продукт під час розробки та фокусуватися на важливих функціях.

Враховуючи зазначені переваги, можна визнати, що методологія Scrum та використання спринтів є ефективними інструментами для розробки програмного забезпечення в умовах швидко змінюючогося ринку та технологій.

Демонстрація планування спринтів

Планування спринтів у команді розробки системи проводиться за методологією Scrum та включає наступні етапи:

1. Визначення функціональностей та завдань, які потрібно реалізувати.
2. Визначення часу та місця проведення планування, а також перегляд поточного стану проєкту.
3. Оцінка складності та трудомісткості кожного завдання за допомогою покажчика Story Points.
4. Визначення кількості завдань, які можна взяти на спринт з урахуванням оцінок та потужності команди.
5. Розподіл завдань між членами команди, визначення завдань, які будуть виконані протягом спринта.
6. Створення конкретного списку завдань, які будуть реалізовані протягом спринта.

На рис. 16 позначені минулі спринти та поточні задачі [13].

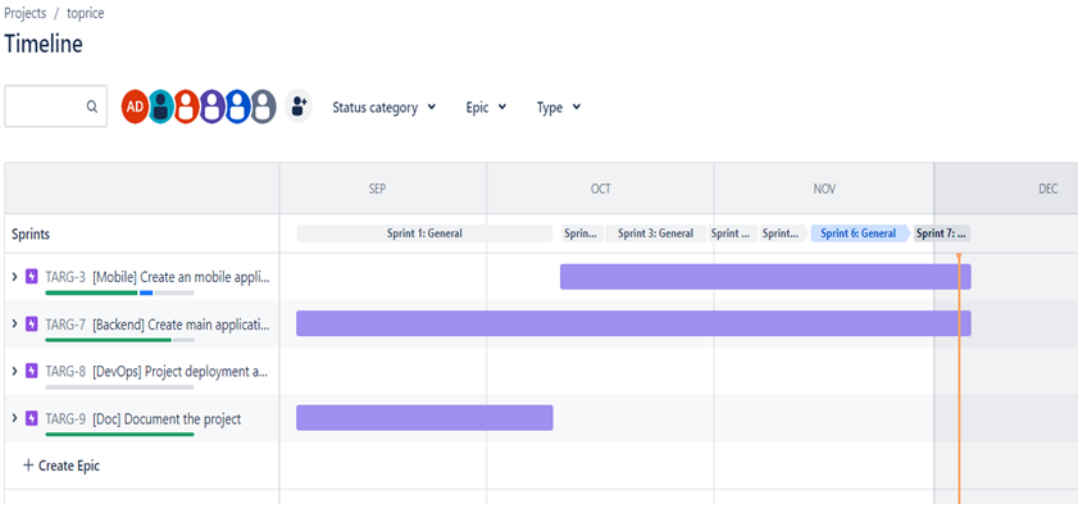


Рисунок 16 – Timeline планування спринтів

Рефлексія

Після завершення кожного спринта проводиться рефлексія з метою івизначення того, що вдалося та чого не вдалося досягти.

Проведена рефлексія після реалізації MVP.

Успішні сторони:

– реалізовано ключовий функціонал, який був визначений для MVP;

– забезпечено наявність функціонального прототипу, який можна випробувати та оцінити;

– виявлено високий рівень співпраці та злагодженості команди під час виконання завдань;

– члени команди ефективно взаємодіяли та обмінювались ідеями для досягнення поставлених цілей.

Проблеми та вирішення:

– в результаті відсутності спеціаліста UI/UX дизайну, деякі аспекти дизайну залишились нереалізованими, що викликає необхідність розглянути можливість залучення нового фахівця в розробку продукту.

Задачі для покращення:

– необхідно запланувати подальше вдосконалення дизайну продукту;

– провести аналіз та визначення додаткових функціональних можливостей, які можна включити в подальших версіях продукту.

Чого навчилися:

– важливість комплексного підходу до розробки застосунку;

– освоєння навичок адаптації до змін та швидкого вирішення проблем для подальшого покращення;

– огляд нових інструментів і покращення технічних навичок.

ВИСНОВКИ

В рамках даного проєкту було розроблено та реалізовано основа продукту, спрямований на полегшення процесу вибору та покупки продуктів для користувачів у супермаркетах. Продукт створено з урахуванням потреб цільової аудиторії, яка включає людей, що цінують економію, швидкість та якість продуктів. Процес розробки системи був здійснений відповідно до методології Scrum, що дозволило ефективно розподіляти завдання та досягати поставлених цілей.

Продукт був спроєктований для забезпечення ефективного збору та аналізу інформації про продукти, рекомендацій для користувачів та можливості порівняння цін. Крім того, застосунок повинен був мати інтуїтивно зрозумілий інтерфейс та забезпечувати стабільну роботу у реальних умовах великої кількості користувачів. В ході розробки були визначені та відпрацьовані вимоги до функціоналу, структури, функціонування та системних характеристик.

MVP був успішно реалізований відповідно до визначених завдань та вимог. Застосунок вже має ключовий функціонал, що дозволяє користувачам порівнювати ціни та економити час при виборі продуктів. Визначено напрями для подальшого вдосконалення, такі як покращення дизайну та розширення функціоналу. Важливим аспектом є гнучкість та готовність команди до швидкої адаптації та розвитку продукту для відповіді на змінні потреби користувачів.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. SuperMarkets [Електронний ресурс]:[Вебсайт] — Режим доступу до ресурсу: https://supermarkets.com.ua/.

2. Сервіс вигідних покупок Costless [Електронний ресурс]:[Вебсайт] – Режим доступу до ресурсу: https://costless.com.ua/
3. LOVESALES [Електронний ресурс]:[Вебсайт] — Режим доступу до ресурсу: https://www.lovesales.com.ua/.
4. Ruby A PROGRAMMER'S BEST FRIEND [Електронний ресурс]:[вебсайт] — Режим доступу до ресурсу: https://www.ruby-lang.org/en/
5. Homebrew Електронний ресурс]:[Вебсайт] — Режим доступу до ресурсу: https://brew.sh/
6. What Is Ruby on Rails, And What Is It Used For? [Електронний ресурс]:[Вебсайт] — Режим доступу до ресурсу: https://www.codingdojo.com/blog/what-is-ruby-on-rails
7. Ruby on Rails [Електронний ресурс]:[Вебсайт] — Режим доступу до ресурсу: https://rubyonrails.org/
8. Dart programming language [Електронний ресурс]:[Вебсайт] — Режим доступу до ресурсу: https://dart.dev/
9. Flutter - Build apps for any screen [Електронний ресурс]:[Вебсайт] — Режим доступу до ресурсу: https://flutter.dev/
10. Tidwell, J. — Designing Interfaces: Patterns for Effective Interaction Design. O'Reilly Media., 2010 — С. 15-16.
11. What is Web Scraping and What is it Used For? [Електронний ресурс]:[Вебсайт] — Режим доступу до ресурсу: https://www.parsehub.com/blog/what-is-web-scraping/
12. Watir [Електронний ресурс]:[Вебсайт] — Режим доступу до ресурсу: http://watir.com/
13. Jira Software [Електронний ресурс]:[Вебсайт] — Режим доступу до ресурсу: https://productsproj.atlassian.net/
14. PostgreSQL [Електронний ресурс]:[Вебсайт] — Режим доступу до ресурсу: https://www.postgresql.org/
15. Ruby [Електронний ресурс]:[Вебсайт] — Режим доступу до ресурсу: https://www.ruby-lang.org/en/

СИСТЕМА АВТОМАТИЧНОЇ КЛАСИФІКАЦІЇ ТЕКСТІВ

Дар'я Кравець, Олександр Лихопуд, Андрій Абдулаєв

Цей проєкт мав на меті розробку автоматичної системи класифікації тексту з використанням трьох відомих методів обробки природньої мови (NLP): Top2Vec, BERTopic і Lbl2vec. Задачею проєкту було створити рішення, яке відповідатиме зростаючій потребі в підвищенні ефективності класифікації тексту в різних областях. Використовуючи одні з найкращих методів NLP для цієї задачі, ми прагнемо автоматизувати категоризацію текстових даних, пропонуючи нестандартні функції та полегшуючи процеси прийняття рішень. Окрім налаштування та оцінки моделей, ми працювали над попередньою та постобробкою даних, шукали додаткові функції для реалізації, щоб разом з стандартним завданням класифікації надати більше корисної інформації про тексти. Експериментальні результати демонструють ефективність обраних методів для автоматизації завдань класифікації текстів. Кожен метод має сильні та слабкі сторони залежно від мети завдання. Враховуючи це, було розроблено рішення для кожного з методів, що забезпечує різноманітність опцій та функцій до використання.

This project aimed to develop an automatic text classification system using three well-known natural language processing (NLP) methods: Top2Vec (an unsupervised method for topic modeling and document clustering), BERTopic (utilizes the BERT language model) and Lbl2Vec (specifically designed for text classification tasks). The task was to create a solution that would meet the growing need to increase the efficiency of text classification in various areas. Using some of the best NLP techniques for this task, we aim to automate the categorization of textual data, offering non-standard features and facilitating decision-making processes. It was important to include all three methods in the study for a more robust evaluation of the text classification. In addition to tuning and evaluating the models, we worked on pre- and post-processing the data, looking for additional features to implement to provide more useful information about the texts along with the standard classification solution. The experimental results demonstrate the effectiveness of the selected methods for the automation of text classification tasks. Each method has strengths and weaknesses depending on the task. Therefore, a solution has been developed for each of the methods, providing a variety of options and functions to use.

МАШИННЕ НАВЧАННЯ, ОБРОБКА ПРИРОДНОЇ МОВИ, КЛАСИФІКАЦІЯ ТЕКСТІВ, ГЕТЕРОГЕННІ ТЕКСТИ, ТЕМАТИЧНЕ МОДЕЛЮВАННЯ

MACHINE LEARNING, NATURAL LANGUAGE PROCESSING, TEXT CLASSIFICATION, HETEROGENEOUS TEXTS, THEMATIC MODELING

ВСТУП

Зі зростанням обсягів текстових даних у різних галузях діяльності людини, розвиток ефективних та якомога більш точних автоматизованих систем класифікації текстів стає надзвичайно актуальним завданням для отримання важливих висновків та полегшення процесу прийняття рішень.

У даному проєкті ми ставимо перед собою задачу розробити систему автоматичної класифікації текстів, використовуючи три відомі методи обробки природної мови (Natural Language Processing, NLP): Top2Vec, BERTopic та Lbl2vec.

Методи дослідження включали аналіз потреб замовника, визначення головних завдань дослідження, пошук різних підходів до реалізації задачі.

Окрім цілей щодо розробки системи, ми також поставили перед собою ціль удосконалити обрані методи шляхом доповнення їх іншими функціями, які під час дослідження нам будуть здаватися потрібними та недостатньо реалізованими.

Ми прагнемо удосконалити свої знання у сфері машинного навчання та, зокрема, методів обробки природньої мови, а також покращити навички програмування, презентації своїх ідей та рішень.

ПОШУК ТА ВАЛІДАЦІЯ ІДЕЇ

Пошук ідеї продукту

Сфера машинного навчання перебуває на стадії активного розвитку, нові рішення з'являються щодня як в галузі комп'ютерного зору, так і в галузі обробки природної мови. Насправді, існує досить багато методів класифікації текстів, від найпростіших статистичних до методів на основі глибинного навчання. Всі вони мають різну мету та різні результати. Ми намагалися знайти найбільш уніфіковане рішення, яке було б застосовне до різних типів задач.

Окрім обраних методів ми також розглядали кластеризацію методом k-середніх, методи моделювання тем від Gensim[1], а також застосування технік UMAP[2] та HDBSCAN[3].

Зрештою ми зупинилися саме на Top2Vec[4], BERTopic[5] та Lbl2vec[6], адже ці методи показують найкращі результати для задач класифікації та кластеризації, є досить гнучкими та легко застосовні в розробці додаткових функцій.

Огляд наявних рішень

Таблиця 1. Порівняння деяких існуючих рішень

	Top2Vec	BERTopic	Lbl2vec	k-середніх	Gensim
Класифікація	-	+	+	-	-
Кластеризація	+	+	-	+	+
Можливість не задавати наперед теми/їхню кількість	+	+	-		
Гнучкість	+	+	+	-	-

За критеріями наведеними в табл. 1 ми попередньо оцінювали методи та обрали перших три для нашого рішення. Окрім цього в процесі експериментів на наборах даних ми також оцінювали точність методів, швидкодію, складність необхідної попередньої підготовки даних та параметрів, формат результуючих даних, тощо.

Мапа продукту

Проблема, яку вирішує наш продукт: відсутність рішення, яке б поєднувало як класифікацію, так і кластеризацію текстів, яке є гнучким, до якого було б легко додавати нові функції окрім як просто класифікації.

Сегменти клієнтів: розробники у сфері машинного навчання; власники програмних продуктів, які шукають рішення для покращення керування наборами текстових даних.

Унікальна ціннісна пропозиція: завдяки можливості переключатися між різними методами класифікації текстів, наше рішення дає змогу по-різному проаналізувати наявні текстові дані, кожне з рішень має свої додаткові функції та розв'язує більше ніж просто задачу класифікації текстів.

Джерела доходів: оплата за виконаний проєкт від замовника.

Структура витрат: заробітня плата розробників, плата за користування платною версією середовища програмування Google Colaboratory, використання серверів.

Ключові метрики: точність методу, швидкодія, гнучкість.

Прихована перевага: розробка уніфікованих функцій обробки текстів, з метою їх застосування у різних методах.

Сильні і слабкі сторони розроблюваного продукту представлені на рисунку 1.

Рисунок 1 – SWOT-аналіз

Огляд використаних технологій

Мова програмування Python: ця мова є найбільш популярною у сфері машинного навчання, більшість бібліотек для аналізу даних та методів штучного інтелекту реалізовано саме для цієї мови, тому наш вибір був очевидним.

Середовище розробки Google Colaboratory: середовище для виконання файлів .ipynb, дозволяє надавати спільний доступ як і до будь-яких інших файлів на Google Drive, дає можливість виконувати файли на відеокарті Tesla T4. Нам було важливо забезпечити спільний доступ до редагування коду та текстових блоків файлів для усіх учасників команди. Також перевагою стала можливість виконання коду на потужній відеокарті, що значно прискорило розробку, в порівнянні з виконанням локально.

Демонстрація роботи завдяки Telegram-боту та сервісу Streamlit. Ми стикнулися з проблемою демонстрації роботи нашої програми. Нам, як розробникам, та викладачам було досить зручно працювати лише з файлами .ipynb в Google Colaboratory. Проте нам потрібна була імплементація нашого рішення з базовим інтерфейсом, щоб демонструвати нашу роботу замовникам. Тож ми знайшли два способи, якими користувалися: Telegram-бот та сервіс Streamlit, що давав змогу деплоїти рішення машинного навчання у вигляді простого вебзастосунку.

Наразі користуємося обома способами через особливості нашого рішення та обмежень зі сторони Telegram та Streamlit.

ОПИС ПРОГРАМНОГО ПРОДУКТУ

Призначення та цілі створення системи

Призначенням є:

- забезпечення належного зчитування даних з файлів різних форматів;

- обробка текстів (видалення цифр, пунктуації, мимодруків, стоп-слів, тощо);
- підбір параметрів для тренування моделей;
- тренування моделей класифікації та кластеризації текстів;
- обробка результатів.

Вимоги до системи

Для коректного та швидкого виконання всіх реалізованих функцій є наступні технічні вимоги:

- наявність відеокарти;
- вхідні дані у вигляді файлів .txt, .json або .csv;
- бібліотеки можуть бути встановлені останніх версій, окрім:
  - tensorflow==2.15.1
  - tensorflow_hub==0.12.0
  - tensorflow_text==2.15.0

Опис реалізованих функцій

Обробка текстів

Задля точнішої роботи методів виконують попередню обробку даних, в нашому випадку текстів з набору даних. Ми впровадили такі етапи: видалення символів, що не належать алфавіту; зведення усіх літер до малих; перевірка слів на правильність написання; видалення стоп-слів; лемматизація.

Моделювання тем

З допомогою різних методів ми реалізовували моделювання тем за заданими текстами. В залежності від методу ми мали в результаті або список тем у вигляді векторів слів, або ж теми визначені одним словом (рис. 2-3).

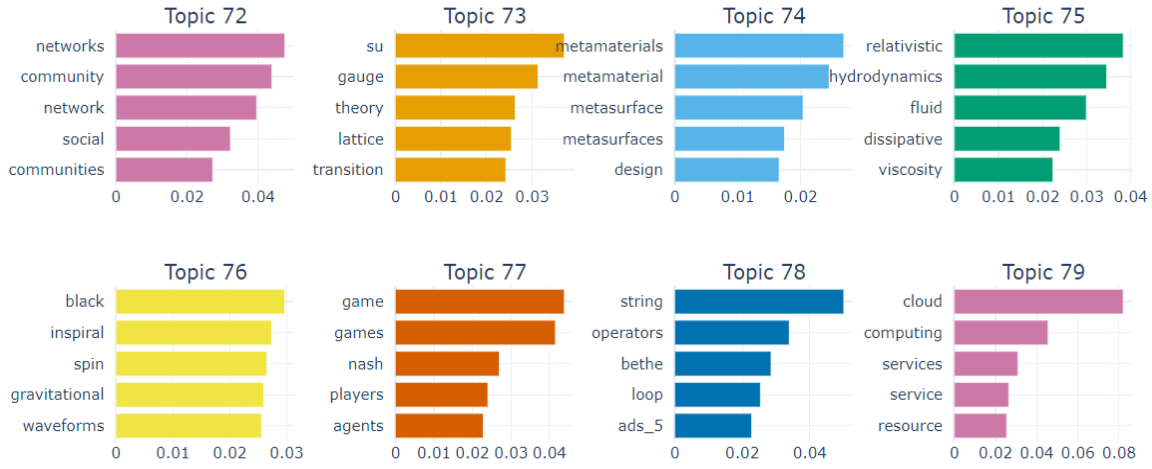


Рисунок 2 – Представлення тем у реалізації методом BERTopic

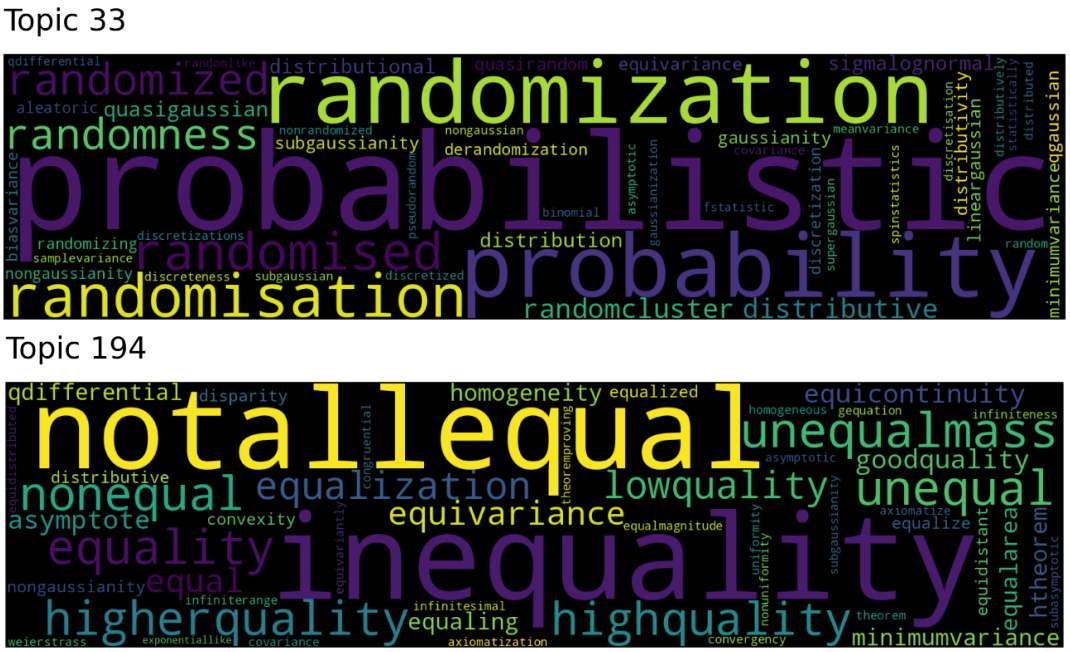


Рисунок 3 – Представлення тем у реалізації методом Top2Vec

У методі Lbl2Vec навпаки ж, теми ми самостійно визначали наперед, адже цей метод вирішував задачу класифікації, а не кластеризації. Тому для набору текстів спочатку ми визначали перелік тем, після чого для кожної теми визначали список найбільш репрезентативних слів в цій темі (рис. 4).

З однієї сторони, це велика перевага, що ми маємо контроль над кількістю тем, їх описом, тощо. Проте в задачах класифікації великої кількості текстових даних це скоріше буде недоліком, адже передбачити всі теми, що можуть зустрітися важко, і тим більше важко підібрати перелік слів на кожну тему такий, щоб він корелював зі словами, які будуть зустрічатися саме в наданих текстах на цю тему.

Проте, чітке визначення тем дає змогу більш якісно оцінювати точність вибраної моделі, тому ми також розглядали способи автоматичного задання переліку слів за темами.

З можливих варіантів були як використання великих мовних моделей, так і об’єднання роботи цього методу з одним з двох інших, які ми розглядали. Ми зупинилися саме на другому варіанті.

```
class_name                                                keywords  \
  astro-ph  [acoustics, audio, speech, signal, noise, musi...
  cond-mat  [light, laser, photon, wave, refraction, diffr...
        cs  [innovation, automation, advancement, disrupti...
      econ  [approximation, differential, discretization, ...
      eess  [ethics, privacy, bias, surveillance, discrimi...
     gr-qc  [superposition, decoherence, computing, wavefu...
    hep-ex  [polymer, liquid, colloid, gel, emulsion, asse...
   hep-lat  [coordination, negotiation, game, communicatio...
    hep-ph  [query, ranking, indexing, relevance, document...
    hep-th  [markov, martingale, brownian, gaussian, rando...
```

Рисунок 4. Представлення тем у реалізації методом Lbl2Vec

Класифікація текстів за темами

Відповідно до знайдених моделлю (або ж заданих вручну) тем відбувається перевірка на відповідність тексту обраній темі. Кожен з методів має свої метрики, але загальна логіка полягає у оцінюванні відповідності тексту до теми, після чого формулюється гіпотеза, що тема з найбільшою оцінкою і буде темою тексту.

Для заданого певним ID документу здійснюється підбір найбільш вдалої теми. На рис. 5 показано такий підбір для деякого тексту з ID 9358 (рис. 5).

```
[ ]  topic_nums, topic_score, topics_words, word_scores = \
         model.get_documents_topics([9358], reduced=False)

     print(f"topic_num:{topic_nums}, topic_score: {topic_score}")
     for word, score in zip(topics_words[0][:10], word_scores[0][:10]):
         print(f"{word:20}: {score}")
```

```
topic_num:[18], topic_score: [0.41695073]
deity                : 0.4635874032974243
divine               : 0.46332794427871704
sinful               : 0.45819300413131714
god                  : 0.4379240870475769
blessed              : 0.42624354362487793
believer             : 0.41524726152420044
righteous            : 0.4123002886772156
worship              : 0.4115995764732361
sinner               : 0.41137436032295227
obedience            : 0.41050368547439575
```

Рисунок 5. Демонстрація підбору теми до тексту

Це є основним результатом нашої роботи, дана функція дозволяє з її допомогою проводити автоматичне визначення теми для кожного тексту з заданого набору даних, проводити так звану індексацію корпусу текстів.

Ієрархічна кластеризація

Також було реалізовано ієрархічну кластеризація текстів, адже виявлення взаємозв'язків між текстами може нести цінну інформацію для аналізу набору даних. Роботу цієї фнкції продемонстровано на рис. 6.

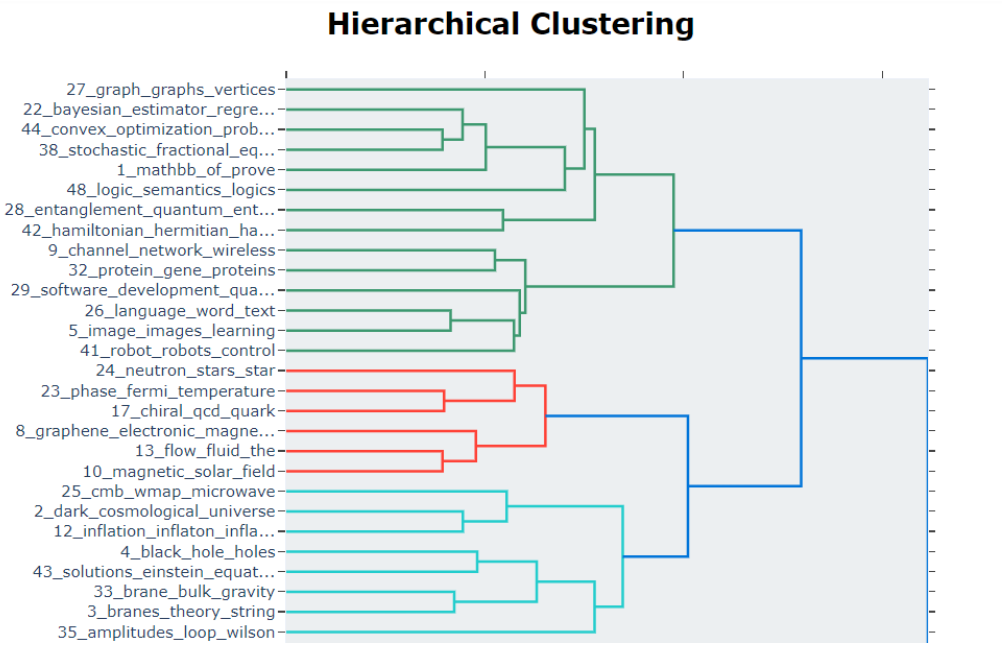


Рисунок 6 – Ієрархічна кластеризація

ОПИС ПРОЦЕСУ РОЗРОБКИ

Учасники команди

У команді було 3 ML-розробника: Дар’я Кравець, Олександр Лихопуд, Андрій Абдулаєв.

За специфікою задачі у нас не було розділення на різні ролі, натомість ми розділяли задачі між собою.

Щотижня у нас було 2-3 зустрічі онлайн. Одна з викладачами, а також одна чи дві (в залежності від потреб) окремо командою.

Фіксували задачі, ідеї, корисні матеріали ми в Notion та Google Docs.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. GitHub - piskvorky/gensim: Topic Modelling for Humans [Електронний ресурс] // GitHub. – Режим доступу: https://github.com/piskvorky/gensim
2. GitHub - lmcinnes/umap: Uniform Manifold Approximation and Projection [Електронний ресурс] // GitHub. – Режим доступу: https://github.com/lmcinnes/umap
3. sklearn.cluster.HDBSCAN [Електронний ресурс] // scikit-learn. – Режим доступу: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.HDBSCAN.html
4. GitHub - ddangelov/Top2Vec: Top2Vec learns jointly embedded topic, document and word vectors. [Електронний ресурс] // GitHub. – Режим доступу: https://github.com/ddangelov/Top2Vec
5. BERTopic [Електронний ресурс] // GitHub Pages. – Режим доступу: https://maartengr.github.io/BERTopic/index.html
6. GitHub - sebischair/Lbl2Vec: Lbl2Vec learns jointly embedded label, document and word vectors to retrieve documents with predefined topics from an unlabeled document corpus. [Електронний ресурс] // GitHub. – Режим доступу: https://github.com/sebischair/Lbl2Vec

CARPATHIAN MYTHOLOGY BASED MOBILE GAME

Roman Vivcharuk, Daria Niemkevych, Diana Dulko, Anastasiia Serytsan, Illia Svyrydov, Mykyta Posmitnyi

Анотація

"Elder Shadows" – однокористувацька 2D рольова гра, заснована на карпатській міфології. Гра пропонує піксельну графіку та спрямована на мобільні платформи, з першочерговим випуском на Android. Проєкт орієнтований на аудиторію шанувальників рольових ігор та людей, зацікавлених українською міфологією. Планується продавати гру в Google Play Store та використовувати такі платформи, як YouTube, Reddit та Discord для залучення спільноти та просування гри. У грі присутні такі основні механіки, як створення персонажа, його переміщення, бій, навички та підвищення рівня разом з покращенням характеристик. Істоти, які з'являються у грі, запозичені з міфології та мають різну поведінку та реакцію на взаємодію з персонажем. Гра також включає статичні та динамічні локації, які генеруються процедурно, щоб забезпечити різноманітність ігрового процесу. Окрім самої гри, було створено вебсайт, який містить інформацію про гру та її розробників, і служить зовнішньою документацією проєкту. Розробка проводилася за методологією Scrum, використовуючи рушій Unity та мову C# для гри, а також PHP для розробки вебсайту.

Annotation

"Elder Shadows " is a single-player 2D role-playing game based on the Carpathian mythology. The game offers pixelated graphics and is aimed at mobile platforms, with the initial release on Android. The project is aimed at the audience of role-playing game fans and people interested in Ukrainian mythology. The game features such basic mechanics as character creation, movement, combat, skills and leveling up along with the improvement of characteristics. The creatures that appear in the game are borrowed from mythology and have different behaviors and reactions on interaction with the character. The game also includes static and dynamic locations that are generated procedurally to provide a variety of gameplay. In addition to the game itself, a website was created that contains information about the game and its developers and serves as external documentation for the project. The development was carried out according to the Scrum methodology, using the Unity engine and C# for the game, and PHP] for the website development.

РОЛЬОВА ГРА, МОБІЛЬНА ГРА, УКРАЇНСЬКА МІФОЛОГІЯ, КАРПАТИ, ПІКСЕЛЬНА ГРАФІКА, МІФОЛОГІЧНІ ІСТОТИ, UNITY

ROLE-PLAYING GAME, MOBILE GAME, UKRAINIAN MYTHOLOGY, CARPATHIANS, PIXEL GRAPHICS, MYTHOLOGICAL CREATURES, UNITY

INTRODUCTION

Assessment of the current state of the object. Today the gaming industry market is one of the most profitable branches of the IT industry both in profits and consumers attention. Most influential owners of global IT companies allocate their resources into gaming in hopes to get more and more gamers to consume their product. Gaming industry became not just a way to gain profits, but to affect customers. This can attract Ukrainian consumers to get interested in foreign content instead of domestic one. Current geopolitical situation in Ukraine shows the need to prevent that and to focus the attention of native customers on their own product.

To this day, there are generally no competitive role-playing games based on Ukrainian folklore, unlike other countries and their games, like Poland and "The Witcher" [4].

The relevance of the project and the reasons for its implementation. There are mainly two goals to this project: first is to lure away Ukrainian gamers from foreign folklore-based games. Gaming industry showed profits of more than $200 billion dollars in 2022 [5], and research shows that more than 63% of Ukrainians play games to "escape routine" [6]. Creating a competitively able product can help our market to get a larger piece of the highlighted profits that will greatly benefit the development of our own gaming industry. Second goal is gaining native consumers' interest in Ukrainian folklore which can contribute in raising the level of knowledge on the topic, national self-awareness and interest in developing native content instead of looking for inspiration in foreign cultures.

The purpose and tasks of the project. The purpose of the project is to develop a competitively-able role-playing single player game based on Carpathian mythology. The tasks are:

- Research the market for available similar projects;
- Research the most popular way to play games among consumers;
- Develop and support a functional game, research market reaction.

Development tools. Due to the simplicity of the development process it was decided upon choosing Unity Game Engine [1] as the main development tool.

SEARCH AND VALIDATION OF THE IDEA

Description of product idea search

The field of game development was chosen because games have always been and will be relevant, so if a high-quality project is created, it can always find its target audience. However, there are quite a lot of games, so only a unique product can impress users. Among all the ideas, three can be distinguished: a game platform, a platform for D&D-like systems, and an RPG.

1) A platform, where multiple minigames were supposed to be placed on. However, after considering all the pros and cons, we realized that the development would be difficult and costly. After all, for this platform, it will be necessary to create some games that could be of interest.

2) A platform for D&D-like systems. Dungeons & Dragons [7] is a fantasy role-playing board game. Although this was an interesting idea, it was rejected. After all, the risk was in the ownership rights of the games. It also required complex development. And the target audience, especially in Ukraine, is quite small. Therefore, the necessity of such a product was in question.

3) The idea of RPG - Role-Playing Game - turned out to be the most attractive. After all, such games are quite popular, which means that our product can be useful. In order for this game to be different from other games, it was decided to create it in the style of Ukrainian mythology, where the characters were various creatures, such as a Lisovyk, a Chugaistr, a Mavka and others. This does not exist yet, and now Ukrainian culture is gaining more and more popularity.

We chose the field of games because a huge number of people play games. This area is always relevant. We aim to create a quality game based on Ukrainian mythology. Our product is unique in its kind. We believe that this game will find its fans in Ukraine, and we hope that also beyond its borders. Inspired by this idea, we want to broadcast love for Ukrainian culture through our product.

Overview of IT products available on the market

1. The game "Bastion" [8]

It is an action RPG from Supergiant Games where players are invited to explore the post-apocalypse world, collect resources, develop their character and solve puzzles.

The game "Bastion" is intended for a wide range of players. May be popular with gamers of all ages (3+), especially those looking for unusual games.

Fully available in English. Text and subtitles are available, but audio is not supported in French, German, Italian and Spanish.

The business model is based on an integration/transaction model of monetization. The game is sold on a one-time basis across multiple platforms (e.g. Steam, consoles, mobile) and players only pay once to access the game and its content.

The game costs $325.

There is an application for Android and iOS.

More than 5,000 downloads per month.

Advantages:

- A flexible character development system allows you to customize your play style.
- The game is also available on various platforms, making it accessible to a wide audience of gamers, and has received numerous positive reviews, indicating its success and popularity among the gaming community.

Disadvantages:

- Limited game duration, especially if you don't consider watching content on repeated playthroughs.
- Some users may find the fights lacking in variety.

2. The game "Children of Morta" [9]

It is an action RPG that tells the story of the Bergson family, who act as the main characters and defenders of their world from dark forces.

The target audience can be diverse in age but mainly consists of players who are looking for deep and rich gameplay.

Fully supported in English, Japanese and Persian. Text and subtitles are available in another 10 languages.

The business model of the game "Children of Morta" is integration/transactional. The game is sold as a one-off across various gaming platforms, and players only pay to access the game itself and its content.

The game costs $329.

There is no application for Android and iOS.

Advantages:

- Dynamic battles with various heroes, each of which has unique skills and play styles.
- Stunning pixel art and detailed locations make the game world incredibly beautiful.
- A deep plot: the story of the Bergson family, ignores common stereotypes in the genre.
- Players can play Children of Morta together with friends, overcoming challenges together.

Disadvantages:

- For some players, the game can become monotonous due to the constant need to go through the same levels.

3. The game "Hades" [10]

It is a roguelike game with action RPG elements. The main character, Hades, the son of the god Hades, goes in search of Persephone's mother, who was kidnapped and sent to hell.

The target audience may include players of all ages (3+) who are looking for an immersive gaming experience with an engaging story.

Fully supported in English. Text and subtitles are available in another 10 languages.

The business model of the "Hades" game is integration/transactional. The game is sold as a one-off across various gaming platforms, and players only pay to access the game itself and its content.

The game costs $515.

There is no application for Android and iOS.

Advantages:

- "Hades" offers dynamic and action-packed battles with a lot of weapons and skills, which make each stage of the passage exciting and varied.
- Beautiful pixel art that creates a rich world of Hell.
- Deep Story: the game reveals complex relationships between gods and characters and makes a heroic epic out of each playthrough.
- An impressive soundtrack and voice acting make the game even more immersive.

Disadvantages:

- Since this is a roguelike, sometimes random factors can cause unfair situations or decisions in the game.

As we can see, there are quite popular RPGs that can be direct competitors of our product. Of course, many moments of the game will be similar to other games. However, the advantage of our product is that the game is based on Ukrainian mythology. This is its uniqueness.

Product Map

Project value. The project aims to deliver a captivating and immersive single-player 2D RPG game rooted in Ukrainian mythology. The game's pixel-based graphics, diverse classes, upgradeable skills, and dynamic world exploration contribute to its entertainment and educational value, promoting Ukrainian folklore. The cultural richness of the game enhances its appeal, offering players a unique and enriching gaming experience.

Customer Segments. Gamers and RPG fans: targeting individuals who enjoy RPG games and are interested in exploring mythology. Indie game supporters: attracting fans of indie games who appreciate unique and niche gaming experiences. Mainly targeting people aged 15 - 35.

Cost structure. Expenses for assets (effects, sounds, music) - estimated 25$. Expenses to register on the distribution platform - estimated 25$.

Revenue streams. Income generated from selling the game on Google Play Store. Estimated cost of game - 1$.

Channels. Official website: providing a central hub for game information, lore, bestiary, and team details. Gaming events: participating in gaming conventions and events to showcase the game and connect with potential players. Digital distribution platforms: utilize Google Play Store to distribute the game. Social media: using platforms like YouTube, Reddit, and Discord for community engagement, updates, and promotions.

Key metrics. Sales metrics: tracking the number of game copies sold and revenue generated. Player feedback: analyzing player reviews and feedback on platforms and forums.

Strengths and weaknesses of a product Strengths:

- Available foreign analogues which developers can take as inspiration or a guide to develop a better product;
- No native competitors, the project is one of its kind, which secures safe work environment and no comparison of the product with other alike;
- Each developer is native Ukrainian and has great knowledge of Ukrainian folklore, that will help in developing a relevant undisprovable game lore;
- Each developer, as a native Ukrainian, has a personal interest in creating a successful project.

Weaknesses:

- Insufficient funds. Project like this requires a lot of work, from development to creating lore and design
- Insufficient time. None of our team members are getting any profit from this project yet, which limits the time every member can allocate.

PRODUCT DESCRIPTION

Purpose and goals of creating the system

Purpose of the system

Game

The main purpose of the system is to provide Ukrainian consumers with Ukrainian folklore-based RPG game to replace foreign competitors while getting their attention in native culture.

Website

The site will be a kind of guide to the game. There you can find information about characters, mechanics, bestiary, and more. There will also be articles, news, and game rules. Users will have their own profiles and will be able to add comments.

Goals

The main goal of the project is to get Ukrainian customers to consume a native product, while also sharing our folklore with foreign gamers. The website is created to be a guide to the game. This will be a useful addition to the game, because the site will become a collection of all information related to the game.

Project requirements

General requirements

The game is supposed to get users' attention for a long-term period, thus having a wide functionality to explore is a must. In addition, it is supposed to contain basic but stable mechanics of any RPG game, like basic combat, leveling up, equipment system, upgrading stats of the main character. Most importantly, all features need to be realistic and if possible based on folklore. For the website, it needs to contain all information about the game, team and other in an easy-to-access manner.

Structure requirements

Since the game is single player, there is no strong need in connecting it to a server with administrative intervention, which means that for the time being, all the systems should be independent in a long term development. This means that, for example, for the stats upgrade system a simple linear function to determine character improvement is not enough, and in RPG games these equations is a must. Considering this, a separate reliable function was created and tested for each upgradeable stat. Among that, the idea of the project implies a great amount of future developments and additions to existing functionality, thus requiring a stable modifiable structure for easier amplification of available content.

In addition, the game is supposed to be available for any mobile device, which means that any feature existing or added later needs to be optimized to be the best quality while being not too much for the device to handle.

System requirements

During the research stage of the project the most controversial topic was choosing the target platform for the game. After long discussion it was decided to focus on mobile platforms, since they are the most accessible way to play games currently in the tech industry. Since the game does not use any complicated technology, any mobile device, Android- or IOS-based, can be used to play. But due to the importance of detailed and realistic visualization of the content, some devices may struggle with rendering. In future, the game will be optimized to function best on any device.

Information Database organization description

Logic structure of database

All information about database structure is shown on figures and tables below (fig.1, tab. 1-8).

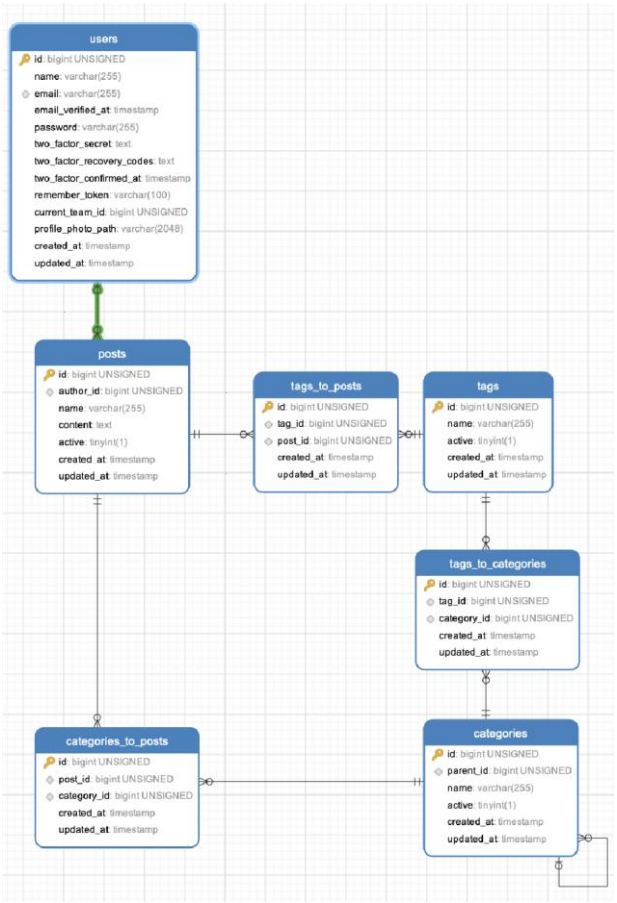


Figure 1 – Logic structure of database

Table 1. Database tables

Number	Table	Description
1	users	A table for storing user information
2	posts	Stores information about posts published by users.
3	tags	Stores information about tags that can be assigned to posts or categories.
4	categories	Stores information about the categories into which posts can be divided.
5	categories_to_posts	Connects posts with relevant categories.
6	tags_to_posts	Connects posts with relevant tags.
7	tags_to_categories	Connects tags with corresponding categories.

Table 2. Users table description

Users Attribute	Type	Description
ID	integer	Identifier
name	varchar(255)	Name
email	varchar(255)	email
email_verified_at	timestamp	user email confirmation time
password	varchar(255)	storing the hashed user password

Users Attribute	Type	Description
two_factor_secret	text	storage of the secret key of two-factor authentication
two_factor_recovery_codes	text	storing two-factor authentication recovery codes
two_factor_confirmed_at	timestamp	confirmation time of the user's two-factor authentication
remember_token	varchar(1000)	token storage
current_team_id	bigint UNSIGNED	the ID of the current team to which the user belongs
profile_photo_path	varchar(2048)	storing the path or file ID of the user's profile photo
created_at	timestamp	time of creating a user record in the database
updated_at	timestamp	the time of the last update of the user record in the database

Table 3. Posts table description

posts Attribute	Type	Description
ID	bigint UNSIGNED	Identifier
author_id	bigint UNSIGNED	ID of the author of the post
name	varchar(255)	saving title of the post
content	text	storing the main content or text of the post
active	tinyint(1)	be used to indicate the activity status of the post (1 - active, 0 - inactive)
created_at	timestamp	the time of creating a record about a post in the database
updated_at	timestamp	the time of the last update of the post record in the database

Table 4. Tags table description

tags Attribute	Type	Description
ID	bigint UNSIGNED	Identifier
name	varchar(255)	tag name
active	tinyint(1)	be used to indicate the activity status of the tag(1 active, 0 - inactive)
created_at	timestamp	the time of creating a record about a tag in the database
updated_at	timestamp	the time of the last update of the tag record in the database

Table 5. Categories table description

categories Attribute	**Type**	**Description**
ID	bigint UNSIGNED	Identifier
parent_id	bigint UNSIGNED	specifies the ID of the parent category, which selects the category hierarchy.
name	varchar(255)	category names
active	tinyint(1)	be used to indicate the activity status of the category (1 - active, 0 - inactive)
created_at	timestamp	the time of creating a record about a category in the database
updated_at	timestamp	the time of the last update of the category record in the database

Table 6. Tags to post table description

tags_to_posts Attribute	**Type**	**Description**
ID	bigint UNSIGNED	Identifier
tag_id	bigint UNSIGNED	field is used to establish a relationship between a record and a specific tag
post_id	bigint UNSIGNED	field is used to establish a relationship between a record and a specific post
created_at	timestamp	indicates the time of creation of the entry in the table
updated_at	timestamp	indicates the time of the last update of the record in the table

Table 7. Tags table description

tags_to_categories Attribute	**Type**	**Description**
ID	bigint UNSIGNED	Identifier
tag_id	bigint UNSIGNED	field is used to establish a relationship between a record and a specific tag
category_id	bigint UNSIGNED	field is used to establish a relationship between a record and a specific category
created_at	timestamp	indicates the time of creation of the entry in the table
updated_at	timestamp	indicates the time of the last update of the record in the table

Table 8. Categories to posts table description

categories_to_posts Attribute	Type	Description
ID	bigint UNSIGNED	Identifier
post_id	bigint UNSIGNED	field is used to establish a relationship between a record and a specific post
category_id	bigint UNSIGNED	field is used to establish a relationship between a record and a specific category
created_at	timestamp	indicates the time of creation of the entry in the table
updated_at	timestamp	indicates the time of the last update of the record in the table

PRODUCT/SYSTEM IMPLEMENTATION

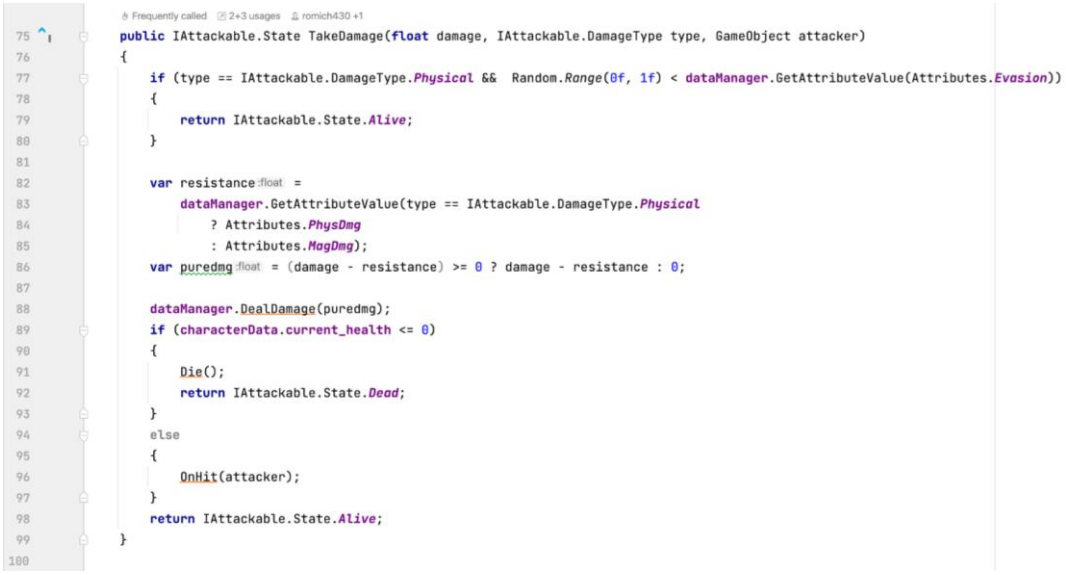
The implementation of the system can be divided into four main parts: development of the main character, development of lore, entities and NPCs, development of map generation and interaction and development of a website.

Development of the main character

The main functionality of any RPG game is the ability to control the character. This includes general movement, physics, combat system, interaction with other elements etc. Since the game is single-player and there can be only one main character, it was decided to use Singleton pattern to create one

CharacterController class that controls all other character-related modules (fig. 2, 3).

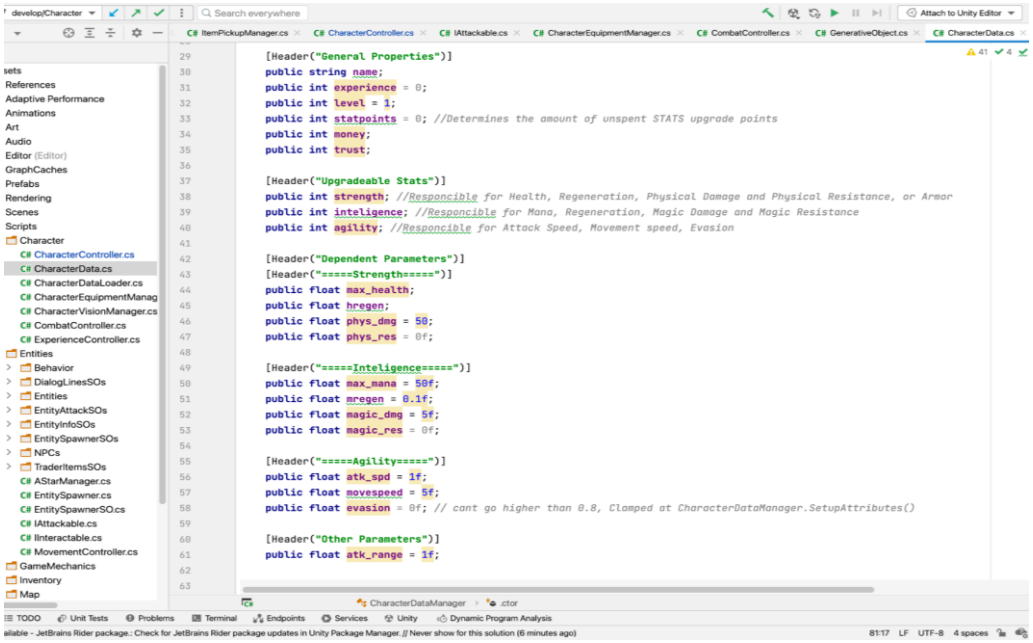
Figure 2 – Part of CharacterController script



```
Frequently called  2+3 usages  romich430 +1
public IAttackable.State TakeDamage(float damage, IAttackable.DamageType type, GameObject attacker)
{
    if (type == IAttackable.DamageType.Physical &&  Random.Range(0f, 1f) < dataManager.GetAttributeValue(Attributes.Evasion))
    {
        return IAttackable.State.Alive;
    }

    var resistance :float  =
        dataManager.GetAttributeValue(type == IAttackable.DamageType.Physical
            ? Attributes.PhysDmg
            : Attributes.MagDmg);
    var puredmg :float  = (damage - resistance) >= 0 ? damage - resistance : 0;

    dataManager.DealDamage(puredmg);
    if (characterData.current_health <= 0)
    {
        Die();
        return IAttackable.State.Dead;
    }
    else
    {
        OnHit(attacker);
    }
    return IAttackable.State.Alive;
}
```

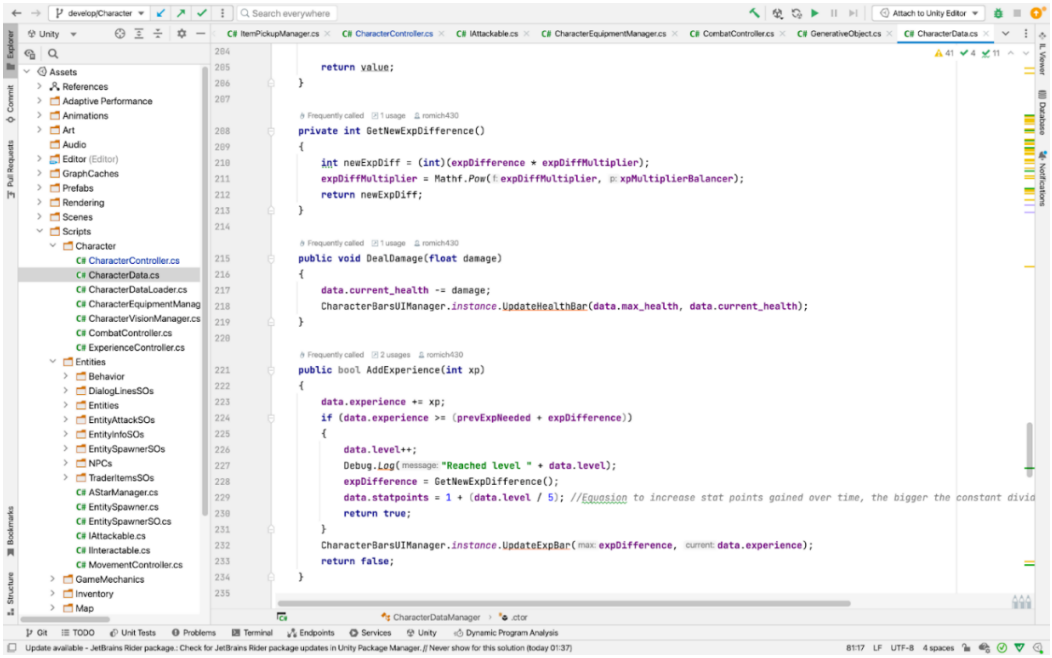
Figure 3 – Part of CharacterController script

The data system was implemented next. Class CharacterData was created to store any data related to character, like name, level or stats (fig. 4). Additionally, CharacterDataManager class was created to serve as a bridge between character data and any other system that requires that data (fig. 5).



```
[Header("General Properties")]
public string name;
public int experience = 0;
public int level = 1;
public int statpoints = 0; //Determines the amount of unspent STATS upgrade points
public int money;
public int trust;

[Header("Upgradeable Stats")]
public int strength; //Responcible for Health, Regeneration, Physical Damage and Physical Resistance, or Armor
public int inteligence; //Responcible for Mana, Regeneration, Magic Damage and Magic Resistance
public int agility; //Responcible for Attack Speed, Movement speed, Evasion

[Header("Dependent Parameters")]
[Header("=====Strength=====")]
public float max_health;
public float hregen;
public float phys_dmg = 50;
public float phys_res = 0f;

[Header("=====Inteligence=====")]
public float max_mana = 50f;
public float mregen = 0.1f;
public float magic_dmg = 5f;
public float magic_res = 0f;

[Header("=====Agility=====")]
public float atk_spd = 1f;
public float movespeed = 5f;
public float evasion = 0f; // cant go higher than 0.8, Clamped at CharacterDataManager.SetupAttributes()

[Header("Other Parameters")]
public float atk_range = 1f;
```

Figure 4 – Character stats in CharacterData



```
    return value;
}

Frequently called | 1 usage | romich430
private int GetNewExpDifference()
{
    int newExpDiff = (int)(expDifference * expDiffMultiplier);
    expDiffMultiplier = Mathf.Pow(f: expDiffMultiplier, p: xpMultiplierBalancer);
    return newExpDiff;
}

Frequently called | 1 usage | romich430
public void DealDamage(float damage)
{
    data.current_health -= damage;
    CharacterBarsUIManager.instance.UpdateHealthBar(data.max_health, data.current_health);
}

Frequently called | 2 usages | romich430
public bool AddExperience(int xp)
{
    data.experience += xp;
    if (data.experience >= (prevExpNeeded + expDifference))
    {
        data.level++;
        Debug.Log(message: "Reached level " + data.level);
        expDifference = GetNewExpDifference();
        data.statpoints = 1 + (data.level / 5); //Equasion to increase stat points gained over time, the bigger the constant divi
        return true;
    }
    CharacterBarsUIManager.instance.UpdateExpBar(max: expDifference, current: data.experience);
    return false;
}
```

Figure 5 – Example of CharacterData methods

In addition to these and all other modules that complete the main character, user interface was created to allow users to comfortably manage any open character information. User interface (fig. 6) includes controls, like movement joystick, "Attack" button, "interact" button, visual representative for health, experience and mana, and a simple character menu (fig. 7, 8), that contains information about stats, ability to upgrade them on leveling up and access to the inventory and equipment system.

Figure 6 – User interface

Figure 7 – Character menu for stats

Figure 8 – Character menu for inventory

NPCs and quests

Quests have unchangeable and changeable data. To keep unchangeable information about a quest, we use scriptable objects designed to save immutable data. To track quest progress during runtime, the quest manager instantiates quest steps, as they are inherited from MonoBehaviour, and, if needed, can monitor player actions every frame. All quest steps are saved as prefabs, and scriptable objects have references to them so they can be easily tracked and created with necessary information. Each type of quest has a different quest step class, but they are all inherited from the base class and can be accessed interchangeably. All steps and global quest state are converted to JSON and added to persistent data to save quest progress between sessions. Then, this data is retrieved and converted back to objects on the next runtime. Most of the quest system interactions are implemented through QuestManager events. At the same time, QuestManager is realized as a singleton, so it is easily accessible to other system elements. Quest givers pull their quests' states from QuestManager and operate on them using its methods; they are also responsible for showing players the quest UI.

A similar approach is used for attacking entities, but there is no need to save changeable data between sessions for them, as new entities are generated every time a scene is loaded. In this situation, prefabs are also used to set visuals and colliders for entities. Different attacks

and their parameters for entities are also stored using scriptable objects, so you can reuse the same attack for different entities. For finding paths during moving entities from one position to another, A* Pathfinding Project [11] is used. When a map is generated, it scans colliders and marks obstacles on the grid so that it can build paths on it. At first, it needs to find a path, and then the entity can move on it every frame.

NPCs' behavior is regulated by behavior trees created for them, where each leaf in the tree represents one action or check. These nodes are united by selector and sequence nodes in one behavior tree. This system's advantages are the modularity and reusability of nodes for different NPCs. Using this system, it is also easy to connect animations to behavior, as you just need to trigger the right animation from the corresponding node.

Game Locations

In this version of our game there are locations that are made by hand and some are procedurally generated. Manually made locations are Players House and Village, while generated locations are Home Location and Dynamic Locations.

Players House is a place where players can meet Domovyk. Currently this location is empty and doesn't have much interaction, but it will be updated a lot in future versions of the game. Village location is a place where players can trade and take quests from villagers. Home Location is a place where the player's house is. There are a lot of mythical creatures and sometimes players can meet lonely villagers that ask for help. Last location for now is Dynamic Location. This location doesn't have any structures yet, but is planned to be used for looking for standard and rare resources.

Maps are currently generated according to their type. Generation is done using the Perlin noise. This is a fairly popular method for generating locations that is widely used in the gaming industry. For our task, two noises are used, which are responsible for humidity and altitude.

Currently there are four map types that can be procedurally generated, which are field, forest, mountains and general. Map types options are stored in scriptable objects and include information about noises values, possible types of map objects and their chances to appear.

Script that is used to create the generated locations is MapGeneration. This script is attached to the _Map object, with the help of which you can set the parameters depending on which the map will be generated (fig. 9).

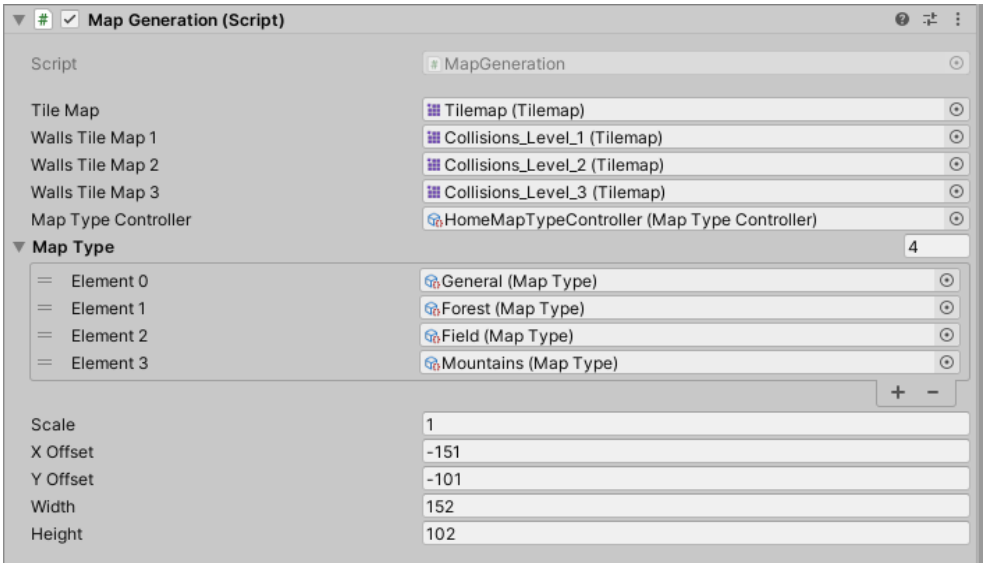


Figure 9 – MapGeneration parameters example

Lakes, trees and similar map objects have colliders that make them impassable. There are also map borders that do not allow the character and the camera to go beyond the game area. For a more realistic spatial perception of the objects on the stage, the order of sorting objects by the Y-axis was included.

The MapManager script is responsible for saving and loading the map (fig. 10). It implements the SaveLevel and LoadLevel methods that perform the corresponding tasks. When saving, each tile or object is written along with its position to the LevelData object, which will then be written to a json file. The map is loaded by reading these files and converting the received data using an object of the LevelData class into game tiles and objects.

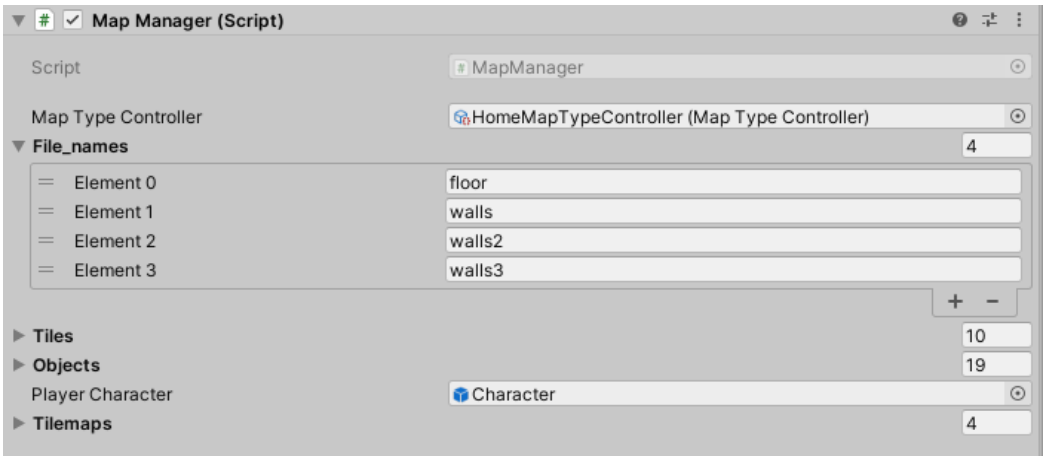


Figure 10 – MapManager parameters example

Website development

In the realm of web development, a comprehensive administrative panel has been crafted to facilitate seamless content management by administrators. This panel offers an intuitive interface empowering managers to efficiently edit and update various aspects of the content (fig. 11).

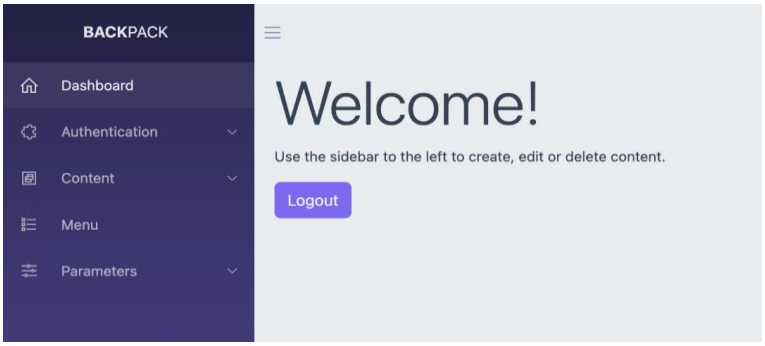


Figure 11 – Website general view

One notable enhancement is the dynamic page constructor (fig. 12), enabling the creation of static pages without the need for direct programming intervention. This empowers non-technical users to contribute to the platform's content, fostering a more collaborative and agile development process.

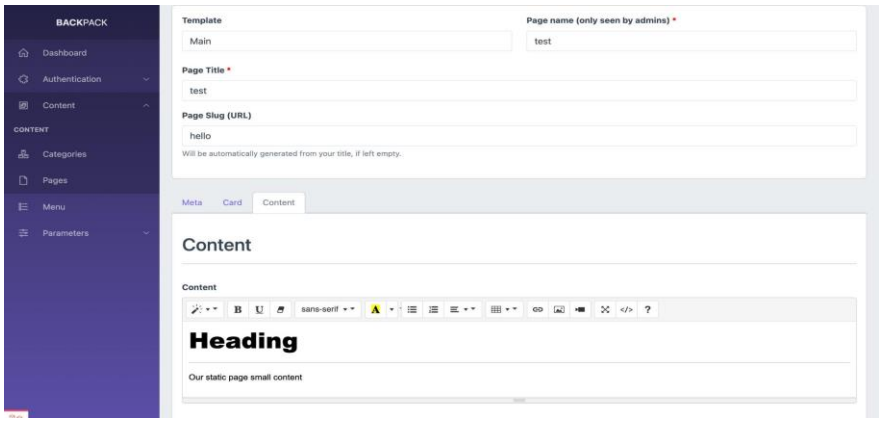


Figure 12 – Page constructor

To streamline the design process, a set of templates and placeholders have been meticulously designed, providing a foundation for consistent and visually appealing web pages. The visual style of the website has been carefully curated, ensuring a cohesive and engaging user experience.

From a technical standpoint, the development effort extended to the creation of controllers and entity models specifically tailored for the seamless creation and editing of various objects within the system. These components play a crucial role in ensuring the underlying structure of the web application is robust and extensible.

Furthermore, a user authentication module has been seamlessly integrated, incorporating user roles and permissions (fig. 13). This not only bolsters the security of the system but also establishes a flexible framework for managing different user levels and their corresponding access rights.

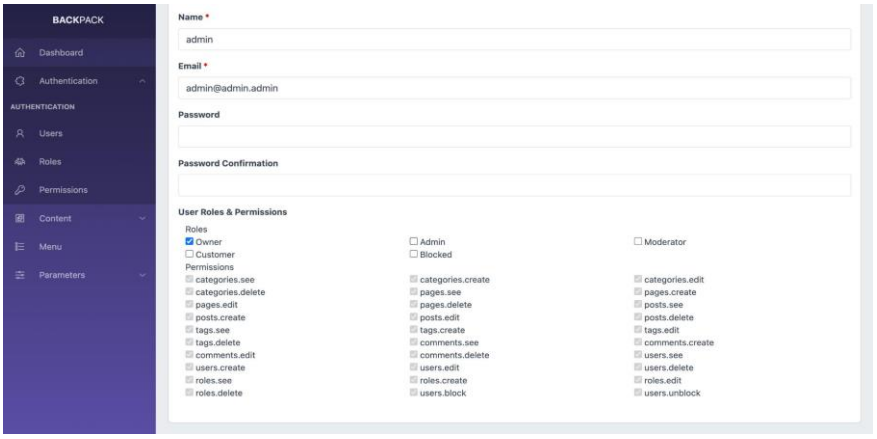


Figure 13 – Configuration of user profile in admin panel

Added a commenting system where users can share their thoughts on articles and posts, making discussions more dynamic and the website more engaging. Additionally, developed a system to track which articles are popular based on factors like views and comments. This helps us understand what topics resonate with our audience and fosters a vibrant community by highlighting engaging content and encouraging meaningful discussions.

TESTING

Since conventional unit testing is not that effective in game development (simple features require too complicated tests, considering the amount of features it is more efficient to use

manual testing), almost every feature was tested manually. Most testing was targeted to analyze how different devices deal with graphics and operations. One of the few features that required actual automatic testing was leveling up system, since it was based on a math formula, created to determine the amount of experience needed to advance to next level after each level advancement. Created test calculated this amount after each level for 100 iterations and displayed them in the console. Different formulas were tested, but after consideration it was decided to use one shown on this image (fig. 14):

```
1 usage
158     private int GetNewExpDifference()
159     {
160         int newExpDiff = (int)(expDifference * expDiffMultiplier);
161         expDiffMultiplier = Mathf.Pow(f: expDiffMultiplier, p: xpMultiplierBalancer);
162         return newExpDiff;
163     }
164
```

Figure 14 – Formula to calculate experience needed for the next level

PROJECT DOCUMENTATION

For recording our documentation, we used Google Disk and project pages in our Jira [12] project, which provided a centralized access point to all documents in Jira. Our documentation model was similar to the "throw it over the wall" model. Still, some critical parts of the project's general planning were documented as soon as they were discussed at the beginning of project development.

At the beginning of development, the team created a project map and project charter and documented a detailed overview of competitors. After that, use case diagrams were made for the game and web parts of the project. The team also decided what content should be in the MVP and documented it in the MVP estimate.

During development, retrospectives were documented for every sprint. All tasks were added to the project backlog and then moved to a backlog of the sprint. These tasks are recorded in our Jira project, where each can be checked in what sprint it was done. We created a separate document for content with statistics and descriptions to keep track of all entities and quests.

Our web part of the project serves as external documentation for the game, as it has pages with information about the game's content.

As we are developing the game, the sphere of use of our product is regulated by the Law of Ukraine "On digital content and digital services"[13].

Instructions for users

To start the game tap on the "Play" button (fig. 15).

Figure 15 – Start menu

To move your character, use the joystick in the bottom left corner, and to attack enemies, use the button in the bottom right part of the screen (fig. 16).

To open your inventory tap on the button in the top right corner (fig. 16).

Figure 16 – Home location

You will see items in your inventory and equipped weapons and armor (fig. 17). You can drag and drop items between slots in your inventory. Tap on the "Stats" button to see your character's current stats.

Figure 17 – Inventory menu

There you will see your base stats and be able to increase them after leveling up (fig. 18). Tap on the "Skills" button to see the skills of your character.

Figure 18 – Stats menu

In the skills menu (fig. 19), you will see a skill tree with available skills and be able to learn new skills. In the "Equipped skills" tab you can equip or unequip learned skills.

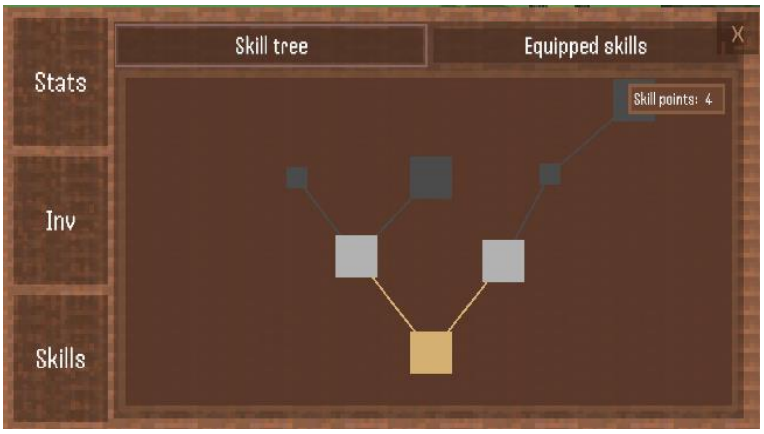


Figure 19 – Skills menu

When speaking with NPCs, you can accept their quests by tapping on the "Accept" button (fig. 20). If you don't like the quest, you can decline it, and maybe the quest giver will propose another one (fig. 21).

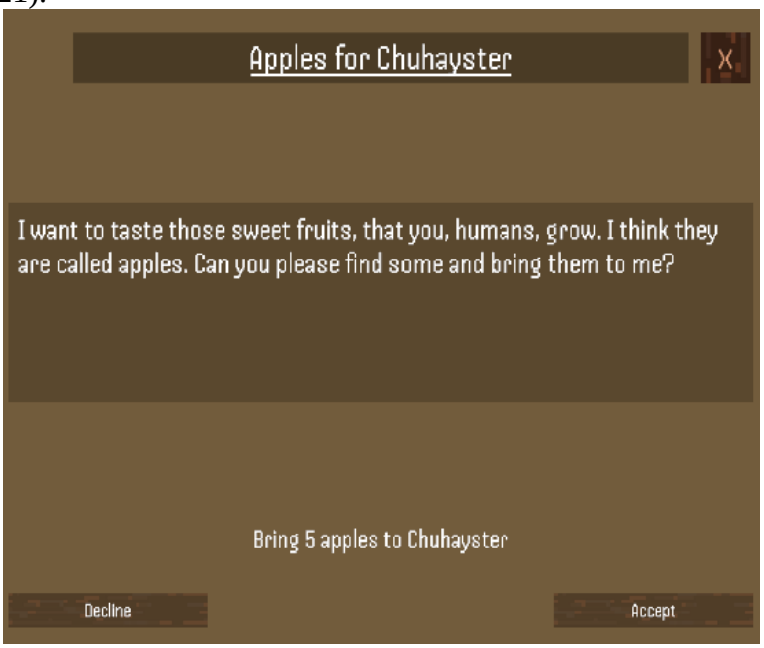


Figure 20 – New quest

If you have a quest in progress and speak with the NPC who gave it to you, you will see information about the quest and the option to abort it. The quest will finish automatically if all objectives are met and you speak with the quest giver. After that, you will receive your reward.

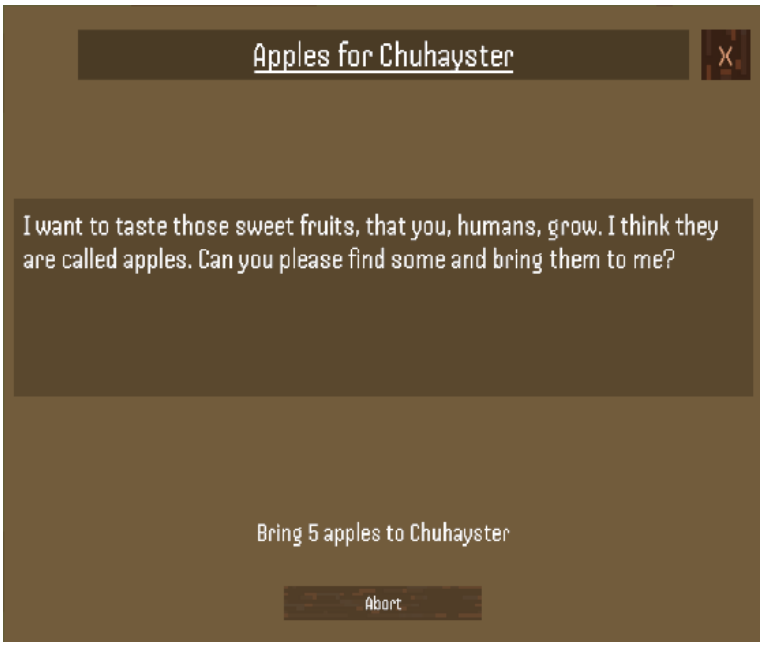


Figure 21 – Quest in progress

If you interact with a trader NPC, you will see a trading window (fig. 22). There, you can buy items from a trader by tapping on the item and then clicking the "Buy" button or sell your items by tapping on the item and then clicking the "Sell" button. The cost of items is listed in yellow, and the amount is listed in white. You can see your money listed near the "Inventory" title.

Figure 22 – Trading window

DEVELOPMENT PROCESS DESCRIPTION

There are 9 members in our team: four Unity Developers, three WEB Developers and 2 Designers. Roles were distributed voluntarily, each member has his own functions, duties and branches of work.

Work Distribution

Since there were three Unity Developers, all tasks on game development (except for design) were divided in three branches: Main character, Lore and Map Generation. Tasks were

divided equally according to individual field of work and abilities. When tasks intersected, a general meeting was called and the solution was created and implemented. Usually, it is fairly simple to solve using Git tools, but when those are unable to help, Team Lead has to manually solve every conflict before proceeding to merge branches.

Methodology

At the first meeting the team agreed to use one of the best methodologies for game development: Scrum. Since Jira Software was the most known for each developer, it was chosen as the main software for management. Jira Software has great features for managing documentation, sprints and tasks, including dividing tickets between developers, tagging each ticket with a type and development stages management.

Sprint planning

Our team agreed upon setting the duration of each sprint to a week. In the end of each sprint team members gathered for retrospective meetings, where everyone discussed what they achieved in this sprint, what could not be achieved and why. Also suggestions were considered to alternate the workflow for the next sprint to improve general development process. After each retrospective members discussed and set up tasks for the next sprint. Each member was assigned with a considerable amount of work, if anything was there to stop developers, it was solved together. After the meeting scrum master started the next sprint.

Reflexion

Each member learned to work in a team as one unit, considering others‘ troubles, helping and managing any problems. When a member wanted to make a suggestion, everyone else listened and considered this suggestion, and, in the future development, adapted to it. Everything was documented in a retrospective. This project has a great volume of work, and our progress was achieved only because of the respect to each others effort and time.

CONCLUSION

In conclusion, the journey of developing our team project has been a rewarding and transformative experience. From the inception of the idea to the final stages of implementation, our team demonstrated unparalleled dedication, creativity, and collaboration. Through meticulous research and a deep dive into Ukrainian folklore, we not only gained a profound understanding of the rich cultural heritage but also successfully translated these elements into an engaging and immersive gaming experience. Our commitment to authenticity ensured that the game not only entertained but also served as a cultural ambassador, sharing the beauty of Ukrainian traditions with a global audience.

The development process was not without its challenges, but each obstacle served as an opportunity for growth and innovation. Our team's resilience and ability to adapt were key factors in overcoming hurdles and turning them into stepping stones toward a more refined and polished end product.

Collaboration played a pivotal role in the success of our project. The diverse skills and perspectives brought by each team member contributed to the game's holistic development. The synergy among artists, writers, programmers, and designers resulted in a harmonious blend of creativity and functionality.

As we reflect on the culmination of our efforts, we can confidently say that we have not only created a game but a piece of art that pays homage to the cultural richness of Ukraine. The game stands as a testament to the potential of teamwork, innovation, and passion when harnessed toward a common goal. Looking ahead, we envision the game becoming a bridge between generations, fostering an appreciation for Ukrainian folklore among players of all ages. We also anticipate potential expansions and updates to further enrich the gaming experience and keep the project dynamic and relevant.

In essence, the development of our team project has been a remarkable journey that goes beyond the realms of game development. It is a celebration of culture, teamwork, and the limitless possibilities that arise when a group of dedicated individuals comes together to bring a collective vision to life. The legacy of our game will endure as a testament to the power of creativity and collaboration in the realm of interactive storytelling.

REFERENCES

1. Unity [Electronic resource] – Mode of access to the resource: https://unity.com/.
2. C# Documentation [Electronic resource] – Mode of access to the resource: https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/.
3. PHP [Electronic resource] – Mode of access to the resource: https://www.php.net/.
4. "The Witcher" game [Electronic resource] – Режим доступу до ресурсу: https://www.thewitcher.com/us/en/.
5. Video game industry - Statistics & Facts [Electronic resource] – Mode of access to the resource: https://www.statista.com/topics/868/video-games/#topicOverview.
6. The majority plays, and teenagers \"kill time\" in this way: Ukrainians were asked in detail about video games (study) [Electronic resource] – Mode of access to the resource: https://www.unian.ua/games/bilshist-ukrajinciv-graye-u-videoigri-shchob-vidklyuchitisya-vid-realnosti-igri-11326055.html.
7. Dungeons & Dragons [Electronic resource] – Mode of access to the resource: https://dnd.wizards.com/.
8. The game "Bastion" [Electronic resource] – Mode of access to the resource: https://store.steampowered.com/app/107100/Bastion
9. The game "Children of Morta" [Electronic resource] – Mode of access to the resource: https://store.steampowered.com/app/330020/Children_of_Morta/
10. The game "Hades" [Electronic resource] – Mode of access to the resource: https://store.steampowered.com/app/1145360/Hades/
11. A* Pathfinding Project [Electronic resource] – Mode of access to the resource: https://arongranberg.com/astar/
12. Jira [Electronic resource] – Mode of access to the resource: https://www.atlassian.com/software/jira
13. Law of Ukraine "On digital content and digital services" [Electronic resource] – Mode of access to the resource: https://zakon.rada.gov.ua/laws/show/3321-20#Text

ПРОЄКТ "Є-ВІДПОЧИНОК"

Данило Кримлов, Тарас Корольчук, Олександр Литвин, Максим Тюльпа, Єлизавета Складанна, Христина Ричок, Катерина Симоненко

АНОТАЦІЯ

"Tyrell" – це інноваційний проєкт, призначений для оптимізації процесів організації відпочинку, що враховує індивідуальні переваги користувачів. Проєкт акцентує увагу на персоналізації та автоматизації управління відпочинковими ресурсами. Платформа включає мобільний застосунок, який не тільки дозволяє користувачам з легкістю управляти своїми планами, але й забезпечує адміністраторам потужний інструментарій для контент-менеджменту.

Основні механіки застосунку включають збір і аналіз даних, щоб надати користувачам оптимальний підбір місць для відпочинку, базуючись на їхніх уподобаннях. Використання експертних систем забезпечує адаптивні рекомендації, які постійно вдосконалюються залежно від виборів та відгуків користувачів.

Поєднання професійного розвитку команди і впровадження інноваційних рішень відображає основну мету проєкту: створення системи, що забезпечує максимально персоналізований та задовільний досвід відпочинку для кожного користувача.

ANNOTATION

"Tyrell" is an innovative project designed to optimize the processes of recreation organization, taking into account the individual preferences of users. The project focuses on the personalization and automation of the management of recreational resources. The platform includes a mobile application that not only allows users to easily manage their plans, but also provides administrators with a powerful toolkit for content management.

The core mechanics of the app include data collection and analysis to provide users with the optimal selection of vacation spots based on their preferences. The use of expert systems provides adaptive recommendations that are constantly improved based on user choices and feedback.

The combination of the professional development of the team and the implementation of innovative solutions reflects the main goal of the project: the creation of a system that provides the most personalized and satisfying recreation experience for each user.

ANDROID, IOS, ЕКСПЕРТНА СИСТЕМА, ОПИТУВАННЯ, ВІДПОЧИНОК
ANDROID, IOS, EXPERT SYSTEM, SURWAY, VACATION

ВСТУП

Сучасний світ вимагає від технологій бути на передньому краї адаптації до швидкозмінних умов та індивідуальних потреб користувачів. Відповідно до цих вимог, проєкт "Tyrell" був розроблений з метою створення ефективної системи для організації відпочинку, яка б враховувала персоналізовані переваги користувачів. Цей проєкт є актуальним, оскільки забезпечує інноваційний підхід до управління відпочинковими ресурсами і відповідає сучасним вимогам ринку туристичних послуг, що потребує індивідуалізації та автоматизації процесів.

Постановка задачі. Проєкт "Tyrell" ставить за мету розробку високофункціональної системи, яка могла б виконувати комплексний збір та обробку даних для оптимального підбору місць відпочинку згідно з індивідуальними запитами

користувачів. Система повинна включати зручний адміністративний інтерфейс для управління контентом, мобільний застосунок й вебсайт продукту

Методи дослідження:

Для досягнення поставлених цілей у проєкті застосовуються методи аналізу даних, проєктування програмного забезпечення та тестування системи. Особлива увага приділяється використанню експертних систем для адаптації рекомендацій до потреб користувачів.

Цілі та завдання:

Головною ціллю проєкту "Tyrell" є створення системи, здатної ефективно адаптуватися до різних типів відпочинку та надавати персоналізовані рекомендації. Завданням проєкту є впровадження сучасного технологічного стеку, забезпечення високих стандартів безпеки та розробка інтуїтивно зрозумілого користувацького інтерфейсу. На професійному рівні, проєкт дозволить закріпити та поглибити знання у сфері технологій розробки вебзастосунків, а також ознайомити команду з методами обробки даних та їх застосуванням у реальних умовах.

ПОШУК ТА ВАЛІДАЦІЯ ІДЕЇ

Опис пошуку ідеї продукту

Нозглянуто сфери IT-продуктів: Fintech, Healthtech, Edtech, Traveltech, Transtech. Перелік обговорюваних ідей:

- застосунок для відстеження фінансів
- робот-консультант
- застосунок для донатів
- соцмережа волонтерів
- єБлагодійність
- програма для навчання масажу
- програма для консультацій по прийому вітамінів
- відгуки на лікарів
- збірник рецептів (з цінами на продукти, вартістю страви, можливістю отримати варіант страви на базі продуктів які в тебе є)
- помічники по навчанню, університет в одному місці
- календар-планер для команди
- вебсайт рейтинг університів від вступників
- оновлений тритон
- інтерактивна географічна карта
- продати застосунок-макет ресторану з меню і балами
- інтерактивна карта закладів з можливістю розпланувати день
- єЇжа
- єАвто

Було виділено три основні ідеї:

1. єБлагодійність. Благодійний застосунок, з можливістю перегляду самих свіжих і відомих перевірених зборів коштів для армії чи постраждалим людям у війні. З можливістю створювати благодійні оголошення про надання послуг чи товарів на безкоштовній основі для окремих категорій людей: військових чи постраждалих.

2. єВідпочинок. Карта ресторанів в обраному регіоні з можливостями сортування і прокладання маршруту до нього, ресторани модеруються, і потім викладаються власниками в застосунку з фото, меню, розташуванням і тд. Власники ресторанів платять місячну ставку для розміщення в списку. Планер дня пропонує автоматичний розпорядок місць для відпочинку зі списку, або користувач може заповнити його самостійно. Серед місць для відпочинку можуть бут: парки, ресторани, музеї тощо.

3. єАвто. Фріланс платформа для водіїв, і користувачів для надання послуг з перевезення або ж отримання таких послуг шляхом розміщення оголошень наявних маршрутів (регулярних і ні) або замовлень на потрібні користувачеві маршрути. Розміщення оголошень водіями здійснюється за валюту (всередині програми) яка безкоштовно нараховується раз в якийсь період, якщо валюта закінчилась, то її можна придбати за гроші.

Після ретельного розгляду кожної з цих ідей було прийняте рішення створити власний продукт **єВідпочинок** у сфері Traveltech.

Огляд наявних на ринку ІТ-продуктів – конкурентів.

1. **To-Do List - Schedule Planner** (непрямий конкурент)

Позиціювання продукту: To-Do List [1] - Schedule Planner надає користувачам зручний інструмент для організації їхнього часу та завдань.

Ключові функції: можливість користувачам планувати свій день, створювати списки справ, і відстежувати їх виконання.

Потреби: To-Do List - Schedule Planner вирішує потребу користувачів у плануванні та організації часу, допомагаючи їм бути більш продуктивними та організованими.

Цільова аудиторія продукту: Продукт призначений для широкого кола користувачів, включаючи студентів, професіоналів, домогосподарок, і всіх, хто потребує ефективно керувати своїм часом та завданнями.

Географія: весь світ.

Мовні локалізації: англомовні користувачі.

Модель монетизації: Продукт може монетизуватися через рекламу, платну версію з додатковими функціями, або підписку для доступу до преміум-функцій.

Наявність застосунку: так.

Наявність застосунку для Android: так.

Наявність застосунку для iOS: так.

Наявність вебплатформи: ні.

Кількість користувачів: 10 мільйонів завантажень на платформах App Store та Google Play.

Конкуренти: Основними конкурентами To-Do List - Schedule Planner є додатки, такі як Todoist, Microsoft To-Do, Any.do, та інші подібні інструменти для планування завдань.

Переваги:

- Простота використання: Інтерфейс додатка є інтуїтивним та зручним для користувача, що робить його легкодоступним для великої аудиторії.
- Організація завдань: З допомогою To-Do List - Schedule Planner користувачі можуть легко планувати свій день та створювати списки справ.
- Доступність на різних платформах: Додаток доступний як на мобільних пристроях, так і на вебплатформах, що дозволяє користувачам синхронізувати свої завдання між пристроями.

Недоліки:

- Відсутність додаткових функцій: To-Do List - Schedule Planner може бути обмежений для користувачів, які шукають розширені можливості планування та організації.
- Відсутність соціальних функцій: Додаток не надає можливостей спільної роботи над завданнями для груп користувачів.

2. **Glovo** (непрямий конкурент)

Позиціювання продукту: Glovo [2]- це сервіс доставки їжі та товарів, який надає зручність і швидкість замовлення для користувачів.

Ключові функції: замовлення їжі та товарів.

Потреби: Glovo відповідає потребам користувачів у зручній доставці продуктів і послуг без необхідності виходити з дому.

Цільові аудиторії продукту: Продукт спрямований на користувачів, які шукають швидку та зручну доставку їжі, товарів і послуг.

Географія: Іспанія, Португалія, Італія, Словенія, Хорватія, Республіка Польща, Україна, Молдова, Боснія і Герцеговина, Сербія, Чорногорія, Болгарія, Грузія, Киргизстан, Казахстан, Марокко, Уганда, Кенія, Гана, Кот-д'Івуар, Нігерія, Туніс, Румунія і Андорра.

Мовні локалізації: відповідно до регіону.

Модель монетизації: Головний спосіб монетизації Glovo - це комісія з кожного замовлення та співпраця з партнерами.

Наявність застосунку: так.

Наявність застосунку для Android: так.

Наявність застосунку для iOS: так.

Наявність вебплатформи: так.

Кількість користувачів: понад 5 млн активних користувачів.

Конкуренти: Основними конкурентами Glovo є інші служби доставки, такі як Bolt Food, DoorDash, та Postmates.

Переваги:

- Розширений вибір закладів: Glovo пропонує широкий вибір ресторанів і магазинів для замовлення їжі і товарів.
- Зручна навігація: Додаток пропонує зручну навігацію до місця призначення, що допомагає користувачам отримувати замовлення швидко.
- Рейтинги та відгуки: Glovo надає відгуки та рейтинги для ресторанів і магазинів, що допомагає користувачам робити інформовані вибори.

Недоліки:

- Високі витрати доставки: Деякі користувачі можуть вважати вартість доставки Glovo досить високою.
- Залежність від доступності: Glovo доступний не в усіх регіонах, що обмежує доступність сервісу для деяких користувачів.

3. **Google Maps** (з функцією пошуку закладів харчування та культурних закладів - непрямий конкурент)

Позиціювання продукту: Google Maps [3] є потужним інструментом для знаходження та дізнавання про ресторани, кафе, музеї, галереї та інші заклади харчування і культурні місця в будь-якому регіоні. Користувачі можуть легко знаходити відгуки, години роботи, та навіть бронювати столи.

Ключові функції: дізнавання про ресторани, кафе, музеї, галереї та інші заклади харчування і культурні місця в будь-якому регіоні.

Цільові аудиторії продукту: Ця функція Google Maps привертає як мандрівників, що шукають ресторани та культурні місця у новому місті, так і місцевих жителів, які хочуть дізнатися більше про заклади у своєму регіоні.

Потреби: Google Maps забезпечує користувачів актуальною інформацією про розташування, рейтинги, меню та години роботи ресторанів і культурних закладів, допомагаючи їм приймати поінформовані рішення.

Географія: понад 150 країн світу.

Мовні локалізації: відповідно до регіону.

Модель монетизації:Google Maps монетизується через рекламу та можливості співпраці з підприємствами, які можуть платити за просування своїх закладів на платформі.

Наявність застосунку: так.

Наявність застосунку для Android: так.

Наявність застосунку для iOS: так.

Наявність вебплатформи: так.

Кількість користувачів: 1 мільярд користувачів щомісяця.

Конкуренти: Основними конкурентами функції пошуку закладів харчування та культурних закладів в Google Maps є Yelp, Foursquare, і деякі інші сервіси, спеціалізовані на відгуках та рекомендаціях.

Переваги:

- Велика база даних: Google Maps має велику кількість даних про ресторани, музеї, галереї і інші заклади, що робить його дуже інформативним.
- Навігація в режимі реального часу: Google Maps надає навігацію в режимі реального часу, враховуючи трафік і швидкість руху, що допомагає користувачам швидко дістатися до свого місця призначення.
- Інтеграція з іншими Google-сервісами: Додаток інтегрований з іншими Google-сервісами, такими як Google Reviews та Google Photos.

Недоліки:

- Відсутність системи рекомендацій: Відсутність системи персоналізованих рекомендацій, що може обмежити користувальницький досвід.
- Залежність від інтернет-з'єднання: Для використання Google Maps потрібне активне інтернет-з'єднання, що може бути незручним в областях з поганим покриттям.

Мапа продукту

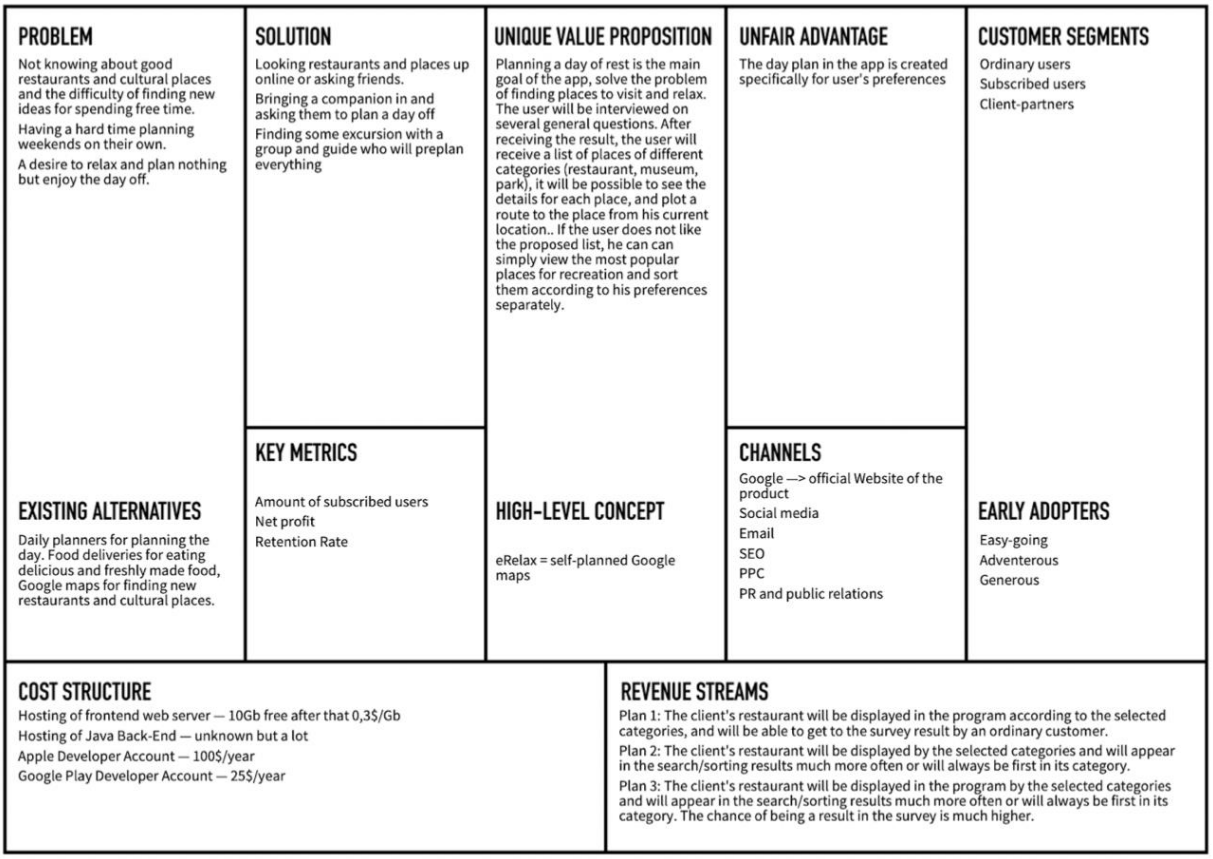


Рисунок 1 – Мапа продукту

Сильні та слабкі сторони розроблюваного продукту

Strong:

При правильній рекламній компанії дуже перспективний і прибутковий продукт, і зручний в користуванні, як приклад план дня для відпочинку без великих зусиль моніторингу наявних ресторанів і музеїв.

Weak:

Потребує реклами, без правильної маркетинг-стратегії запустити продукт майже нереально.

Opportunities:

Можливість переглядати карту ресторанів, музеїв, парків і загалом місць для відпочинку в обраному регіоні України. Можливість спланувати свій день відпочинку автоматично, або мануально.

Threats:

При слабкій модерації ресторанів які розміщуються на платформі є ризик просувати неякісні і погані заклади (з поганими санітарними умовами та якістю обслуговування)

Огляд використаних технологій

Інструментарій (технології проєктування) включає draw.io, Microsoft Word, Microsoft Excel, Microsoft PowerPoint, Canva.

Інструментарій (технології розробки) включає Git, BitBucket, Java, JUnit, Spring Framework, MongoDB, Docker, Kotlin, Jetpack Compose, Vue.js, Vuetify, Cypress, Firebase, HTML/CSS/JS, Figma, Swift/SwiftUI, Android SDK.

Призначення і цілі створення системи

Призначення системи

Система "Tyrell" призначена для комплексного збору, обробки та аналізу даних в контексті організації відпочинку. Основна мета системи - надати користувачам зручний інструмент для вибору оптимальних місць для відпочинку на основі їхніх переваг та потреб. Система охоплює різноманітні аспекти відпочинку, включаючи категорії відпочинкових зон та індивідуальні запитання користувачів.

Ключовим аспектом системи є її здатність адаптуватися до різних типів відпочинку та надавати персоналізовані рекомендації. Це досягається завдяки використанню експертної системи, яка враховує велику кількість параметрів.

Система "Tyrell" також включає адміністративний інтерфейс для управління базою даних відпочинкових місць. Це дозволяє легко та ефективно управляти контентом, забезпечуючи актуальність інформації.

Враховуючи цілі та призначення системи, "Tyrell" спрямована на забезпечення якісного, інтуїтивно зрозумілого та ефективного сервісу для організації індивідуального відпочинку.

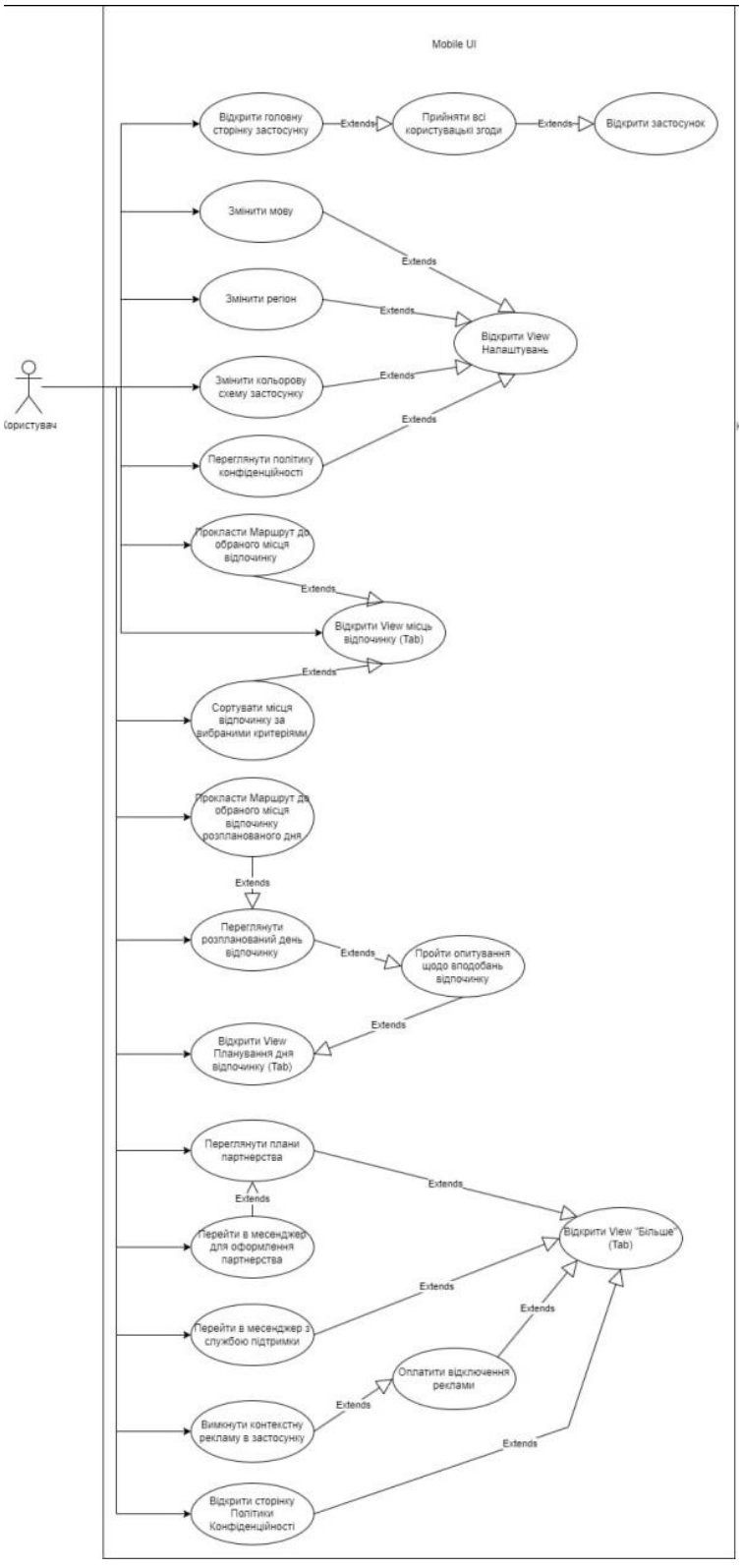


Рисунок 2 – Use-case діаграма

Цілі створення системи

Система "єВідпочинок" створюється з метою:

1. **Забезпечення збору, обробки та аналізу інформації місця відпочинку**: Ця ціль охоплює збір детальної інформації про різні локації для відпочинку, включаючи їхні

особливості: популярність, доступність, та відгуки користувачів. Система аналізує ці дані для створення точних і корисних рекомендацій для користувачів.

2. **Підвищення якості планування відпочинку**: Система допомагатиме користувачам планувати свій відпочинок, забезпечуючи пропозиції, які відповідають їхнім індивідуальним перевагам та потребам. Це включає вибір локацій, допомогу в організації маршрутів відпочинку, та надання рекомендацій щодо подій та активностей.

ВИМОГИ ДО СИСТЕМИ

Вимоги до системи в цілому

Проєкт "Tyrell" передбачає розробку застосунку "єВідпочинок" з використанням комплексного та сучасного технологічного стеку, враховуючи важливі аспекти функціонування системи та її безпеки. Ось ключові вимоги:

1. **Технологічний стек**: Система має бути розроблена з використанням Java/Spring для бекенду та Vue/Firebase для фронтенду.
2. **Хостинг та система контролю версій**: Система повинна бути розміщена на платформі BitBucket, з доступом через SSH-ключі та використанням Git для управління версіями і Pull Requests для злиття змін. Система Front-End має бути доступною в браузері через свій власний URL, і знаходитись на хостингу Google Firebase.
3. **Архітектура сервера**: Сервер має бути реалізований на основі архітектури REST.
4. **Інтерактивність з користувачем**: Система повинна надавати можливість вибору регіону та забезпечувати функціонал пошуку та сортування місць відпочинку.
5. **Персоналізація**: Можливість планувати день відпочинку на основі відповідей користувача у опитуванні.
6. **Реклама**: Інтеграція вбудованої реклами в список місць відпочинку.
7. **Адміністрування**: Система повинна містити панель адміністрування для управління даними.
8. **Безпека та Надійність**: Захист від DDOS атак (наприклад, через Captcha) та забезпечення стабільної роботи сервера та клієнтської частини.
9. **Масштабованість та Гнучкість**: Легкість додавання нових функцій, опитувань або категорій.
10. **Інтерфейс Користувача**: Інтерфейс повинен бути інтуїтивно зрозумілим та привабливим, відповідаючи сучасним стандартам UX/UI дизайну.
11. **Додаткові сторінки**: Наявність інформаційних сторінок (про команду, контакти, політику конфіденційності, опис продукту).
12. **Швидкість роботи**: Система має бути оптимізована для швидкої роботи навіть при великій кількості користувачів.
13. **Безпека даних**: Забезпечення конфіденційності та безпеки даних користувачів.

Ці вимоги охоплюють основні аспекти розробки та експлуатації вебзастосунку, забезпечуючи його функціональність, безпеку, користувацький досвід і масштабованість.

Вимоги до структури та функціонування системи

Основні Вимоги до Структури

Система "єВідпочинок " повинна бути реалізована як комплексне веб рішення, що забезпечує високу працездатність, безпеку, інтуїтивний інтерфейс та гнучкість у використанні. В Системі передбачається виділити наступні функціональні підсистеми:

1. **Адміністративна підсистема**: Призначена для керування контентом, даними користувачів та налаштуваннями системи. Ця підсистема має включати інструменти для

додавання, редагування та видалення даних з бази даних, а також механізми валідації для запобігання введення некоректних даних.

2. **Підсистема користувача**: Забезпечує функціонал, необхідний для планування дня відпочинку. Включає інтерактивний вибір регіону, перегляд та пошук місць відпочинку, персоналізовані рекомендації на основі відповідей в опитуванні та можливість перегляду та коригування плану дня.

Вимоги до функцій, які виконуються системою представлені в таблицях 1-2.

Підсистема адміністратора

Таблиця 1. Перелік функцій, задач що підлягають автоматизації

Функція	**Задача**
Перегляд категорій та пошук за назвою	Розробка інтерфейсу для відображення всіх категорій у системі.
	Інтеграція функціонала пошуку, щоб адміністратори могли легко знаходити категорії за ключовими словами або назвами.
Додавання категорій	Створення форми для введення інформації про нові категорії, включаючи назву, опис, зображення та інші релевантні поля.
	Розробка процесу перевірки введених даних на відповідність заданим критеріям перед їх збереженням у системі.
	Впровадження механізму видалення, який дозволяє адміністраторам легко видалити категорію з бази даних з відповідним підтвердженням дії.
Видалення категорій	Впровадження механізму видалення, який дозволяє адміністраторам легко видалити категорію з бази даних з відповідним підтвердженням дії.
Редагування окремих полів категорій	Розробка інтерфейсу для редагування існуючих категорій, включаючи оновлення інформації, такої як час роботи, адреса, контактні дані тощо.
	Забезпечення валідації оновлених даних для запобігання введенню помилкової інформації.

Підсистема користувача

Таблиця 2. Перелік функцій, задач що підлягають автоматизації

Функція	**Задача**
Планування Дня Відпочинку	Опитування користувачів для збору переваг
	Генерація персоналізованих рекомендацій
	Відображення списку планів дня з можливістю коригування
Експорт плану дня відпочинку	Створення формату експорту (наприклад, PDF)
	Надання можливості поділитися планом з іншими
Пошук та сортування місць відпочинку	Реалізація пошукової системи за назвами та іншими критеріями
	Сортування місць за вибраними критеріями
Обрання регіону	Вибір користувачем регіону для відпочинку
	Відображення інформації про обране місце відпочинку
Відкрити сторінку програми в AppStore/Google Play	Перенаправлення користувача на відповідну сторінку магазину додатків
	Відображення інформації про програму

Функція	Задача
Відкрити інформаційні сторінки (Про нас, Політика конфіденційності, Як користуватись, Допомога)	Надання інформації про команду та проєкт
	Пояснення політики захисту даних користувачів
	Інструкції щодо використання системи
	Надання контактної інформації для підтримки
Перегляд планів партнерства	Надання інформації про умови партнерства
	Відображення переваг для партнерів
Формування та оплата партнерства	Надання форми для оформлення партнерства
	Інтеграція платіжних систем для оплати партнерства
Підтримка користувачів	Перехід у месенджер для спілкування зі службою підтримки
	Надання допомоги користувачам у вирішенні проблем

Системні вимоги

Вебзастосунок має належним чином відображатися в інтернет-браузерах MS Internet Explorer версії 5.5 і вище, Mozilla версії 1.7 і вище, та Opera версії 7.54 і вище.

При помилкових діях користувача, таких як введення некоректних даних, пропуск обов'язкових полів у формах та інших ситуаціях, які система може обробити, будуть виводитися відповідні повідомлення про помилки українською мовою, які інтегровані у загальний дизайн вебзастосунку.

Цільовий рівень Android API: 34

Цільовий рівень IOS SDK: 16

Джерелом даних для Системи повинна бути NoSQL база даних Firebase Firestore.

Опис організації інформаційної бази (табл. 3-6, рис.3).

Логічна структура бази даних

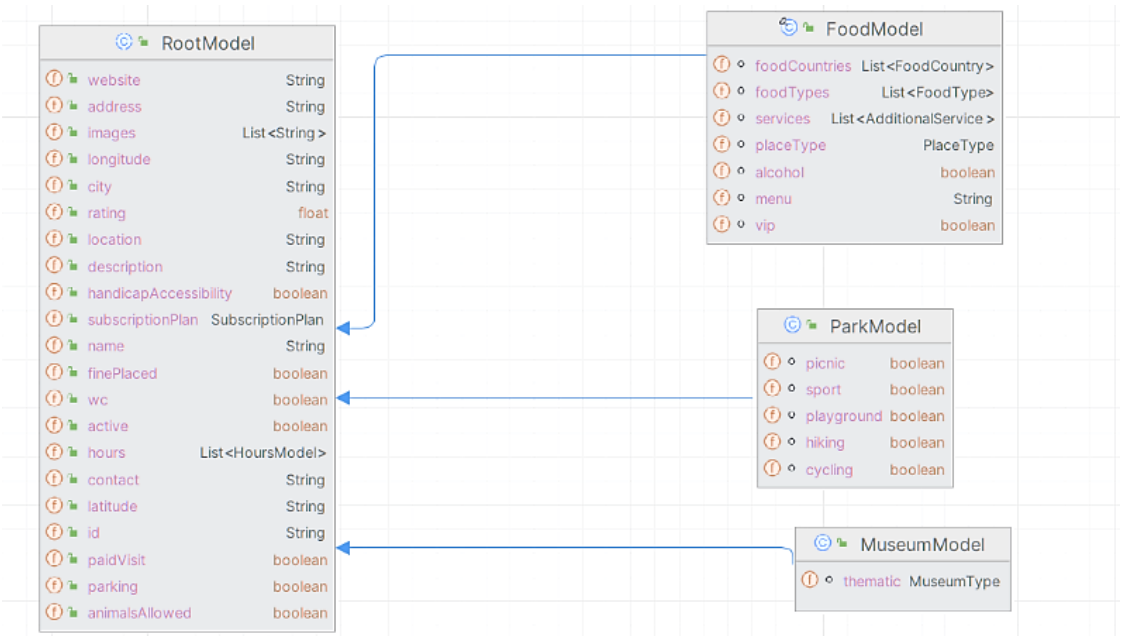


Рисунок 3 – Схематика взаємодії класів системи Back-End

Таблиця 3. Перелік таблиць в сховищі Firestore:

Номер	**Таблиця**	**Опис**
1	foodplace	Категорія для ресторанів, кафе та інших закладів харчування.
2	museumplace	Категорія для музеїв та галерей.
3	parkplace	Категорія для парків, садів та інших зелених зон для відпочинку.

Таблиця 4. Опис таблиці foodplace

Таблиця foodplace Атрибут	**Тип**	**Опис**
id	String	Ідентифікатор
name	String	Назва закладу
description	String	Опис закладу
rating	float	Рейтинг закладу
website	String	Посилання на вебсайт закладу
address	String	Адреса закладу
contact	String	Телефон закладу
parking	boolean	Чи є зручна парковка
animalsAllowed	boolean	Чи дозволено приходити з тваринами
paidVisit	boolean	Чи платний вхід
wc	boolean	Чи є туалет
finePlaced	boolean	Чи зручне розташування
handicap Accessibility	boolean	Чи доступно для людей з інвалідністю
hours	List<HoursModel>	Часи роботи закладу
images	List<String> images	Фотографії закладу
location	String	Посилання на координати на Google Maps
longitude	String	Довгота
latitude	String	Широта
city	String	Місто, в якому розташований заклад
subscriptionPlan	SubscriptionPlan	План підписки
active	boolean	Чи активна підписка
menu	String	Посилання на меню закладу
placeType	PlaceType	Тип закладу
foodCountries	List<FoodCountry>	Країни кухонь, страви з яких є в меню
foodTypes	List<FoodType>	Які типи їжі є в меню
services	List<Service>	Які додаткові сервіси пропонує заклад
alcohol	boolean	Чи дозволено алкоголь
vip	boolean	Чи є заклад елітним

Таблиця 5. Опис таблиці museumplace

Таблиця museumplace Атрибут	**Тип**	**Опис**
id	String	Ідентифікатор
name	String	Назва закладу
description	String	Опис закладу
rating	float	Рейтинг закладу
website	String	Посилання на вебсайт закладу
address	String	Адреса закладу
contact	String	Телефон закладу
parking	boolean	Чи є зручна парковка
animalsAllowed	boolean	Чи дозволено приходити з тваринами
paidVisit	boolean	Чи платний вхід
wc	boolean	Чи є туалет
finePlaced	boolean	Чи зручне розташування
handicapAccessibility	boolean	Чи доступно для людей з інвалідністю
hours	List<HoursModel>	Часи роботи закладу
images	List<String>	Фотографії закладу
location	String	Посилання на координати на Google Maps
longitude	String	Довгота
latitude	String	Широта
city	String	Місто, в якому розташований заклад
subscriptionPlan	SubscriptionPlan	План підписки
active	boolean	Чи активна підписка
thematic	MuseumType	Тематика музею

Таблиця 6. Опис таблиці parkplace

Таблиця parkplace Атрибут	**Тип**	**Опис**
id	String	Ідентифікатор
name	String	Назва закладу
description	String	Опис закладу
rating	float	Рейтинг закладу
website	String	Посилання на вебсайт закладу
address	String	Адреса закладу
contact	String	Телефон закладу
parking	boolean	Чи є зручна парковка
animalsAllowed	boolean	Чи дозволено приходити з тваринами
paidVisit	boolean	Чи платний вхід
wc	boolean	Чи є туалет
finePlaced	boolean	Чи зручне розташування
handicapAccessibility	boolean	Чи доступно для людей з інвалідністю
hours	List<HoursModel>	Часи роботи закладу
images	List<String>	Фотографії закладу
location	String	Посилання на координати на Google Maps

Таблиця parkplace Атрибут	**Тип**	**Опис**
longitude	String	Довгота
latitude	String	Широта
city	String	Місто, в якому розташований заклад
subscriptionPlan	SubscriptionPlan	План підписки
active	boolean	Чи активна підписка
playground	boolean	Чи є дитячий майданчик
picnic	boolean	Чи можна влаштувати пікнік
hiking	boolean	Чи підходить для піших прогулянок
sport	boolean	Чи є знаряддя для занять спортом
cycling	boolean	Чи можна орендувати/кататись на велосипеді

UI/UX-дизайн продукту представлено на рисунках 4-8.

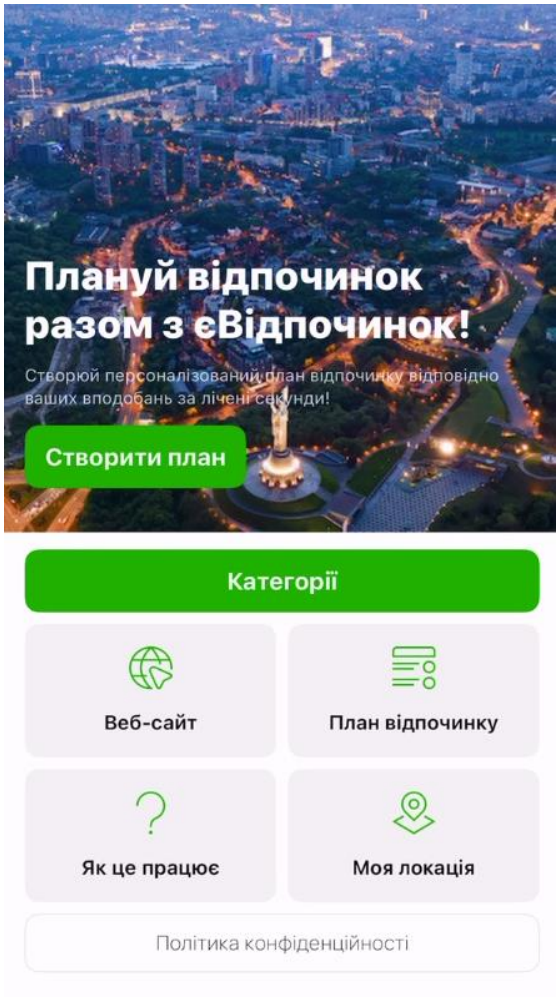


Рисунок 4 – Головна сторінка мобільного застосунку "єВідпочинок"

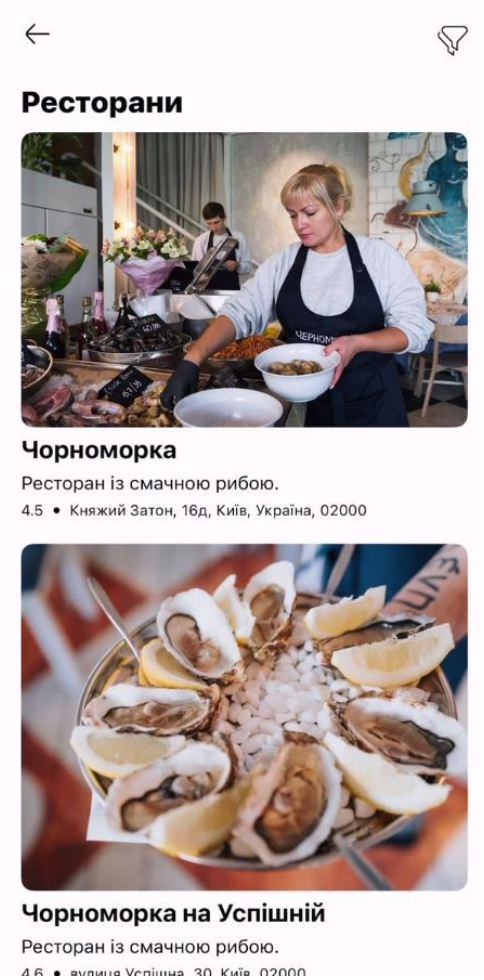


Рисунок 5 – Перегляд всіх ресторанів у мобільному застосунку "єВідпочинок" з обраним м.Київ

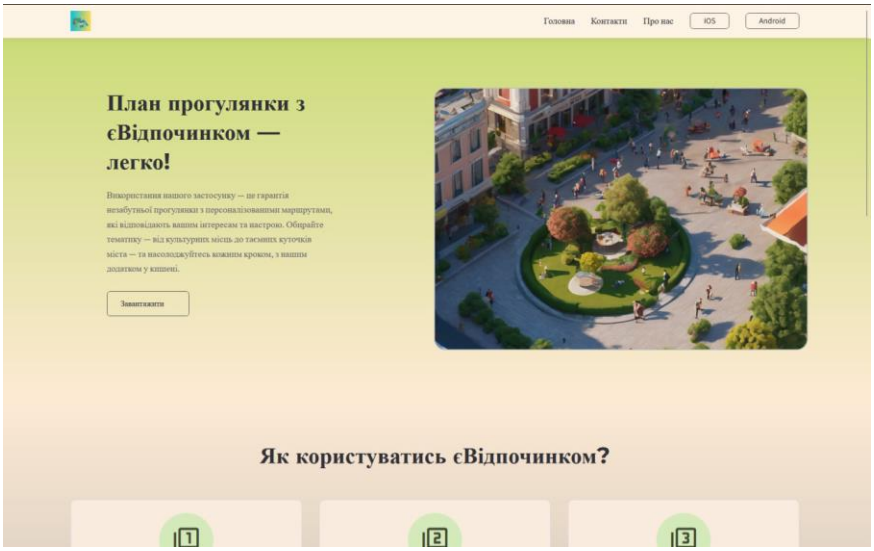


Рисунок 6 – Головна сторінка вебсайту продукту

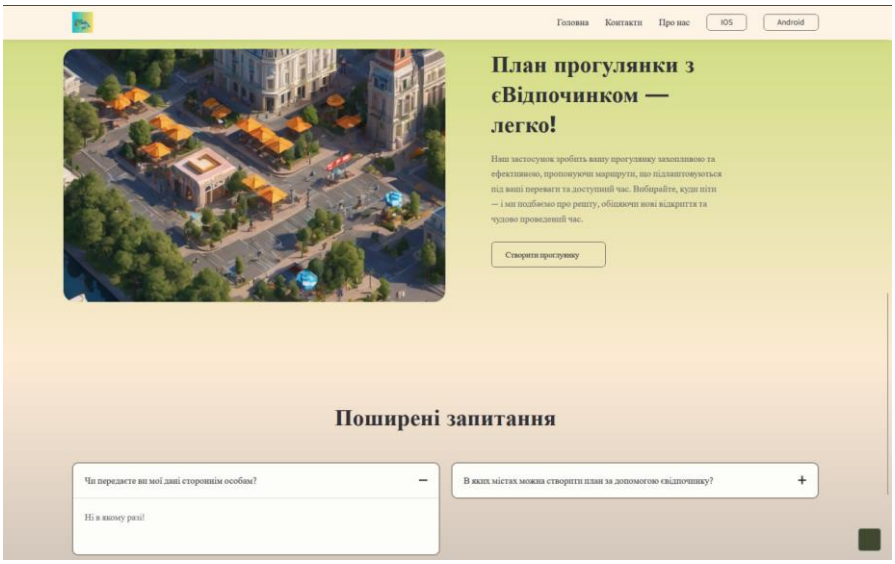


Рисунок 7 – Головна сторінка вебсайту продукту

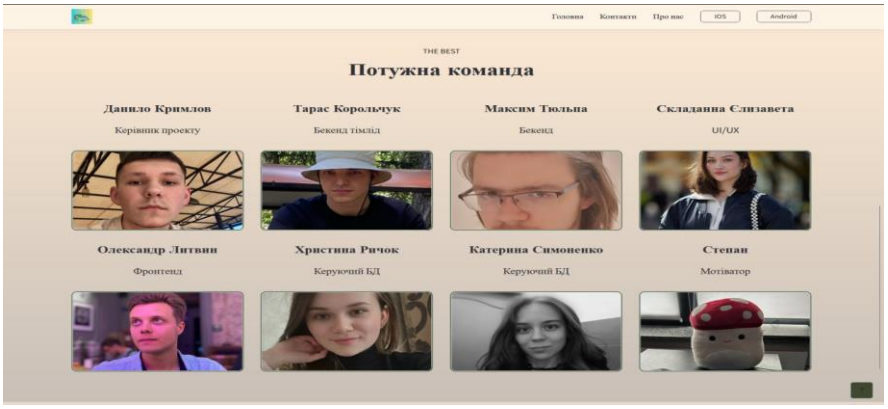


Рисунок 8 – Сторінка "Про нас" вебсайту продукту

На основі скриншотів інтерфейсу системи "єВідпочинок" можна виділити кілька основних принципів та законів UX-дизайну, які були застосовані при її розробці:

1. **Закон Фіттса (Fitts's Law)**:

○ Закон Фіттса стверджує, що час, необхідний для швидкого переміщення до цільової області, зменшується, коли розмір цільової області збільшується та коли відстань до неї зменшується. На головній сторінці вебсайту кнопки "Створити план" та "Шукати заклад" є великими та легко натискними, що знижує час, потрібний для взаємодії з цими елементами.

2. **Закон Джекоба (Jakob's Law)**:

○ Закон Джекоба вказує на те, що користувачі віддають перевагу вебсайтам, які виглядають та функціонують подібно до тих, з якими вони вже знайомі. В категорії "Ресторани" використовується звичний шаблон списку з картками, що містять фотографію та короткий опис, це полегшує розуміння та навігацію для користувачів.

3. **Принцип простоти (Principle of Simplicity)**:

○ Принцип простоти заснований на ідеї, що простіше розуміння системи веде до кращого досвіду користувача. На скриншотах з системи ми бачимо чистий дизайн з легким для сприйняття шрифтом і достатньою кількістю білого простору, що дозволяє

користувачу легко зосередитися на завданнях, таких як планування дня або вибір місць для відпочинку.

4. **Принцип консистентності (Principle of Consistency)**:

○ Принцип консистентності підкреслює важливість узгодженості в інтерфейсі користувача. Всі сторінки "єВідпочинок" мають єдиний стиль навігації, з однаковими шрифтами, кольорами та розміщенням елементів. Це допомагає користувачам швидше орієнтуватися на вебсайті.

5. **Принцип пріоритету (Principle of Priority)**:

○ Принцип пріоритету вказує на необхідність ранжування елементів на сторінці за їх важливістю. Наприклад, на сторінці планування дня відпочинку важливі елементи, такі як "Переглядати план" та "Шукати заклад", виділені як великі кнопки, що привертають увагу користувача.

Застосування цих законів та принципів UX-дизайну сприяє створенню інтуїтивно зрозумілого, ефективного та приємного користувацького досвіду, що спонукає користувачів частіше та охочіше користуватися системою "єВідпочинок".

ІМПЛЕМЕНТАЦІЯ ПРОДУКТУ

Продукт "єВідпочинок" — це вебзастосунок, призначений для планування особистого дозвілля користувачів. Система реалізована з використанням мікросервісної архітектури, що забезпечує гнучкість у розширенні та надійність у функціонуванні. Використання такої архітектури дозволяє окремо масштабувати кожен сервіс залежно від потреб та навантаження.

Основні компоненти застосунку включають:

1. Лендінг сайт (Frontend):

○ Розроблений з використанням фреймворку Vue.js, який є прогресивним JavaScript фреймворком для створення інтерфейсів користувача.

○ Інтегрує Vuetify, UI бібліотеку з великою кількістю готових до використання компонентів, які відповідають Material Design від Google. Це забезпечує зручність та естетичність інтерфейсу, а також прискорює розробку завдяки наявності готових шаблонів і компонентів.

○ Адаптивний дизайн, що забезпечує коректне відображення сайту на різних пристроях, від мобільних телефонів до настільних комп'ютерів.

2. **Клієнтська частина (Mobile):**

○ Для Android додаток розроблений з використанням мови програмування Kotlin та бібліотеки Jetpack Compose, яка дозволяє створювати декларативні UI інтерфейси, що спрощує розробку і підвищує реактивність додатків. Android SDK забезпечує інтеграцію з апаратними та програмними можливостями пристроїв.

○ Для iOS додаток розроблений з використанням мови Swift, що забезпечує високу продуктивність і безпеку. Використання SwiftUI дозволяє ефективно створювати сучасні та динамічні інтерфейси з використанням декларативного підходу в програмуванні. Це надає можливість легко адаптувати дизайн під різні розміри та особливості пристроїв Apple, включаючи iPhone та iPad.

3. **Серверна частина (Backend):**

○ Побудована на базі Spring Boot, потужного інструменту для створення мікросервісів з використанням Java.

○ Spring Boot забезпечує швидку конфігурацію та запуск сервісів, використовуючи вбудовані рішення для таких завдань, як управління залежностями, конфігурація безпеки, підключення до баз даних тощо.

○ Пропонує API ендпойнти для різних функцій додатку, таких як аутентифікація користувачів, управління даними закладів та створення планів відпочинку.

4. **База даних:**

○ Використовує Firestore, NoSQL базу даних від Firebase, що забезпечує реальний час синхронізації даних та їх збереження у хмарі.

○ Firestore дозволяє легко масштабувати додаток без необхідності ускладнення інфраструктури, а також забезпечує потужні можливості запитів та сортування даних.

○ База даних оптимізована для високої продуктивності та ефективної взаємодії з додатком, що дозволяє швидко відображати та оновлювати інформацію на стороні користувача.

5. **Система адміністрування:**

○ Являє собою захищений інтерфейс, призначений для управління контентом і користувацькими даними.

○ Доступ до системи управління БД обмежений внаслідок авторизації адміністраторів

○ Забезпечує інструменти для створення, редагування та видалення записів в базі даних, таких як список закладів, користувачів та відгуків.

Розробка та імплементація "єВідпочинок" виконана з врахуванням сучасних практик та стандартів вебдизайну та розробки, що дозволяє створювати ефективні та надійні рішення для кінцевих користувачів.

ТЕСТУВАННЯ

Інтеграційне тестування (рис. 9-12).

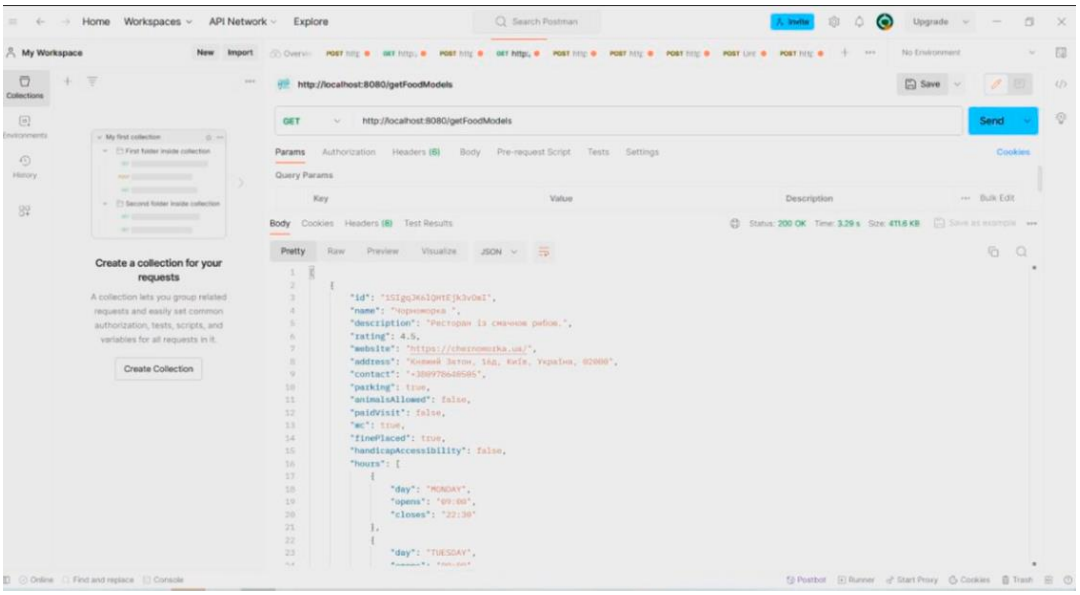


Рисунок 9 – Приклад тестування контролеру getFoodModels

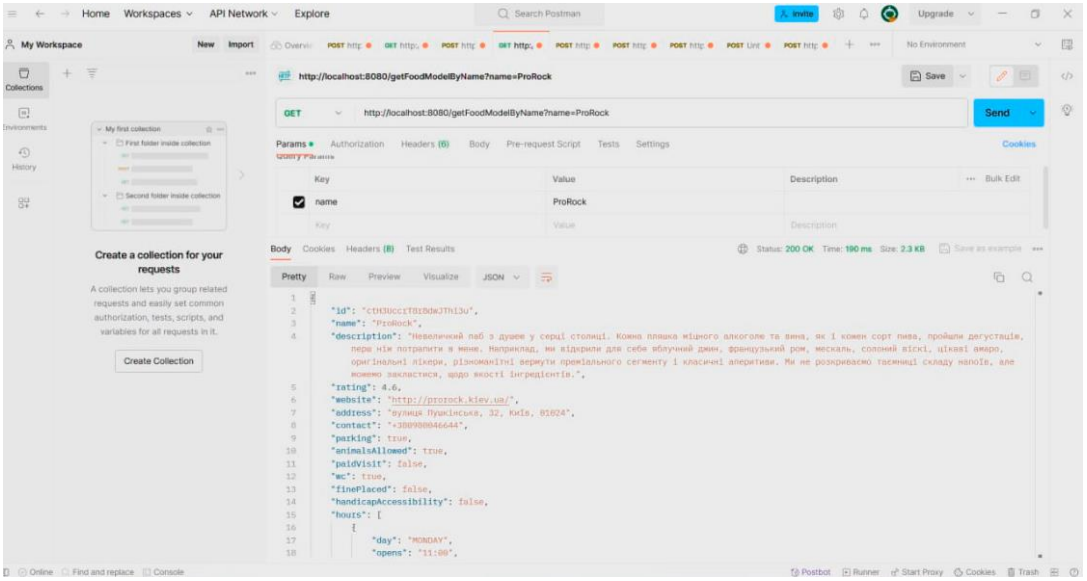
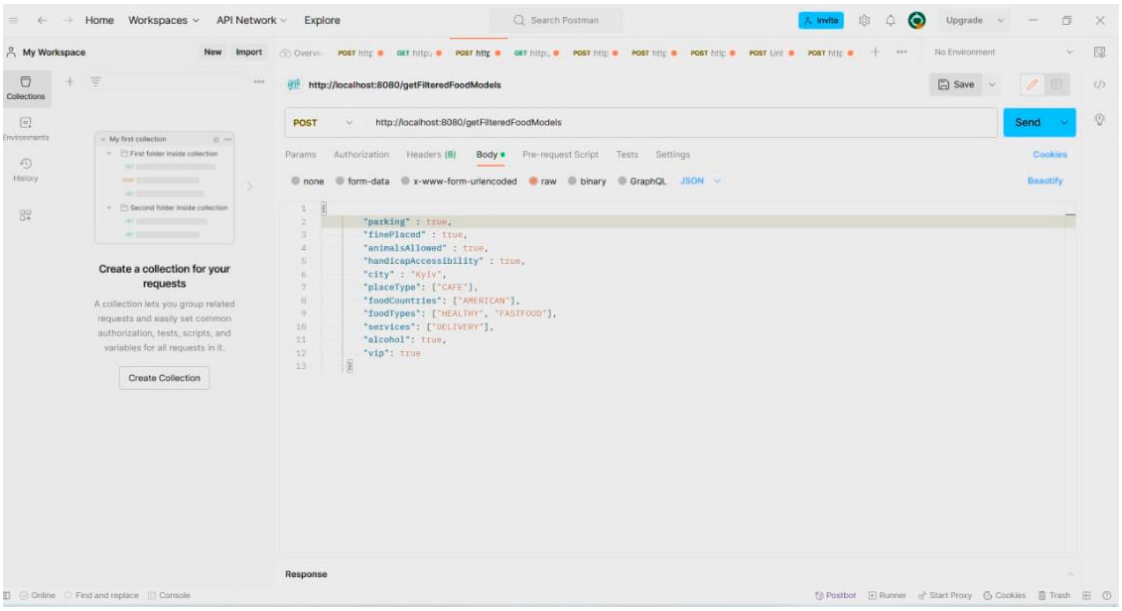


Рисунок 11 – Приклад тестування контролеру getFilteredFoodModels (приклад запиту)



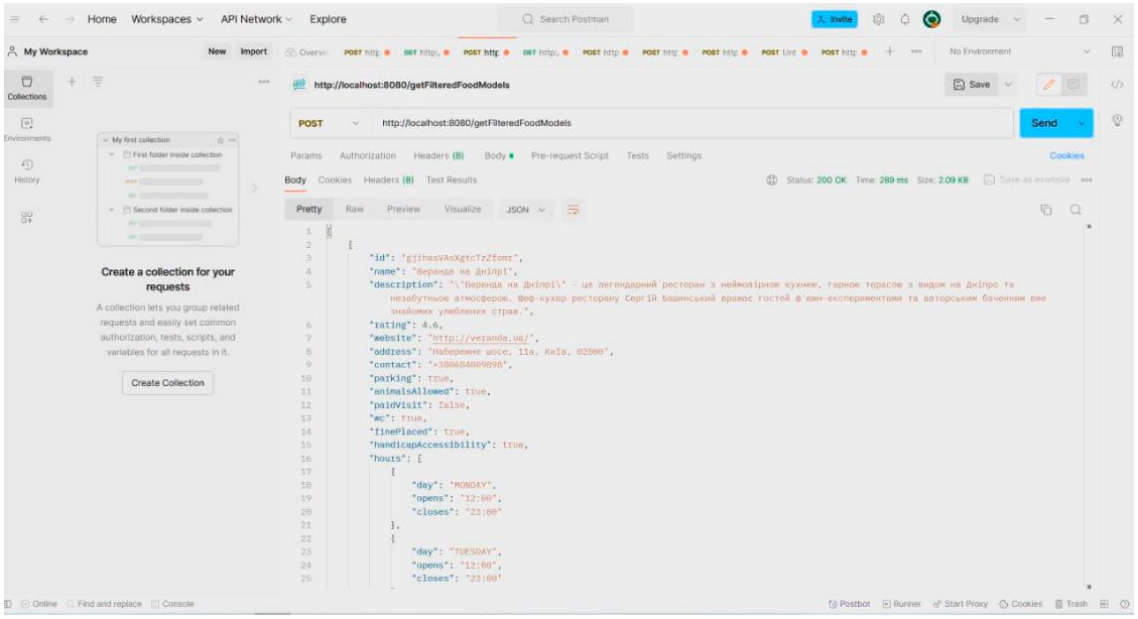


Рисунок 12 – Приклад тестування контролеру getFilteredFoodModels (приклад відповіді сервера)

Інтеграційне тестування "єВідпочинок" виконується для забезпечення сумісності та правильної взаємодії між фронтендом (реалізованим на Vue.js і Vuetify) та бекендом (Spring Boot), а також між бекендом і обраною нами хмарною базою даних Firestore.

Використовуючи Postman, команда розробників може здійснювати тестові запити до API для перевірки відповідей сервера на різноманітні сценарії використання. Наприклад, можна відправити GET-запит до ендпойнту /getFoodModels, щоб переконатися, що система повертає коректний список закладів харчування. Аналогічно, можна тестувати POST-запити для додавання нових закладів харчування, використовуючи ендпойнт /addFoodModel. Ці запити допомагають перевірити, що дані відправляються і приймаються правильно, і що бізнес-логіка на сервері працює належним чином.

Системне тестування

Системне тестування об'єднує всі компоненти системи "єВідпочинок" і перевіряє їх спільну роботу з метою виявлення будь-яких невідповідностей між взаємодією різних частин системи. Цей вид тестування дозволяє перевірити, чи відповідає система загальним вимогам та, чи готова до використання кінцевими користувачами.

У контексті системного тестування з Postman, можна виконувати послідовність запитів, які імітують реальні користувацькі сценарії, наприклад, реєстрацію нового користувача, створення плану відпочинку, додавання закладу до системи, редагування інформації про заклад та видалення закладу.

Всі згадані вище тести важливі для виявлення проблем на ранніх етапах розробки, забезпечуючи високу якість продукту до його релізу.

ДОКУМЕНТАЦІЯ ПРОЄКТУ

Обрана модель документації це Документ (*Document*) формату docx.

В процесі створення продукту "єВідпочинок" було створено документи форматів docx, drawio, drawio.svg, pdf, xlsx, jpg, fig, які знаходяться за посиланням на Google Drive - Project Tyrell.

Зовнішньою документацією для користувачів, партнерів, наглядових органів можуть бути презентації "проєкт Tyrell ідеї", "проєкт Tyrell єВідпочинок", "проєкт Tyrell конкуренти", "проєкт Tyrell прототип дизайну єВідпочинку", "проєкт Tyrell аналітика продуктивності спринти 4-7", "WhereToGo", "ЧеклистSCRUM_1", "ЧеклистSCRUM_2", "ЧеклистSCRUM_3", "ЧеклистSCRUM_4", "ЧеклистSCRUM_5", "ЧеклистSCRUM_6", "ЧеклистSCRUM_7", "ЧеклистSCRUM_8", "Вимоги".

ОПИС ПРОЦЕСУ РОЗРОБКИ

Команда (табл. 7):

Danylo Krymlov, 4th year of bachelor`s degree, project leader, frontend developer, mobile developer for IOS.

Taras Korolchuk, 4th year of bachelor`s degree, backend developer, tester.

Maxym Tulpa, 4th year of bachelor`s degree, backend developer, mobile developer for Android.

Elyzaveta Skladanna, 4th year of bachelor`s degree, moderator, UI/UX, content-creator.

Olexander Lytvyn, 4th year of bachelor`s degree, frontend developer, documentation technician.

Chrystyna Rychok, 2nd year of master`s degree, manager of data collection and data base.

Kateryna Symonenko, 2nd year of master`s degree, manager of data collection and data base.

Таблиця 7. Розподіл ролей в команді

Учасник	**Роль учасника на початку проєкту**
1. Данило Кримлов	керівник проєкту, фронтенд, мобільний розробник (IOS)
2. Тарас Корольчук	бекенд тімлід, тестувальник
3. Максим Тюльпа	бекенд, мобільний розробник (Android)
4. Єлизавета Складанна	модератор, UI/UX, Content-creator
5. Олександр Литвин	фронтенд, фахівець з документації
6. Христина Ричок (географ)	менеджер зі збору інформації та бази даних
7. Катерина Симоненко (географ)	менеджер зі збору інформації та бази даних

Опис обраної методології, інструментарію:

Методологія та система управління проєктами SCRUM.

Інструментарій (технології проєктування): draw.io, Microsoft Word, Microsoft Excel, Microsoft PowerPoint, Canva.

Інструментарій (технології розробки): Git, BitBucket, Java, JUnit, Spring Framework, MongoDB, Docker, Kotlin, Jetpack Compose, Vue.js, Vuetify, Cypress, Firebase, HTML/CSS/JS, Figma, Swift/SwiftUI, Android SDK.

Для розробки під мобільні платформи було обрано Swift/SwiftUI та Kotlin, Jetpack Compose, Android SDK, оскільки це домінуючі технології на ринку, а інші не надають такої ж кількості можливостей.

Для фронтенду було обрано Vue та Vuetify, оскільки Vue - це легкий і гнучкий фреймворк, який добре підходить для поетапної розробки, а Vuetify забезпечує готові компоненти, що спрощують процес.

Для бекенду вибір зупинився на Java Spring Framework і Firebase, оскільки вони надають потужні інструменти для масштабування і безпеки, а також вбудовані служби для швидкої розробки, відповідно.

Планування спринтів задокументовано в файлах "Беклог" і "Беклог 2сем" за посиланнями Беклог Беклог 2сем

Рефлексія:

На щастя, вдавалося зробити все заплановане.

Чого навчилися:

Організували роботу в команді, сформували та написати основні вимоги до проєкту, створили беклог проєкту. Зробили аналіз конкурентів. Фіналізувати та протестували версію системи управління БД та Front-End в рамках MVP. Додали 100 тестових місць закладів харчування, більше ніж 50 парків та музеїв. Створили звіт по MVP проєкту. Створили дизайн Front-End для початку розробки Front-End системи. Створили алгоритм для опитування.

ВИСНОВКИ

1. Система "Tyrell" ефективно інтегрує комплексний збір, обробку та Gloаналіз даних для організації відпочинку, враховуючи індивідуальні переваги користувачів.
2. Через адміністративний інтерфейс система забезпечує зручне управління контентом та актуальність інформації про місця відпочинку.
3. Використання експертної системи дозволяє адаптуватися до різних типів відпочинку та надавати персоналізовані рекомендації.
4. Проєкт "Tyrell" передбачає застосування сучасного технологічного стеку та високі стандарти безпеки, що є ключовими для надійної роботи вебзастосунку.
5. Структурна комплексність системи "єВідпочинок" та її підсистем забезпечує високу працездатність, безпеку та інтуїтивно зрозумілий користувацький інтерфейс.
6. Для досягнення більшої ефективності системи, слід розглянути можливість її масштабування, що дозволить легко додавати нові функції та категорії відпочинку.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. To-Do List - Shedule Planner [Електронний ресурс]: https://play.google.com/store/apps/details?id=todolist.scheduleplanner.dailyplanner.todo.reminders&hl=uk&gl=US
2. Glovo - Більше, ніж доставка їжі [Електронний ресурс]: https://play.google.com/store/search?q=glovo&c=apps&hl=uk&gl=US
3. Google maps [Електронний ресурс]: https://play.google.com/store/search?q=google%20maps&c=apps&hl=uk&gl=US

НОТАТКИ