МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ПРОЄКТУВАННЯ ВЕБСИСТЕМИ ДЛЯ ПІДТРИМКИ МЕНЕДЖМЕНТУ ТА УПРАВЛІННЯ ПЕРСОНАЛОМ

Спеціальність 122 «Комп'ютерні наук»

122 – КРБ – 401.220100705

Виконала студентка 4-го курсу, групи 401
_____ І. О. Шавріна
«19» червня 2024 р.

Керівник: канд. фіз.-мат. наук, доцент.
_____ І. В. Кулаковська
«19» червня 2024 р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«19» червня 2024 р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 401 факультету комп'ютерних наук Шавріній Ірині Олександрівні.

1. Тема кваліфікаційної роботи «Проєктування вебсистеми для підтримки менеджменту та управління персоналом».

Керівник роботи Кулаковська Інесса Василівна, канд. фіз.-мат. наук, доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «28» грудня 2023 р. № 217

2. Строк представлення кваліфікаційної роботи студентом «19» червня 2024 р.

3. Вхідні (початкові) дані до роботи: експертні оцінки програм за визначеними критеріями (функціональність, зручність використання); вимоги користувачів щодо функціоналу та інтерфейсу вебсистеми; технічні характеристики та обмеження обраних технологій для розробки.

Очікуваний результат: розроблена вебсистема для підтримки менеджменту та управління персоналом, яка включатиме систему управління завданнями, моніторинг виконання робіт та функції для комунікації між співробітниками

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

– сутність менеджменту та аналіз сучасних вебсистем для управління проєктами та контролю персоналу;

– технології та інструменти для розробки вебсистеми управління проєктами;

– опис вебсистеми управління проєктами та контролю персоналу.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: «Розробка заходів з поліпшення умов праці»

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Алексєєва А. О., доцент кафедри екології	

Керівник роботи __канд. фіз.-мат. наук, доцент Кулаковська І. В._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Шавріна І. О._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «14» січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Проєктування вебсистеми для підтримки менеджменту та управління персоналом

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників КРБ	10.11.2023	15.11.2023	Виконано
2	Отримання завдання на виконання КРБ	10.01.2024	15.01.2024	Виконано
3	Складання календарного плану роботи на весь період виконання КРБ	16.01.2024	30.01.2024	Виконано
4	Отримання завдання на переддипломну практику	15.04.2024	29.04.2024	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до КРБ	29.04.2024	11.05.2024	Виконано
6	Розробка звіту з переддипломної практики	12.05.2024	15.05.2024	Виконано
7	Виконання КРБ: аналіз сучасних вебсистем для підтримки менеджменту та розробка власного програмного продукту	13.05.2024	22.06.2024	Виконано
8	Перший попередній захист КРБ на засіданні комісії кафедри	27.05.2024	27.05.2024	Виконано
9	Доробка та остаточне оформлення КРБ	28.05.2024	09.06.2024	Виконано
10	Другий попередній захист КРБ на засіданні комісії кафедри	10.06.2024	10.06.2024	Виконано
11	Подання КРБ рецензенту	13.06.2024	13.06.2024	Виконано
11	Подання КРБ, її електронної копії та інших документів (відгуку, рецензії) до захисту	17.06.2024	21.06.2024	Виконано
12	Захист КРБ перед екзаменаційною комісією (ЕК)	24.06.2024	28.06.2024	Виконано

Розробив студент __Шавріна І. О._____ _____
(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи __канд. фіз.-мат. наук, доцент Кулаковська І. В. _____
(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«29» січня 2024 р.

АНОТАЦІЯ

кваліфікаційної роботи студентки групи 401 ЧНУ ім. Петра Могили
Шавріної Ірини Олександрівни
Тема: «Проєктування вебсистеми для підтримки менеджменту та управління персоналом»

Об’єктом дослідження є процеси проєктування та впровадження вебсистеми для підтримки менеджменту та управління персоналом.

Предметом дослідження є методи та технології, які використовуються для створення вебсистеми, що забезпечує автоматизацію та оптимізацію процесів управління проєктами та персоналом.

Мета роботи – розробка вебсистеми для підтримки менеджменту та управління персоналом, яка буде враховувати переваги та недоліки існуючих рішень на ринку і відповідати потребам користувачів.

Робота складається з фахового розділу і спеціальної частини з охорони праці. Пояснювальна записка складається зі вступу, трьох розділів та висновків.

У першому розділі було проведено аналіз сучасних вебсистем для управління проєктами та контролю персоналу, визначено переваги та недоліки деяких сучасних програмних продуктів для реалізації кращих функціональних рішень у власні вебсистемі.

У другому розділі проведено аналіз технології та інструменти для розробки системи управління проєктами.

У третьому розділі було змодельовано вебсистему управління проєктами та контролю персоналу, описано її зовнішній вигляд, функціонал та деякі етапи програмної реалізації.

В результаті розроблено вебсистему для підтримки менеджменту та управління персоналом до функціоналу якої входить можливість авторизація та реєстрації, створення різної кількості проєктів з канбан-дошками, чатами та можливістю управління контактами системи.

Бакалаврська кваліфікаційна робота містить __ сторінок, __ рисунків, __ таблиць, __ використаних джерел та __ додатків.

Ключові слова: вебсистема, менеджмент, Python, Django, Vite, Svelte, Tailwind.

ABSTRACT

To the qualification work by student of group 401 of

Petro Mohyla Black Sea National University

Shavrina Iryna

«Designing a web system for supporting management and personnel administration»

The object of research is the processes of designing and implementing a web system for supporting management and personnel administration.

The subject of the study is the methods and technologies used to create a web system that provides automation and optimization of project and personnel management processes.

The aim of this thesis is the development of a web system to support management and personnel management, which will take into account the advantages and disadvantages of existing solutions on the market and meet the needs of users.

The work consists of a professional section and a special part on labor protection. The explanatory note consists of an introduction, three sections and conclusions.

The first section analyzes modern web systems for project management and personnel control, and also the advantages and disadvantages of some modern software products were determined for the implementation of better functional solutions in their own web systems.

The second section analyzes technology and tools for project management system development.

In the third section, the web system of project management and personnel control was modeled, its appearance, functionality and some stages of software implementation were described.

As a result, a web system was developed to support management and personnel management, the functionality of which includes the possibility of authorization and registration, the creation of various projects with kanban boards, chats, the ability to save files, as well as the administration of system users.

Thesis contains: __ pages, __ figures, __ tables, __ formulas, __ references.

Keywords: web system, management, Python, Django, Vite, Svelte, Tailwind.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ООП	–	Об'єктно-орієнтоване програмування
API	–	Application Programming Interface
COVID-19	–	коронавірусна хвороба 2019 року
CSS	–	Cascading Style Sheets
DOM	–	Document Object Model
DRF	–	Django REST Framework
HTML	–	HyperText Markup Language
HTTP	–	HyperText Transfer Protocol
IT	–	Інформаційні технології
JWT	–	JSON Web Token
ORM	–	Object-Relational Mapping
PDF	–	Portable Document Format
PERT	–	Program Evaluation and Review Technique
POST	–	HTTP POST Method
REST	–	Representational State Transfer
SPA	–	Single Page Application
UI	–	User Interface
URL	–	Uniform Resource Locator

ВСТУП

Актуальність теми. В наш час швидкість розвитку технологій постійно зростає, ефективне управління роботи проєктів та підтримка персоналу стають ключовими факторами успіху для будь-якої організації чи підприємства. У сучасному світі ефективне управління персоналом та проєктами є ключовим фактором успіху будь-якої організації. З розвитком технологій, особливо в сфері інформаційних технологій, з'являються нові інструменти та платформи, що дозволяють автоматизувати та оптимізувати ці процеси.

Особливу актуальність ця тема набула в умовах пандемії COVID-19, яка значно змінила формат роботи більшості компаній, перевівши їх на віддалений або гібридний режим. Це створило додаткові виклики для менеджменту, оскільки потрібно було забезпечити ефективну комунікацію, контроль виконання завдань та управління персоналом в нових умовах. З метою відповіді на ці вимоги та покращення робочих процесів, було вирішено створити вебсистему для підтримки менеджменту та управління персоналом.

Актуальність даної роботи також обумовлена необхідністю впровадження ефективних інструментів для організації робочих процесів та взаємодії між учасниками команди.

Об'єкт: процеси проєктування та впровадження вебсистеми для підтримки менеджменту та управління персоналом.

Предмет: методи та технології, які використовуються для створення вебсистеми, що забезпечує автоматизацію та оптимізацію процесів управління проєктами та персоналом.

Мета: розробка вебсистеми для підтримки менеджменту та управління персоналом, яка буде враховувати переваги та недоліки існуючих рішень на ринку і відповідати потребам користувачів.

Основні завдання

Для досягнення поставленої мети необхідно вирішити такі завдання:

– визначитись з поняттям менеджменту та основними етапами роботи над проєктом;

– проаналізувати сучасний стан задачі управління проєктами та персоналом;

– вивчити існуючі аналоги систем та визначити їхні переваги і недоліки;

– проаналізувати сучасні технології розробки вебсистем та обрати необхідні для розробки власного проєкту;

– розробити архітектури системи та її компоненти;

– реалізувати функціонал системи відповідно до вимог користувачів.

Практичне значення: розроблена вебсистема призначена для використання в різних організаціях, де є потреба в ефективному управлінні проєктами та персоналом. Наприклад, університети та студентські стартапи можуть використовувати цю систему для керування своїми проєктами, розподілу завдань та спільної роботи над інноваційними ідеями. Вона забезпечить зручний інтерфейс для керування завданнями, моніторингу виконання робіт та комунікації між членами команди. Інтелектуальна система дозволить підвищити продуктивність роботи, зменшити витрати часу на адміністративні задачі та покращити якість управління персоналом та проєктами.

1 АНАЛІЗ СУЧАСНИХ ВЕБСИСТЕМ ДЛЯ УПРАВЛІННЯ ПРОЄКТАМИ ТА КОНТРОЛЮ ПЕРСОНАЛУ

1.1 Сутність проєктного менеджменту в інформаційних технологіях

Менеджмент в інформаційних технологіях — це процес управління інформаційними системами, технологіями та проєктами, пов'язаними з розробкою, впровадженням і підтримкою програмного забезпечення, апаратного забезпечення, мереж та інших IT-інфраструктур. Він вимагає поєднання технічних знань і управлінських навичок для ефективного керування технологічними ресурсами та забезпечення їхнього внеску в досягнення стратегічних цілей організації.

Головною метою менеджменту в інформаційних технологіях є повноцінне та злагоджене ведення проєкту та створення унікального за функціонального програмного забезпечення.

Проєкт — це тимчасова й унікальна спроба створити продукт, послугу або результат із визначеним початком і кінцем (зазвичай обмежений у часі та часто обмеженим фінансуванням або персоналом), що здійснюється для досягнення унікальних цілей і завдань, як правило, для досягнення корисні зміни або додана вартість[2].

Управління проєктом - це процес керівництва роботою команди для досягнення всіх цілей проєкту в рамках заданих обмежень[1]. Основна мета менеджменту в IT полягає в забезпеченні ефективної роботи інформаційних систем, підтримці їхньої безперервності, розвитку та відповідності бізнес-цілям організації.

Тож, враховуючи те, що в інформаційних технологіях розробка програмного продукту є кінцевим етапом ітеративного процесу роботи над проєктом групи фахівців, можемо зробити висновки, що менеджмент в інформаційних технологіях найчастіше представлено як проєктний менеджмент.

Кожний проєкт має свій життєвий цикл. Це послідовні групи робіт на які можна розбити весь процес створення кінцевого результату проєкту. Життєвий цикл проєкту має шість етапів, а саме, ініціалізація, планування, виконання, моніторинг та контроль та закриття.

Ініціалізація призначена для визначення основних параметрів проєкту та його доцільності. До основних завдань ініціалізації входять: формулювання ідеї проєкту, аналіз можливостей, ризиків, витрат та переваг проєкту, визначення ключових цілей і завдань, визначення зацікавлених сторін, розробка документ, що формально затверджує початок проєкту і визначає його межі, завдання та основні ресурси проєкту.

Головною метою планування є розробка детального плану реалізації проєкту. До основних завдань планування проєктом входять: розробка чіткого переліку завдань та етапів проєкту, встановлення термінів виконання завдань, розподіл ресурсів та визначення відповідальних осіб, розрахунок всіх витрат і формування фінансового плану, розробка плану управління ризиками, а саме, виявлення потенційних ризиків і розробка стратегій їхнього зменшення, розробка плану комунікацій, встановлення критеріїв якості і методів їхнього контролю.

Найдовшим та найоб’ємнішим за обсягом робіт під час розробки проєкту є етап його виконання. Метою цього етапу є реалізація запланованих завдань і досягнення цілей проєкту. До основних завдань можна віднести: розподіл завдань між членам команди, забезпечення ефективної співпраці між усіма учасниками проєкту, управління використанням ресурсів для забезпечення їхньої оптимальної ефективності, мотивація і підтримка команди, управління змінами, а саме, внесення коректив до плану проєкту у разі необхідності.

Постійний нагляд за ходом виконання проєкту для забезпечення його відповідності плану забезпечується за допомогою моніторингу та контролю. Основні завдання цього етапу розробки проєкту такі: регулярний аналіз виконання завдань і порівняння з планом, вимірювання ефективності, контроль якості, виявлення нових ризиків і впровадження заходів для їхнього зменшення,

підготовка звітів для зацікавлених сторін про стан проєкту, та внесення коректив до плану для усунення виявлених відхилень за необхідності.

Останнім етапом є закриття проєкту. Ого основна мета є своєчасне та вдале завершення проєкту і формалізація його результатів. До основних завдань цього етапу відносять: виконання всіх залишкових завдань і закриття відкритих питань., офіційна передача результатів проєкту замовнику або кінцевим користувачам, аналіз досягнутих результатів і порівняння з початковими цілями, вивчення уроків проєкту і документування найкращих практик для майбутніх проєктів, завершення всіх договірних зобов'язань і розрахунок з постачальниками, проведення оцінки роботи команди і надання зворотного зв'язку.

Кожна з цих фаз має важливе значення для успішного завершення проєкту і забезпечення досягнення поставлених цілей. Для забезпечення зручного та швидкого проходження командою фахівців всіх етапів створення проєкту було прийнято рішення щодо розробки системи для комунікації та зручної організації роботи над проєктом.

1.2 Системи для управління проєктами та контролю персоналу

Системи управління проєктами – це комплексне програмне забезпечення, що включає в себе програми для планування завдань, складання графіків, контролю вартості та управління бюджетом, розподілу ресурсів, спільної роботи, комунікації, швидкого управління, документування та адміністрування системи, які використовуються для управління великими проєктами.

Однією з ключових можливостей управління проєктами є можливість планування подій та керування завданнями. Вимоги до цього можуть різнитися залежно від інструментів, які використовуються. До найпоширеніших можливостей належать:

- планування послідовних подій та їх взаємозв'язків;
- розподіл робочого графіка співробітників та управління ресурсами;

– оцінка часу, необхідного на вирішення кожної конкретної задачі;
– впорядкування завдань в залежності від їхнього терміну виконання;
– одночасне керування кількома проєктами.

Програмне забезпечення для управління проєктами забезпечує доступ до різноманітної інформації, такої як:

– список завдань для співробітників та розподіл ресурсів;
– огляд термінів виконання завдань;
– попередження про можливі ризики, пов'язані з проєктом;
– інформація про робоче навантаження;
– статус проєкту, показники та їх прогнозування.

Розглянемо різновиди програмного забезпечення для управління проєктами.

1.2.1 Стаціонарне ПЗ

Такі програми зазвичай дозволяють зберігати інформацію у файлі, який може бути викладений у загальний доступ для інших користувачів, або ж дані можуть зберігатися у центральній базі даних. Деякі приклади цього типу програм:

– GanttProject: програма для планування проєктів на основі діаграм Ганта та діаграм типу PERT;
– Microsoft Project: програма управління проєктами, розроблена та продається корпорацією Microsoft;
– OpenProj: багатоплатформне програмне забезпечення для управління проєктами, розроблене компанією LibreOffice.

1.2.2 Вебдодатки

До вебдодатків можемо віднести:

– Worksection: український онлайн сервіс для управління проєктами, командної роботи та контролю за виконанням завдань;
– Trello: безкоштовна багатоплатформна система управління проєктами;

– Basecamp: онлайн-інструмент для управління проєктами, спільної роботи та постановки завдань, створений компанією 37signals;

– Pivotal Tracker: онлайн-система для управління процесами розробки проєктів та постановки завдань, створена компанією PivotalLabs;

– Bontq: вебдодаток для управління проєктами та відстеження помилок;

– Easy Projects.NET: багатофункціональна система управління проєктами.

1.2.3 Системи, розраховані на велику кількість користувачів

Ці системи зазвичай побудовані на базі технології клієнт-сервер та мають вебінтерфейс. До них відносяться:

– Jira: серверний багатоплатформний вебзастосунок для управління проєктами, відстежування помилок та організації спілкування з користувачами;

– Redmine: відкритий серверний багатоплатформний вебдодаток для керування проєктами та завданнями;

– Trac: серверний багатоплатформний вебдодаток для управління проєктами та відстеження помилок в програмному забезпеченні;

– Mantis Bug Tracker: відкритий серверний багатоплатформний вебдодаток для керування проєктами та відстеження помилок;

– GitHub: вебсервіс для спільної розробки програмного забезпечення на базі системи керування версіями файлів Git.

1.3 Огляд деяких систем управління та комунікації персоналу та їх функціонал

Для створення власної системи управління проєктами, було детально ознайомлено з деякими існуючими системами. Багато сучасних систем мають свій унікальній функціонал та можливості. Щоб створити зручний та корисний функціонал застосунку, треба дослідити вже створені сучасні технології і методики в програмному забезпеченні та перейняти кращий досвід для власної системи.

Далі розглянемо деякі популярні системи управління проєктами.

1.3.1 Інструмент Trello

Цей інструмент для організації робочих завдань працює на основі канбан-дошки, яка зазвичай має три колонки: "треба зробити", "в роботі" та "зроблено". Trello - це дуже простий інструмент. У ньому є дошка з різними списками завдань. Завдання представлені у вигляді карток із заголовками, прапорцями, коментарями та мітками, які розміщуються у списках відповідно до їхнього статусу: "заплановані", "в процесі виконання", "виконані" або інші категорії. Потім їх можна редагувати або перетягувати з одного списку до іншого.

Інтерфейс Trello дозволяє користувачам використовувати одну дошку для власної роботи або надавати доступ до іншої дошки з завданнями для спільного проєкту. Хоча сервіс є простим у використанні, він також має різноманітні функції, які роблять його надзвичайно зручним, такі як інтеграція з календарем, Slack, Google Drive, зв'язок з акаунтом GitHub та інші.

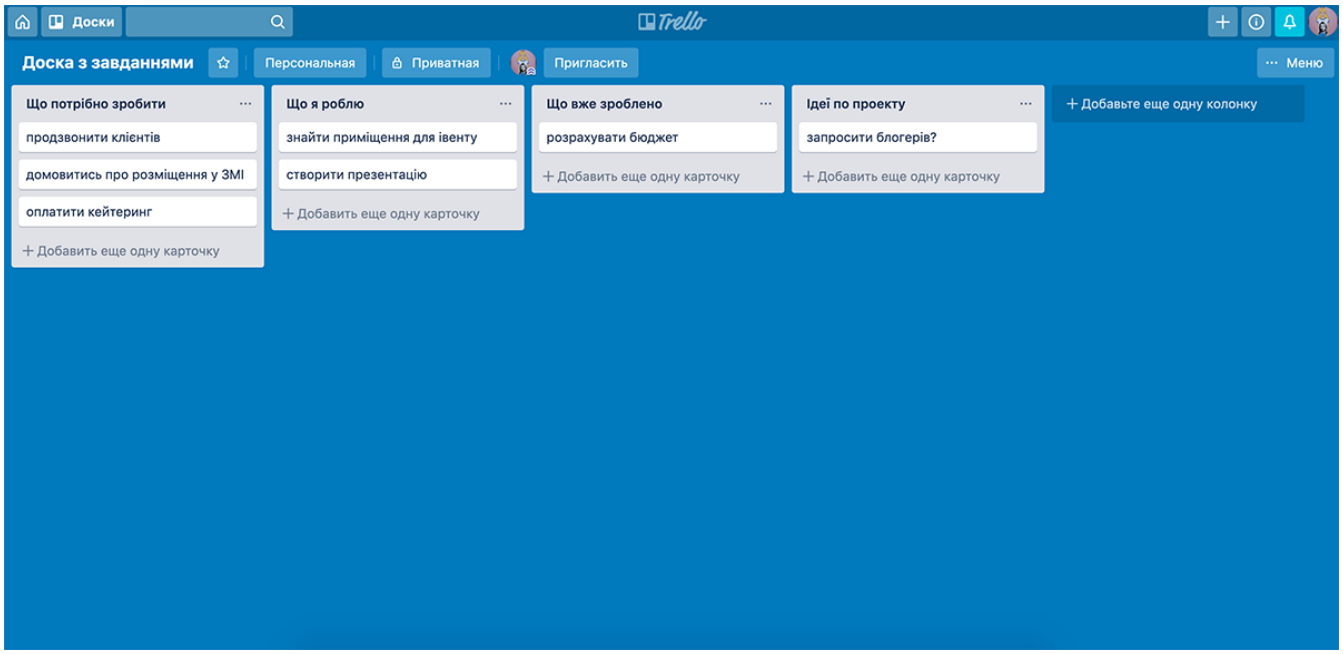

Рисунок 1.1 – Інтерфейс Trello

1.3.2 Інструмент для відстеження часу Harvest

Harvest - це інструмент для відстеження часу. Він дозволяє з'ясувати, скільки часу фактично витрачається на весь проєкт і його окремі етапи. Принцип роботи полягає у створенні списку завдань (з можливістю додавання запланованої оплати, якщо це необхідно), визначенні початку/паузи/завершення роботи. Сервіс надає докладну інформацію щодо прогресу, витрачених годин і оплати. Потім надається можливість експортувати дані в Excel.

Використання Harvest для відстеження часу, витраченого на роботу, не означає постійного контролю. Це зроблено з метою уникнення перевантаження команди та відстеження етапів проєкту, які забирають більше часу, для подальшої можливості оптимізувати процеси. Harvest також можна інтегрувати з іншими сервісами, наприклад, з Trello.

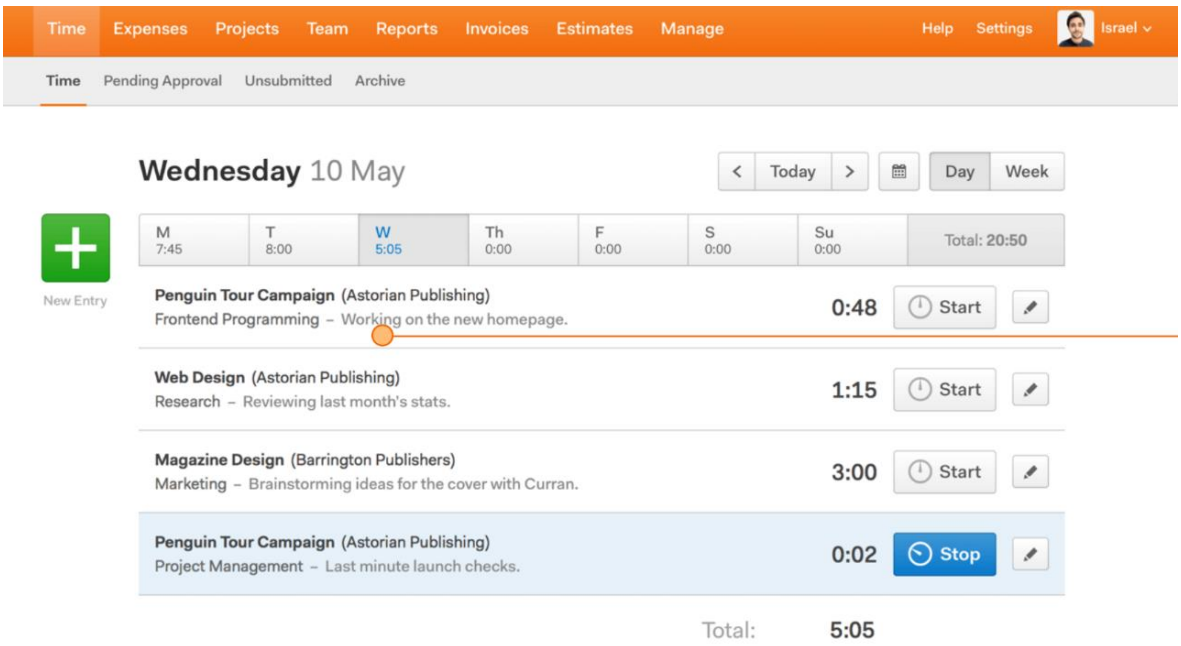

Рисунок 1.2 – Інтерфейс Harvest

1.3.3 Інструмент Asana

Asana - це інструмент для розподілу навантаження та відстеження прогресу роботи команди. Основні переваги включають можливість для колег бачити

розподіл завдань один серед одного та структурування завдань за допомогою канбан-дошок~~,~~ та календаря ~~та таймлайну~~.

Інтерфейс Asana також має можливість сформувати схему з переходами завдань між працівниками, наприклад, коли у рамках створення email- розсилки завдання переходить від копірайтера до верстальника. Однак слід зауважити, що інтерфейс програми та її функціонал більш складні, ніж в інших сервісах.

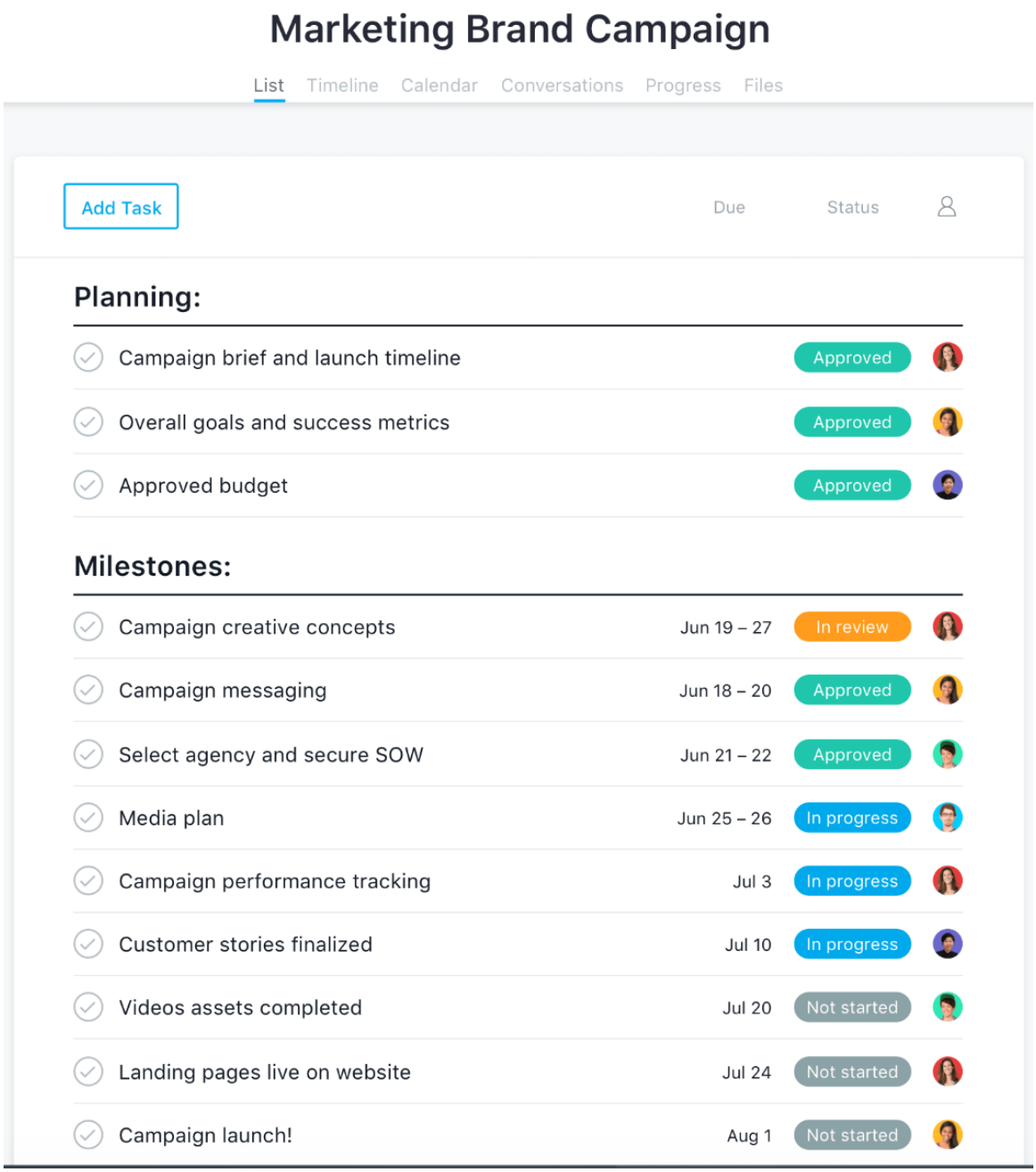

Рисунок 1.3 – Інтерфейс Asana

1.3.4 Веб-сервіс Ganttpro

GanttPRO - це веб-сервіс для створення діаграм Ганта, яка відображає відрізки часу та завдання на часовій шкалі. Однією з переваг GanttPRO є можливість розподілу роботи між колегами та встановлення зв’язків між різними етапами завдань.

Інтерфейс GanttPRO досить простий у використанні, але виникають деякі складнощі, пов'язані з особливостями самої діаграми Ганта. Наприклад, при строгих обмеженнях часу навіть невелика зміна дедлайну може вплинути на інші завдання або скоротити час на їх виконання. Крім того, графік проєкту можна експортувати у форматах PDF або Excel для друку або подальшої передачі замовнику.

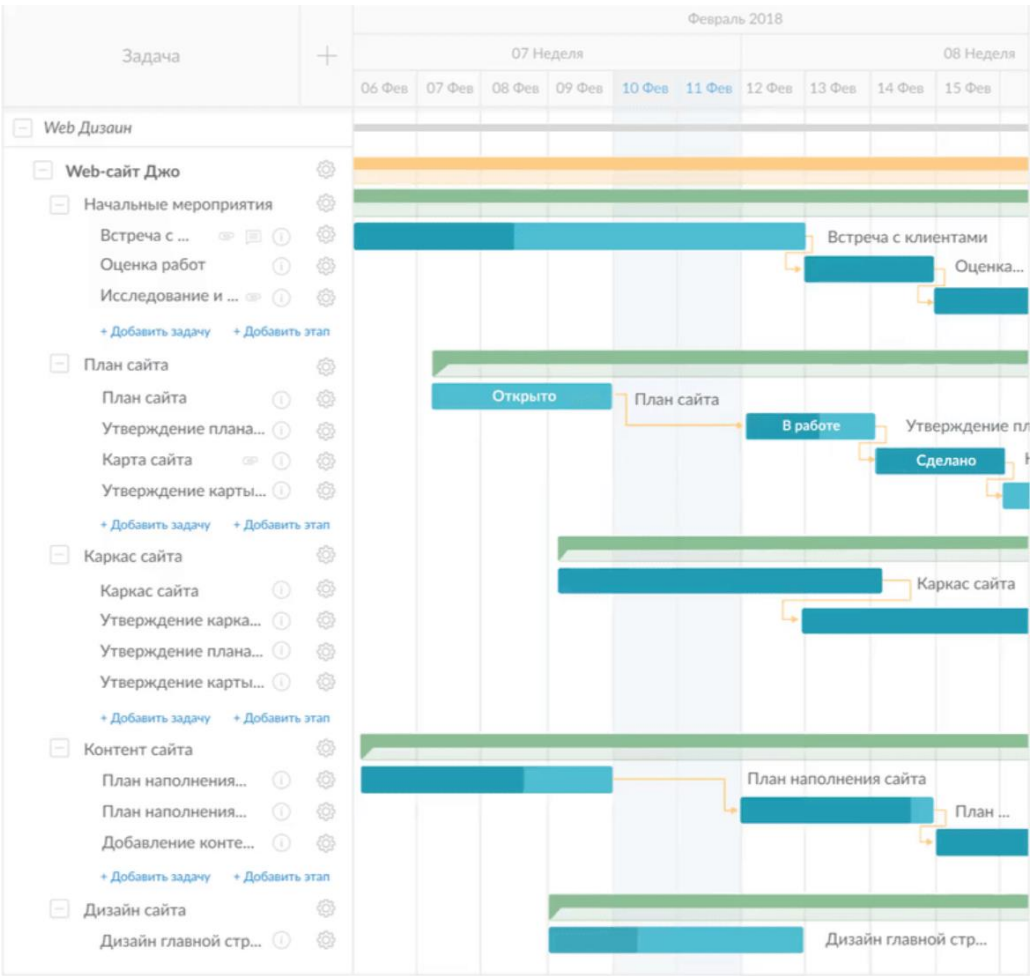

Рисунок 1.4 – Інтерфейс GanttPRO

1.3.5 Інструмент для комунікації Slack

Slack - це зручний інструмент для комунікації в команді, який дозволяє створювати окремі канали для різних тем. Наприклад, надає можливість відокремити робочі чати від інших обговорень.

Інтерфейс Slack дозволяє додавати файли та обговорювати їх, що дуже зручно для комунікації щодо прототипів або ідей. Також у месенджері можна проводити відеодзвінки та ставити нагадування про важливі події. Крім того, Slack можна інтегрувати з Google Drive, Dropbox, GitHub, Trello, Outlook Calendar, Gmail.

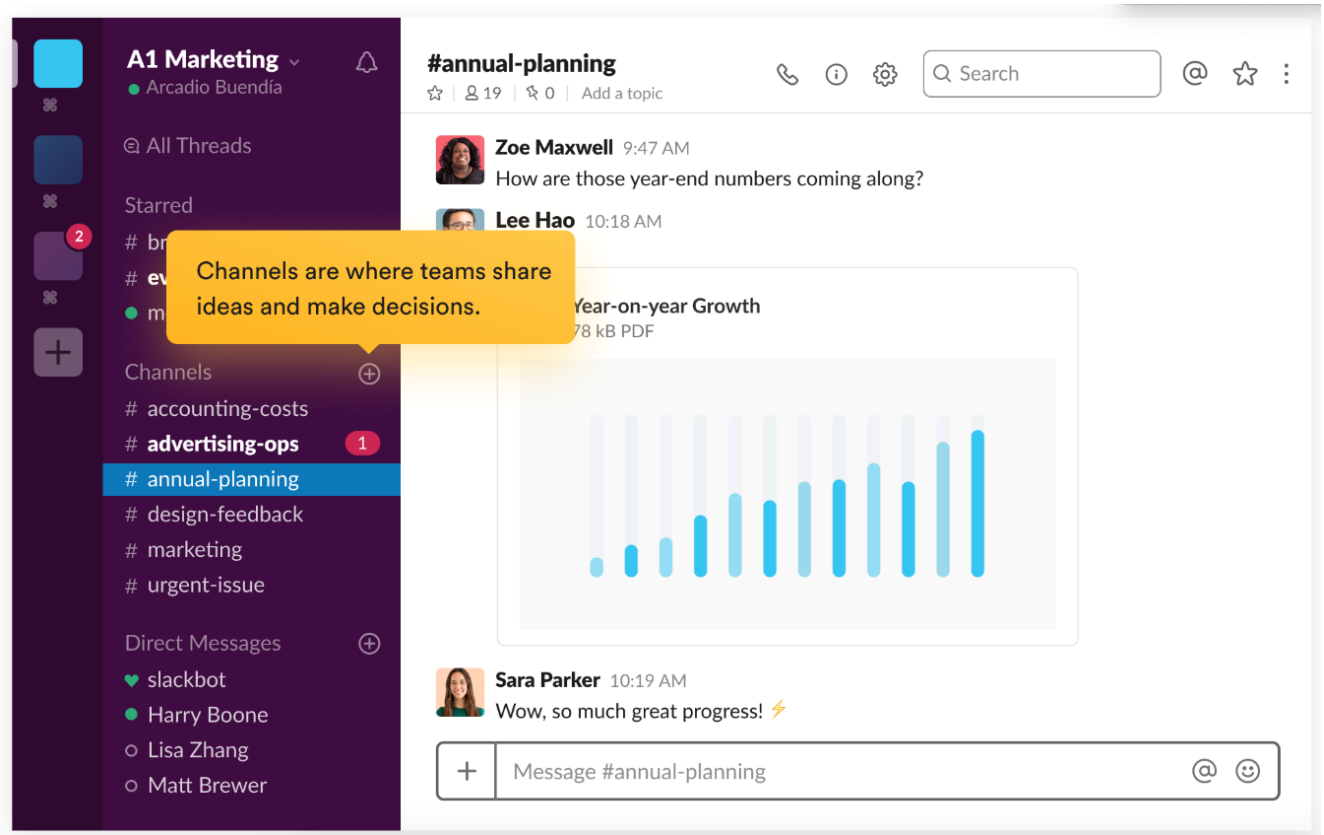

Рисунок 1.5 – Інтерфейс Slack

1.3.6 Програмний інструмент Jira

Jira – це програмний інструмент для управління проєктами, розроблений компанією Atlassian. У багатьох IT-компаніях Jira використовується для створення списку завдань, відстеження загального прогресу команди та вирішення проблем, що

виникають у процесі розробки продукту. Програмне забезпечення від Atlassian базується на принципах канбан- та скрам-дошок, традиційних методів організації завдань. Проте до цих принципів додаються додаткові механізми, спрямовані на спрощення створення нових додатків, додавання нових функцій, виправлення помилок тощо. Також ця система управління проєктами використовує методику Agile у розробці.

Інтерфейс Jira розділений на кілька ключових вкладок. У вкладці "Проєкти" зберігаються всі дошки Канбан/Скрам, які ви можете переглядати або редагувати. Це є основне робоче середовище, яке надає можливість перейти до режиму відстеження релізів продукту, переглянути всі активні спринти, проаналізувати звіти про виконану роботу і так д далі. Також у списку вкладок є вікно з інформаційними панелями - зручно скомпонованими аналітичними зведеннями. Окреме вікно зі списком співробітників для взаємодії, система планування релізів у стилі інструментів, схожих на OmniPlan, і вкладка з додатками від сторонніх компаній, які інтегровані в профіль Jira.

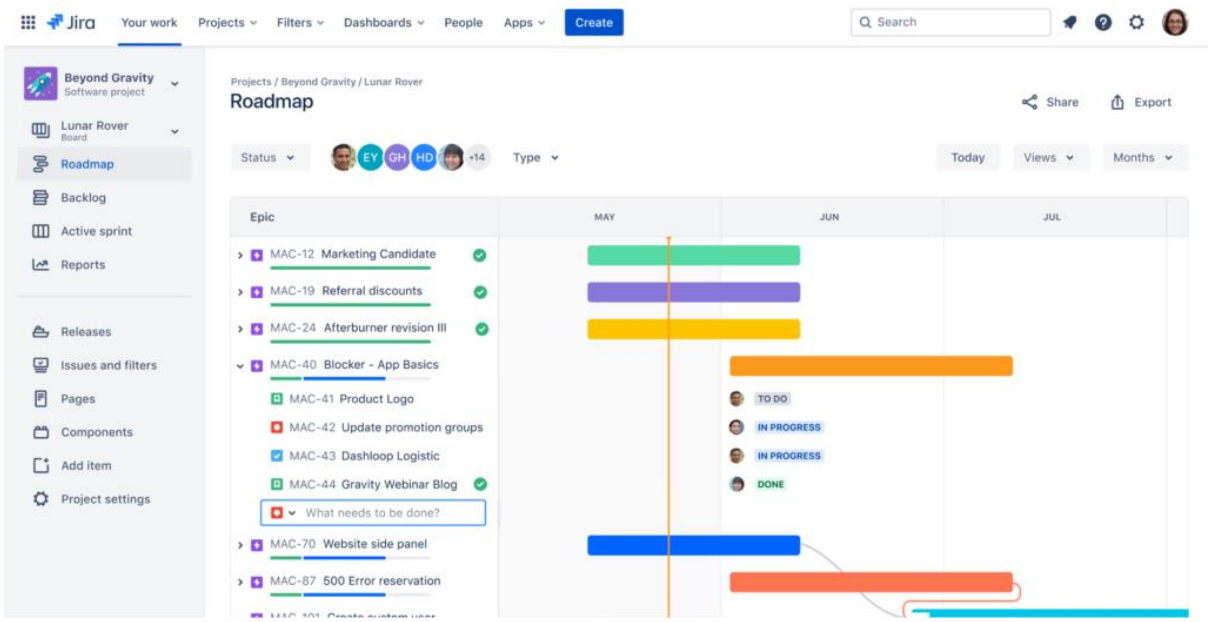

Рисунок 1.6 – Інтерфейс Jira

Висновки до першого розділу

У першому розділі роботи було проведено всебічний аналіз сучасних вебсистем для управління проєктами та контролю персоналу. Зокрема, було досліджено програмні продукти Trello, Harvest, Asana, GanttPRO, Slack та Jira. Кожна з цих систем має свої унікальні особливості, що можуть бути корисними для різних організаційних потреб.

В ході аналізу систем обуло визначено, що головними недоліками кожної є складність використання чи недостатність функціоналу не тільки для роботи над проєктом, а й для зручної комунікації між учасниками в одному середовищі. Тому було вирішено об’єднати головний функціонал деяких з них для створення зручного застосунку.

Наприклад, Trello пропонує простий та інтуїтивно зрозумілий інтерфейс з канбан-дошками, що дозволяють легко організувати завдання та відстежувати їхній статус. Основними перевагами Trello є його зручність у використанні та можливість швидкої адаптації до різних робочих процесів, що чудово допов~~н~~нить функціонал майбутнього застосунку.

Asana надає багатофункціональне середовище для управління проєктами, а реалізація його інтерфейсу є ідеальним рішення для створення статусів виконання завдань.

Slack є потужним інструментом для командної комунікації, його основні переваги включають зручність у використанні та можливість швидкого обміну інформацією. Тому саме ця система обрана за основу для створення чату проєкту.

Jira є одним з найпопулярніших інструментів для управління проєктами, особливо у сфері розробки програмного забезпечення. Вона має дуже зручно кан~~д~~бан-дошку для відстеження задач, помилок та інших робочих процесів, однак її складний інтерфейс та багатофункціональність можуть вимагати значного часу на навчання.

На основі аналізу вирішено, що саме об’єднання канбан-дошки як на основі Jira чи Trello та чат стануть чудовим рішенням для застосунку, що зробить його

унікальним та наддасть можливість на тільки роботи над проєктами, а й зручної комунікації між його учасниками. Також було визначено, що жодна з проаналізований систем не має доступу користувача до всіх його контактів, що розширило б функціонал застосунку і зробило можливість пошуку та об’єднання користувачів швидким та зручним. Тому є необхідність додати саме цей функціонал до майбутнього застосунку.

2 ТЕХНОЛОГІЇ ТА ІНСТРУМЕНТИ ДЛЯ РОЗРОБКИ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ

Для створення системи правління проєктами були обрані такі технології:

– для бекенд-розробки – Python, Django, Django REST Framework, Swagger;
– для фронтенд-розробки – Vite, Svelte, Tailwind CSS.

Розглянемо більш детально кожну з технологій.

2.1 Мова програмування Python

Python — це високорівнева, об'єктно-орієнтована мова програмування загального призначення з відкритим вихідним кодом.

Python часто порівнюють з мовами Perl та Ruby, адже всі вони є інтерпретованими та мають схожу швидкість виконання програм. Як і Perl, Python чудово підходить для написання скриптів. А подібно до Ruby, Python має добре продуману систему для об'єктно-орієнтованого програмування (ООП).

Ця мова відома своєю простотою, читабельністю та великою спільнотою.

Далі наведено деякі з ключових характеристик Python.

1. Python має лаконічний та зрозумілий синтаксис, який нагадує звичайну англійську мову. Це робить його чудовим вибором для початківців, а також полегшує читання та розуміння коду іншими розробниками.

2. Python можна використовувати для широкого кола задач, включаючи веб-розробку, машинне навчання, наукові обчислення, роботу з даними та автоматизацію.

3. Python поєднує в собі простоту з високою продуктивністю, що робить його чудовим вибором для розробки різноманітних програм.

4. Python має велику та активну спільноту розробників, яка створила безліч бібліотек та фреймворків для вирішення різноманітних задач.

5. Python є інтерпретованою мовою, тобто код не потрібно компілювати перед запуском. Це робить розробку швидшою та гнучкішою.

Python – це потужна та універсальна мова програмування, яка може бути корисною для людей з різним досвідом. Python є ідеальним вибором для розробки серверної частини застосунків завдяки своїй простоті, читабельності та великій спільноті розробників. Він дозволяє швидко реалізувати бізнес-логіку та забезпечує гнучкість у розширенні функціоналу.

2.2 Фреймворк Django

Django – це безкоштовний веб-фреймворк з відкритим кодом, написаний на мові програмування Python. Він використовується для розробки динамічних веб-сайтів та веб-додатків. Django полегшує та прискорює процес розробки, надаючи ряд компонентів та функцій.

Далі наведено деякі ~~компнентик~~компоненти та функції Django.

1. Система адміністрування. Django має вбудовану панель адміністрування, яка дозволяє легко керувати контентом сайту, користувачами та іншими налаштуваннями.

2. URL-маршрутизація. Django використовує систему URL-маршрутизації для зіставлення URL-адрес з відповідними функціями представлення.

3. Шаблони. Django використовує систему шаблонів, яка дозволяє розділяти HTML-код та логіку Python.

4. Обробка форм. Django має вбудовану підтримку обробки веб-форм, що спрощує збір даних від користувачів.

5. Система автентифікації та авторизації. Django має вбудовану систему автентифікації та авторизації, яка дозволяє контролювати доступ до сайту.

6. Підтримка баз даних. Django підтримує різні типи баз даних, включаючи PostgreSQL, MySQL та SQLite.

Django – це потужний та універсальний фреймворк, який можна використовувати для створення широкого спектру веб-сайтів та веб-додатків. Він є популярним серед розробників завдяки своїй простоті використання, масштабованості та великій спільноті. Він пропонує широкий спектр функцій, таких як адміністрування сайту, маршрутизація, аутентифікація та авторизація.

2.2.1 Огляд Django REST Framework

Django REST Framework (DRF) – це потужний та гнучкий набір інструментів для створення веб-API з використанням Django.

Далі наведено функціонал функцій, який полегшує процес розробки API:

- серіалізатори. DRF надає серіалізатори, які автоматично перетворюють об'єкти Python в JSON та навпаки. Це спрощує процес надсилання та отримання даних API;
- перегляди. DRF використовує перегляди для визначення логіки API. Перегляди написані на Python і легко настроюються, що робить їх гнучким інструментом для створення складних API;
- класи представлення. DRF пропонує класи представлення, які надають набір зручних методів для загальних завдань API, таких як отримання списків об'єктів, створення нових об'єктів та оновлення існуючих;
- аутентифікація та авторизація. DRF підтримує різні механізми аутентифікації та авторизації, що дозволяє вам контролювати доступ до вашого API;
- документація. DRF має вбудовану систему документації, яка автоматично генерує документацію для вашого API;
- розширюваність. DRF розширюється за допомогою системи класів та розширень, що дозволяє вам налаштувати його відповідно до своїх потреб.

2.3 Інструменти Swagger

Swagger – це набір інструментів для опису та документування інтерфейсів прикладних програм (API). Він використовується для визначення структури та можливостей API у форматі, зрозумілому як для людей, так і для машин. Цей формат опису називається специфікацією OpenAPI.

Swagger пропонує кілька інструментів для роботи зі специфікаціями OpenAPI, зокрема:

- Swagger Editor: веб-інтерфейс для створення та редагування специфікацій OpenAPI;
- Swagger Codegen: інструмент для генерації коду клієнта API з специфікації OpenAPI;
- Swagger UI: веб-інтерфейс для документування та тестування API на основі специфікації OpenAPI;

Swagger – це популярний інструмент для розробки API.

Далі наведено функціонал який полегшує Swagger.

1. Swagger генерує чітку та зрозумілу документацію для вашого API, що робить його більш доступним для розробників.

2. Специфікації OpenAPI є відкритими та ~~легкообміннимилегкообмінними~~, що полегшує співпрацю над API.

3. Swagger можна використовувати для автоматизації завдань, таких як генерація коду клієнта API та тестування API.

2.4 Аналіз інструментів Vite

Vite – це інструмент розробки Frontend з відкритим кодом, який використовується для створення швидких, гнучких та масштабованих веб-застосунків. Він використовує інноваційну архітектуру, засновану на pre-bundling, що робить його значно швидшим, ніж інші популярні інструменти, такі як Webpack або Rollup.

Далі наведено переваги використання Vite.

1. Швидкість. Vite використовує pre-bundling, щоб попередньо компілювати код JavaScript, CSS та інші ресурси, що робить запуск веб-застосунків значно швидшим.

2. Гнучкість. Vite має модульну систему, яка дозволяє легко імпортувати та експортувати модулі. Це робить його ідеальним для створення складних веб-застосунків.

3. Масштабованість. Vite може масштабуватися для обробки великих веб-застосунків з великою кількістю модулів.

4. Простота використання. Vite має простий у використанні API, що робить його доступним для розробників з будь-яким рівнем досвіду.

2.5 Фреймворк Svelte

Svelte – це JavaScript-фреймворк для веб-розробки, який набирає популярності завдяки своїй унікальній архітектурі та декларативному підходу до розробки інтерфейсів користувача. На відміну від інших фреймворків, таких як React або Vue.js, Svelte не використовує віртуальний DOM, а натомість компілює декларативний код у машинний, що робить його надзвичайно швидким і ефективним.

Далі наведено переваги Svelte.

1. Швидкість. Svelte є одним з найшвидших JavaScript-фреймворків, доступних сьогодні, завдяки його унікальній архітектурі та компіляції коду.

2. Простота. Svelte має простий і зрозумілий синтаксис, що робить його легким для вивчення та використання.

3. Ефективність. Svelte використовує мінімальний обсяг коду, що робить його ідеальним для створення веб-додатків з низьким ресурсним профайлом.

4. Гнучкість. Svelte можна використовувати для створення будь-якого типу веб-додатків, від простих односторінкових до складних багатосторінкових.

2.6 Фреймворк Tailwind

Tailwind CSS – це CSS-фреймворк з відкритим кодом, який пропонує інноваційний підхід до стилізації веб-інтерфейсів. На відміну від традиційних фреймворків, що ґрунтуються на компонентах, Tailwind використовує набір утилітних класів, які надають низькорівневий контроль над візуальними властивостями HTML-елементів.

Далі наведено основні характеристики Tailwind CSS.

1. Tailwind надає широкий спектр готових класів, що охоплюють майже всі аспекти стилізації, від базових властивостей (колір, шрифт, відступи) до складних компонентів (меню, кнопки, форми).

2. Завдяки комбінуванню класів можна створювати практично будь-який дизайн, маючи повний контроль над візуальними аспектами.

3. Tailwind вбудовано підтримує адаптивність, дозволяючи легко налаштовувати стилізацію для різних розмірів екранів.

4. Навчання Tailwind CSS не потребує багато часу, адже його синтаксис простий та зрозумілий.

5. Tailwind генерує мінімальний набір CSS-коду, необхідний для вашого проєкту, що покращує продуктивність.

Висновки до другого розділу

У другому розділі роботи детально розглянуто технології та інструменти, що використовуються для розробки системи управління проєктами. Вибір цих інструментів був зроблений на основі їхньої відповідності завданням проєкту та вимогам користувачів.

Python був обраний як основна мова програмування для бекенд-розробки завдяки його простоті, широким можливостям і великій кількості бібліотек та фреймворків, які спрощують процес розробки. Django забезпечує високу швидкість розробки, безпеку та масштабованість, що робить його ідеальним вибором для

створення складних вебзастосунків. Swagger використовується для документування API, що дозволяє розробникам швидко орієнтуватися у функціоналі системи та полегшує процес інтеграції з іншими системами.

Для фронтенд-розробки було обрано Vite, Svelte та Tailwind CSS.

Vite використовується як інструмент для швидкого створення проєктів та збірки додатків, що забезпечує високу продуктивність та зручність у використанні. Svelte є сучасним фреймворком для створення інтерфейсів користувача, який забезпечує високу швидкість роботи додатків та зручність розробки завдяки своїй простоті та ефективності. Tailwind CSS використовується для стилізації додатка, надаючи гнучкі інструменти для створення адаптивних та естетично привабливих інтерфейсів.

Тому саме ці технології було обрано через велику кількість переваг, які дозволяють створити необхідну систему швидко та легко. Загалом, їх використання в сукупності дозволить створити потужну та ефективну систему підтримки ~~менеджменту~~менеджменту з інтуїтивно зрозумілим інтерфейсом та швидкою реакцією на потреби користувачів.

3 РЕАЛІЗАЦІЯ ВЕБСИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ ТА КОНТРОЛЮ ПЕРСОНАЛУ

3.1 Моделювання вебсистеми

Моделювання системи вебсайту для університетських стартапів — це складний процес, який передбачає кілька важливих етапів, які охоплюють аналіз вимог користувачів, визначення функціональних вимог, розробку архітектури системи, вибір технологій та реалізацію функціональності.

Почнемо з аналізу потреб. Це перший крок у моделюванні системи — розуміння, які конкретно потреби мають користувачі. У цьому випадку це студенти, викладачі та адміністратори університету, які працюють над стартапами. Студентам потрібен інструмент для співпраці та керування проєктами, який би дозволяв їм ефективно комунікувати, розподіляти завдання та відстежувати прогрес. Викладачі хочуть мати можливість надавати студентам підтримку та відстежувати прогрес їхніх проєктів. Адміністратори бажають мати зручний інструмент для управління користувачами та даними.

На основі аналізу потреб можна сформулювати функціональні вимоги до системи. Це може включати створення проєктів, управління завданнями, чат для спілкування, канбан-дошки для організації завдань та керування.

Наступним етапом є розробка архітектури системи.

Для реалізації функціональних вимог потрібно спроектувати архітектуру системи. Це включає розробку бази даних для зберігання інформації про користувачів, проєкти, завдання та повідомлення. Також потрібно розробити серверну частину системи, яка відповідає за логіку бізнес-процесів та взаємодіє з базою даних. На клієнтській стороні потрібно розробити інтерфейс користувача, який би був зручним та інтуїтивно зрозумілим.

Розглянемо візуальну модель майбутньої вебсистеми більш детально.

На першій сторінці клієнт буде мати форму для реєстрації чи авторизації. На рисунку 3.1 представлено макет форми.

Рисунок 3.1 – Сторінка форми реєстрації та авторизації

При натисканні на кнопку клієнт буде потрапляти на головну сторінку вебсистеми. На верхній панелі якої розташовані логін користувача та кнопки для переходу на сторінки Реєстрація та Контакти.

На головному просторі сторінки будуть розташовані проєкти користувача. Кожен проєкт буде мати вигляд картки із зображенням, назвою та описом даного проєкту.

На рисунку 3.2 представлено макет головної сторінки.

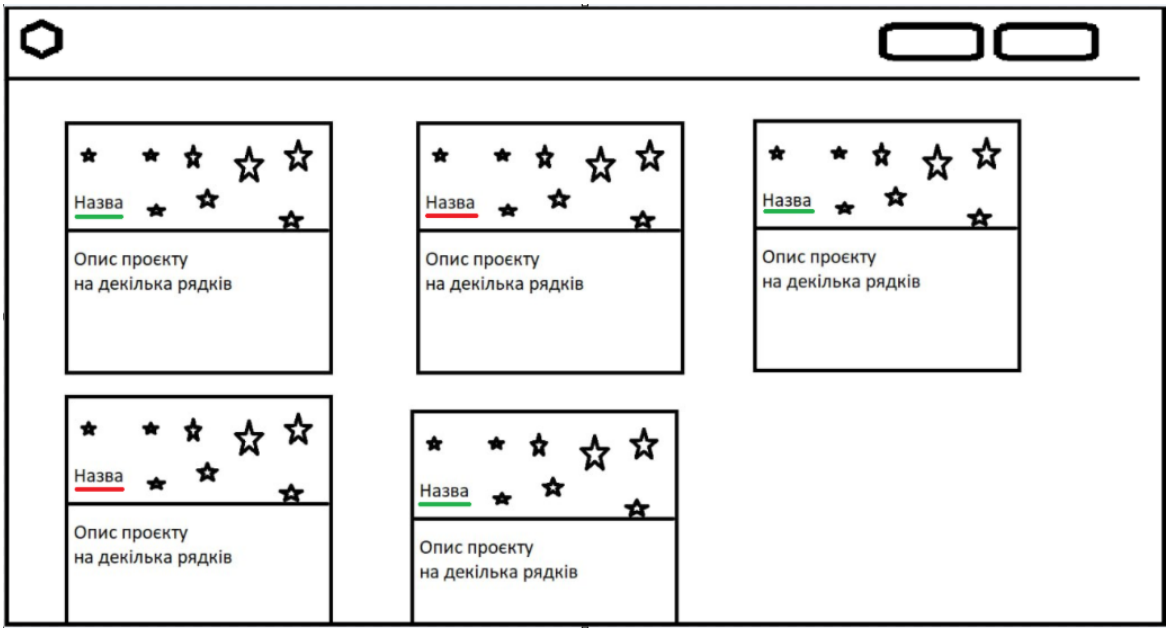

Рисунок 3.2 – Макет головної сторінки

При натисканні на карту одного з проєктів клієнт буде переходити на сторінку, з розгорнутою інформацією про цей проєкт. На головному просторі цієї сторінки буде розташована канбан-дошка з задачами різної категорії. Бічна панель сторінки буде мати чат учасників даного проєкту. На рисунку 3.3 представлено макет сторінки.

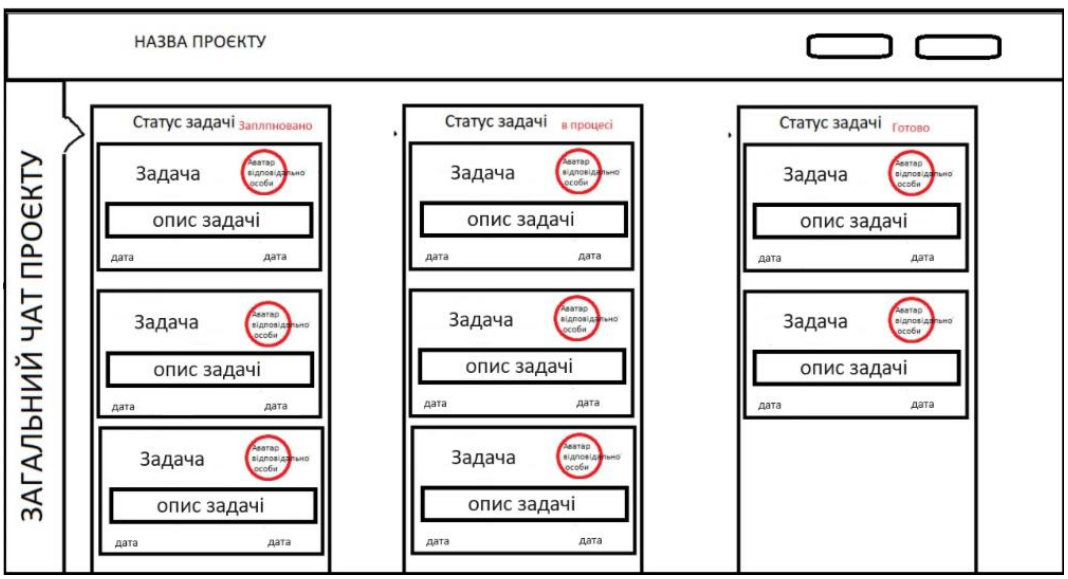

Рисунок 3.3 – Макет сторінки проєкту

Також у вебсистеми передбачається сторінка з контактами користувача, які додаються автоматично після роботи над спільним проєктом. Цей функціонал розроблена для швидкого знаходження користувачами один одного з метою додавання до нових проєктів.

На рисунку 3.4 представлено вигляд макету сторінки з контактами.

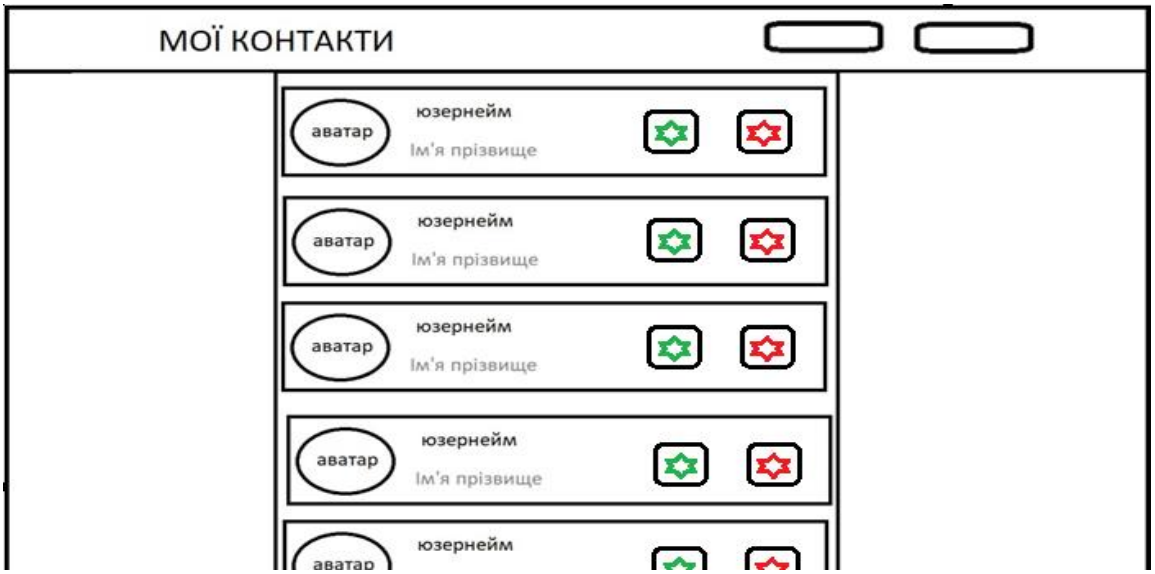

Рисунок 3.4 – Макет сторінки проєкту

Сторінка буде мати вигляд списку з контактами користувача. Кожен контакт буде мати повну інформацію даного користувача, а саме, його юзрнейм, ім’я та прізвище, аватар. Поряд з інформацією передбачаються кнопки додавання та видалення ~~контакта~~контакта з необхідних проєктів.

Після визначення архітектури системи потрібно вибрати технології, за допомогою яких вона буде реалізована. Наприклад, для серверної частини можна використовувати мови програмування, такі як Python або JavaScript, та фреймворки, такі як Django або Node.js. Для бекенд-розробки можна використовувати Python, Django, Django REST Framework, Swagger, а для фронтенд-розробки – Vite, Svelte, Tailwind CSS. Для бази даних можна використовувати реляційні бази даних MySQL. Більш детально технології розписані в розділі 2.

Після вибору технологій можна розпочати розробку системи. Розробка проводиться крок за кроком, здійснюючи реалізацію різних функцій та компонентів системи. Після завершення розробки систему необхідно протестувати, перевіривши її на відповідність функціональним вимогам та виявивши та виправивши можливі помилки та недоліки.

Після успішного завершення розробки та тестування систему можна впровадити в експлуатацію. Після впровадження вона потребуватиме постійної підтримки та поновлень, щоб забезпечити її надійну та ефективну роботу.

3.2 Зовнішній вигляд та функціонал робочого проєкту

Для керування проєктами університетських стартапів, зручності комунікації між студентами та налагодження роботи у команді будо вирішено розробити вебсистеми MicroJira, яка б перекривала усі потреби та покращила роботу студентів.

Вебсайт має виглядати сучасно та інтуїтивно зрозуміло, забезпечуючи всі необхідні інструменти для керування проєктами, комунікації між учасниками та організації робочих процесів. Він має надавати користувачам всі необхідні інструменти для ефективної роботи, забезпечуючи сучасний та інтуїтивно зрозумілий інтерфейс.

Мета такого вебсайту — допомогти студентам і викладачам створювати, керувати та реалізовувати стартапи, полегшуючи комунікацію та організацію робочих процесів за допомогою зручних чатів та канбан-дошок.

Рисунок 3.5 – Макет сторінки проєкту

Вебсистема зустрічає користувача сторінкою з коротким описом платформи, її особливостями, що дає можливість користувачу швидко зрозуміти її переваги та зробити вибір щодо використання цього програмного продукту. На рисунку 3.5 представлено вигляд першої сторінки вебсистеми.

3.2.1 Навігаційне меню вебсистеми

Розроблена вебсистема має статичне навігаційне маню, яке розташоване зверху і зберігається на всіх сторінках. Верхня ліва частина містить заголовок з назвою вебсайту та логотипом, створюючи перше враження про нього. В правій частині розташоване навігаційне меню з основними кнопками, яке представлено на рисунку 3.6.

Кнопка «Домашня» відповідає за головну сторінку вебсистеми з переліком проєктів. За допомогою кнопки «Про нас» користувач може переглянути повну інформацію щодо проєкту, а кнопка «Контакти» дозволяє переглянути всіх користувачів і швидко знайти необхідного в зручному списку. «Увійти» та «Реєстрація» відповідають за сторінки реєстрації та авторизації користувачів.

Рисунок 3.6 – Відображення навігаційного меню

Після авторизації замість кнопки «Увійти» з’являються юзернейм та аватар користувача, що представлено на рисунку 3.7.

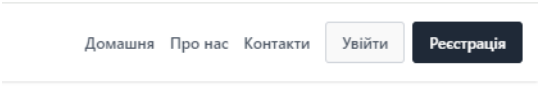

Рисунок 3.7 – Відображення меню після авторизації користувача

Це меню завжди залишається на видимому місці, забезпечуючи легкий доступ до всіх основних розділів сайту що дозволяє швидко орієнтуватися на сайті та знаходити потрібну інформацію.

3.2.1 Головна сторінка вебсистеми

Головна сторінка вебсайту вітає користувачів сучасним і чистим дизайном.

Центральним елементом головної сторінки є список всіх проєктів. Він представлений у вигляді карточок, де кожен проєкт відображається з коротким описом, назвою та статусом виконання. Також передбачено поле для пошука необхідних проектів, що полегшує роботу користувача з великим обсягом даних.

Кожна картка проєкту також є кнопкою, при натисканні на котру, користувач переходить на сторінку з детальною інформацією про нього. На рисунку 3.8 представлено вигляд сторінки з проєктами. В кінці списку проєктів розташована кнопка «Додати проєкт», при натисканні на яку користувач може створити новій проект та додати всі необхідні дані.

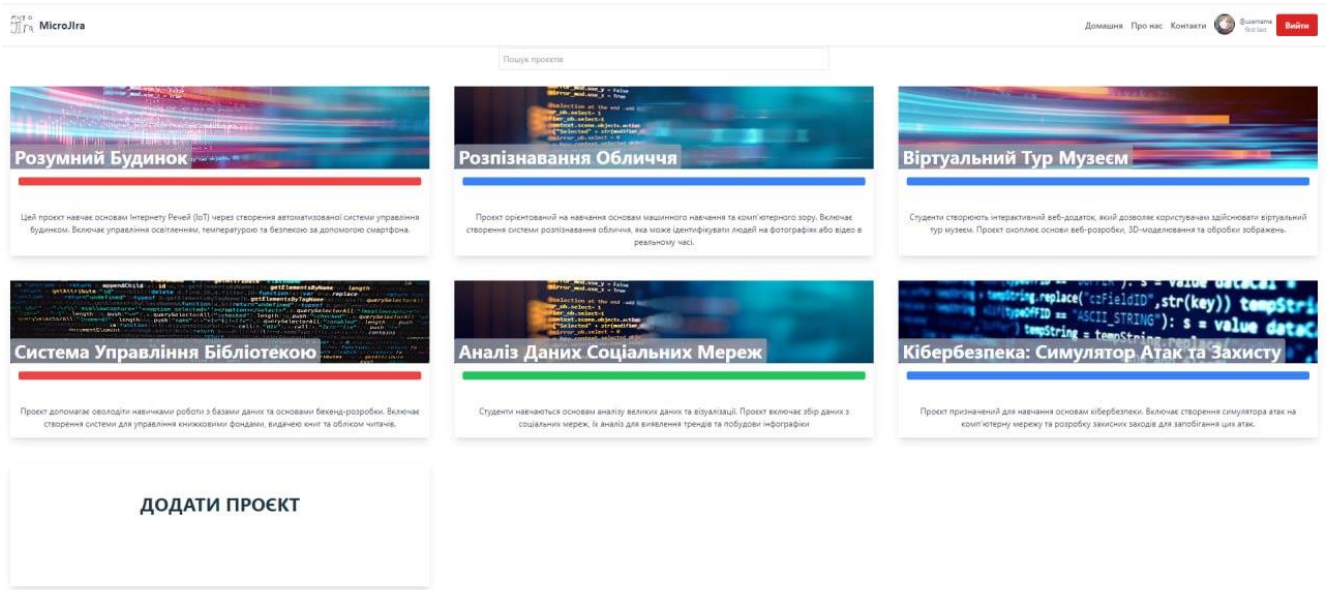

Рисунок 3.8 – Відображення сторінки з проєктами

На рисунках 3.9 та 3.10 представлено вигляд картки з проєктом.

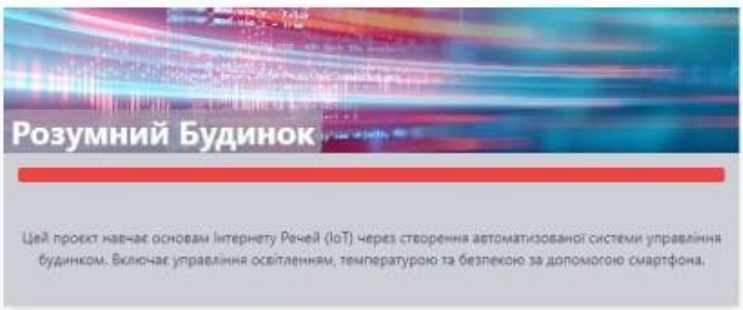

Рисунок 3.9 – Відображення проєкта при натисканні

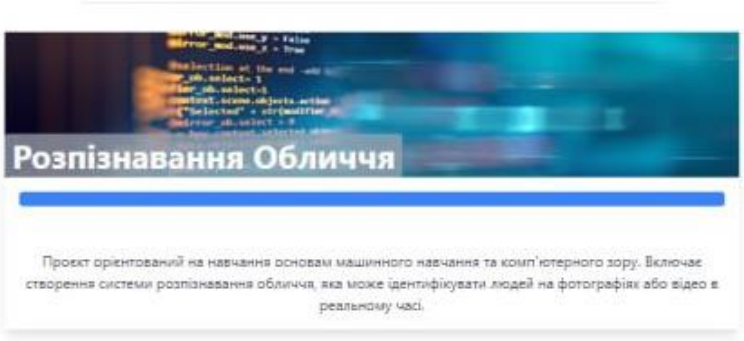

Рисунок 3.10 – Відображення проєкта при натисканні

Кожна картка проєкту має приємний та функціональний зовнішній вигляд. Вони можуть включати також зображення або логотип проєкту для полегшення ідентифікації. При натисканні на картку колір фону змінюється с білого на сірий що сприяє зрозумілій навігації та виділенню обраної картки з проєктом. Кожна може мати різні кольорові маркери або іконки для швидкої ідентифікації статусу проєкту, наприклад, "на стадії розробки", "в процесі тестування", "завершено".

3.2.2 Опис сторінки проєкту

При переході до конкретного проєкту, користувач потрапляє на детальну сторінку. Тут великий заголовок з назвою проєкту та описом привертає увагу, надаючи всю необхідну інформацію про проєкт на першому погляді. Загальний вигляд сторінки з проєктом представлено на рисунку 3.11.

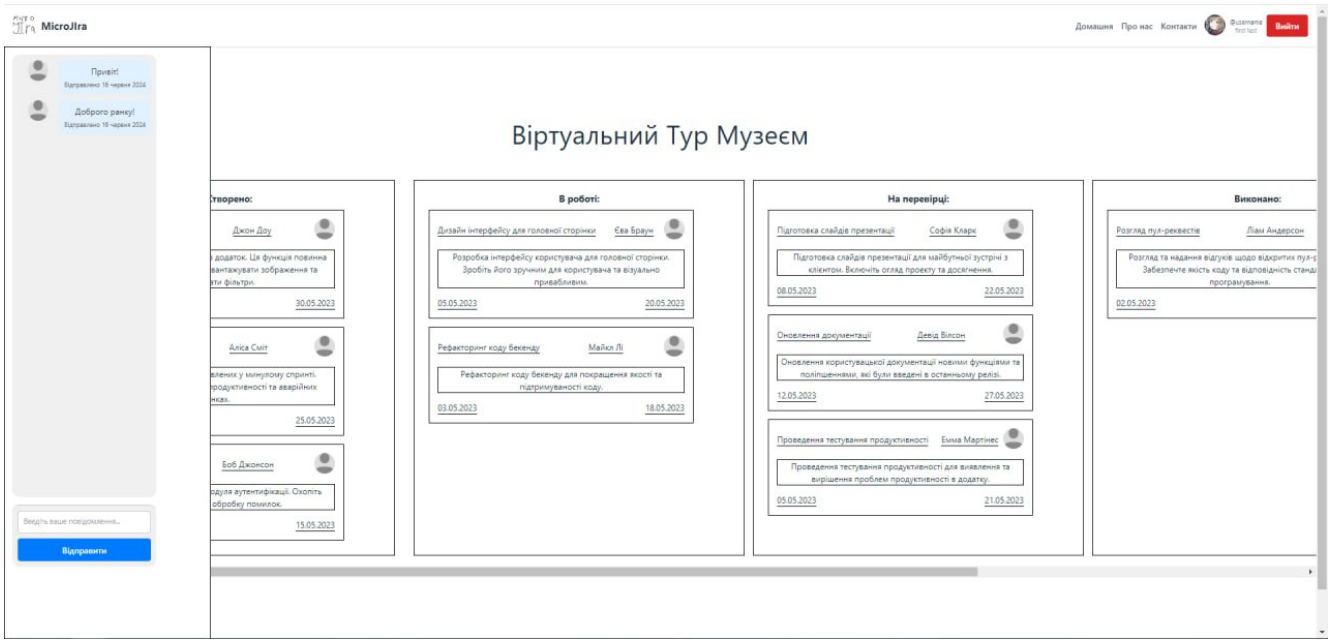

Рисунок 3.11 – Загальний вигляд сторінки з проєктом

Одразу під заголовком розташована канбан-дошка, яка є ключовим інструментом для організації робочого процесу. Вона представлена на рисунку 3.12. Канбан-дошка поділена на колонки, такі як «Зробити», «В роботі», «На перевірці» та «Виконано». Завдання представлені у вигляді карточок, які можна

перетягувати між колонками в залежності від статусу виконання. Це забезпечує наочне відображення статусу кожного завдання та дозволяє легко керувати ними. При необхідності створити нове завдання користувач може клікнути на поле з проектами та додати нову задачу.

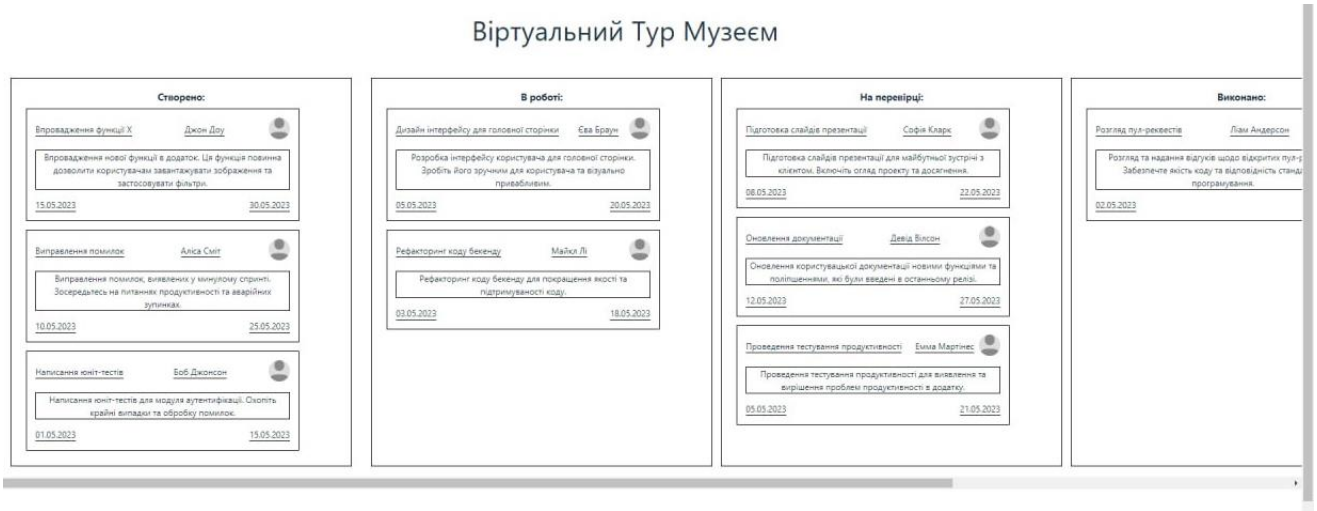

Рисунок 3.12 – Відображення канбан-дошки проєкту

На рисунку 3.13 представлено вигляд колонки зі статусом «В роботі».

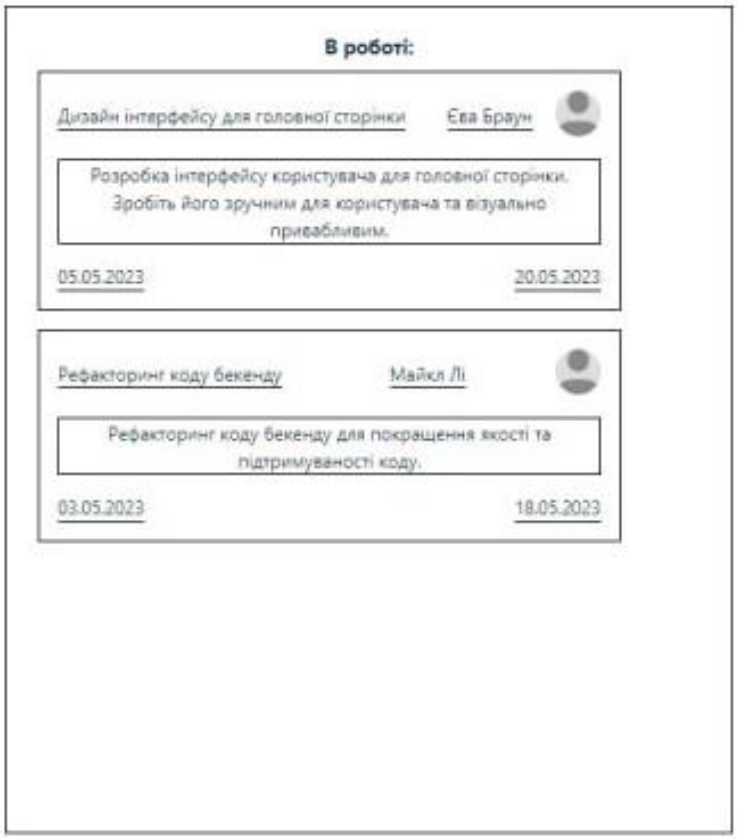

Рисунок 3.13 – Вигляд колонки з завданнями

На першому рядку кожної картки з завданням розташовані назва завдання – справа, та данні визначеного виконавця, а саме його ім’я, прізвище та аватар – зліва. Це дозволяє легко орієнтуватись між завдання та швидко знаходити необхідні. Далі йде опис кожного завдання, що дає розуміння необхідного обсягу роботи для його виконання. Внизу кожної картки розташовані початок та дедлайн виконання завдання зліва та справа відповідно. На рисунку 3.14 представлено вигляд картки з завданням.

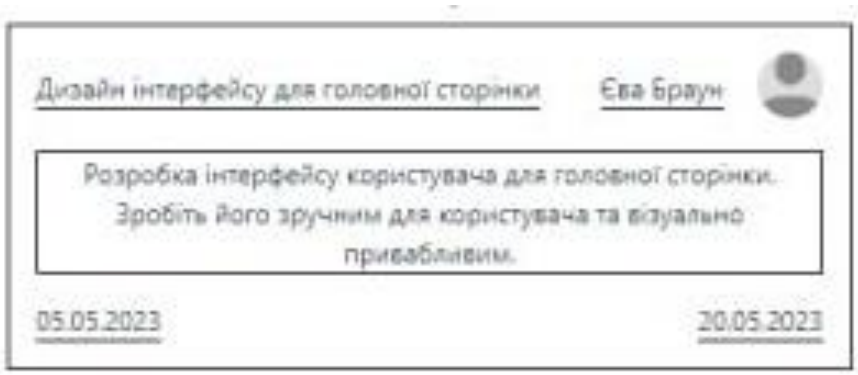

Рисунок 3.14 – Вигляд картки з завданням

3.2.4 Вигляд проєктного чату

Однією з ключових функцій сайту є інтегрований чат. Він дозволяє учасникам обговорювати всі питання в реальному часі. Чат розміщений збоку сторінки з проєктом, забезпечуючи зручний доступ без необхідності переходити на іншу сторінку, як представлено на рисунку 3.11. Тут учасники можуть обмінюватися повідомленнями та залишати коментарі до завдань.

Чат має вигляд бокової панелі з полем для вводу повідомлення, кнопкою для його відправлення знизу. На основному полі чату розміщуються повідомлення кожного учаснику проєкту, які, окрім тексту, який відправив користувач, містять аватар відправника та дату відправки, що дозволяє швидко орієнтуватися та легко обмінюватись думками та ідеями з колегами.

На рисунку 3.15 представлено вигляд чату проєкту.

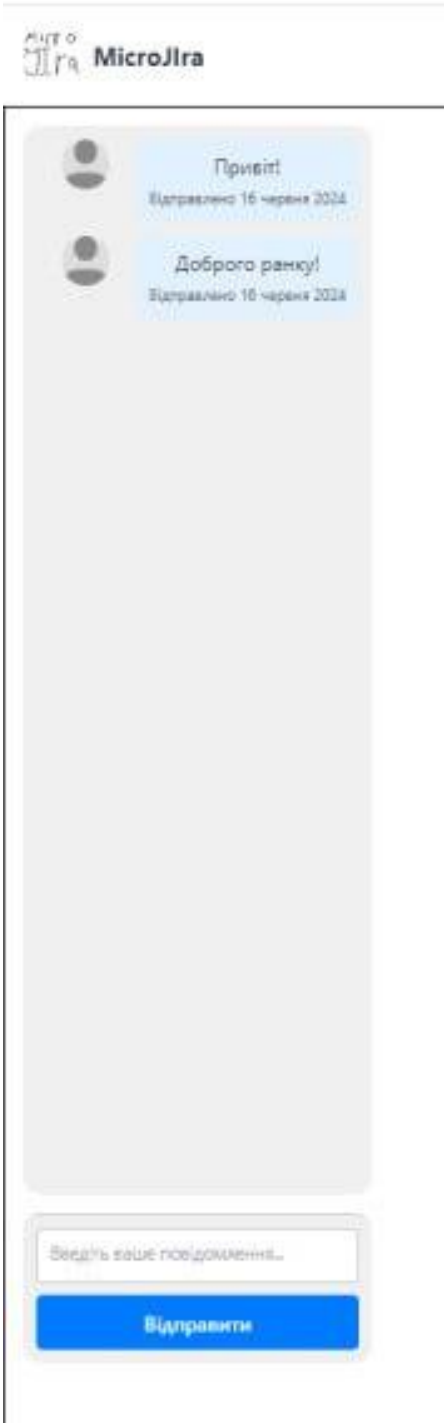

Рисунок 3.15 – Відображення проєкта при натисканні

3.2.4 Сторінка контактів

Для зручної навігації на адміністрування проєктів було вирішено розробити сторінку контактів. На ній користувач може переглянути всі власні контакт, а також додати чи видалити необхідний контакт з будь-якого проєкту.

Сторінка контактів представлена на рисунку 3.16.

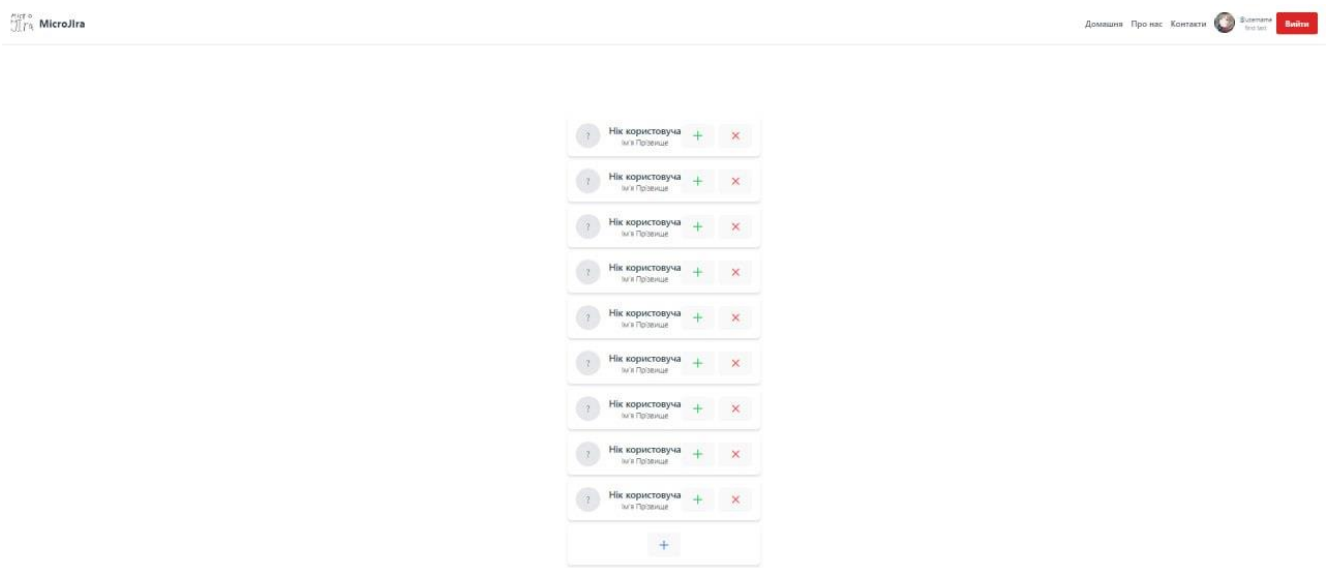

Рисунок 3.16 – Відображення сторінки контактів

Кожен контакт має вигляд картки з інформацією щодо відповідного користувача. Для швидкої на зручної навігації між ними на екрані розташовано аватар користувача, його юзернейм, під яким уточняється повне ім’я та прізвище.

На рисунку 3.17 представлено вигляд одного з контактів списку.

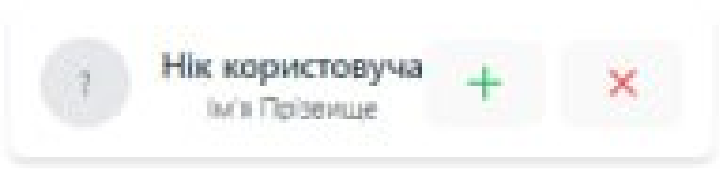

Рисунок 3.17 – Відображення вигляду контакта

Для додавання та видалення контакту передбачено дві кнопки справа з зеленим чи червоним хрестиком відповідно. Після натискання на одну з кнопок з’являється вікно з переліком активних проєктів, і при натисканні на будь-який з

них буде виконана команда, передбачена кнопкою. На рисунку 3.18 представлено вікно для обрання проекту.

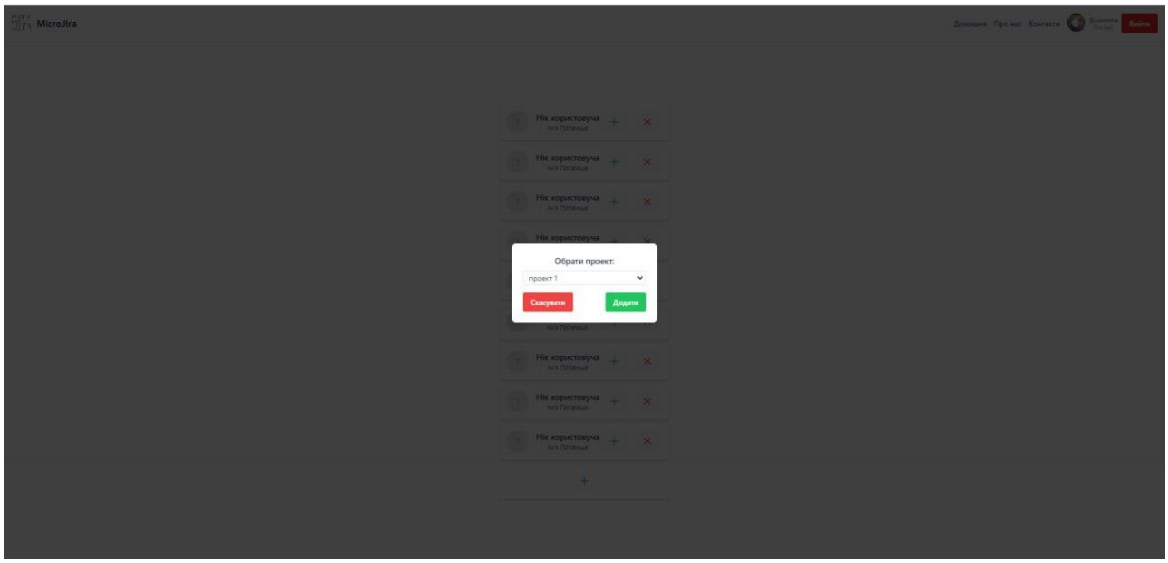

Рисунок 3.18 – Вікно з переліком проектів

3.2.5 Функціонал авторизації та реєстрації

Для авторизації та реєстрації на сайті передбачені окремі сторінки з відповідними формами, які представлені на рисунках 3.19 та 3.20.

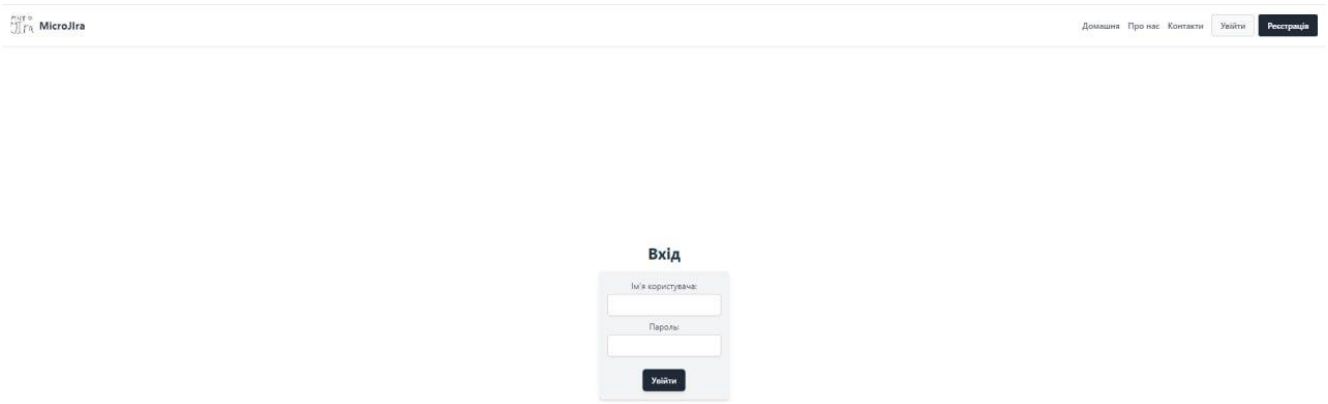

Рисунок 3.19 – Сторінка авторизації

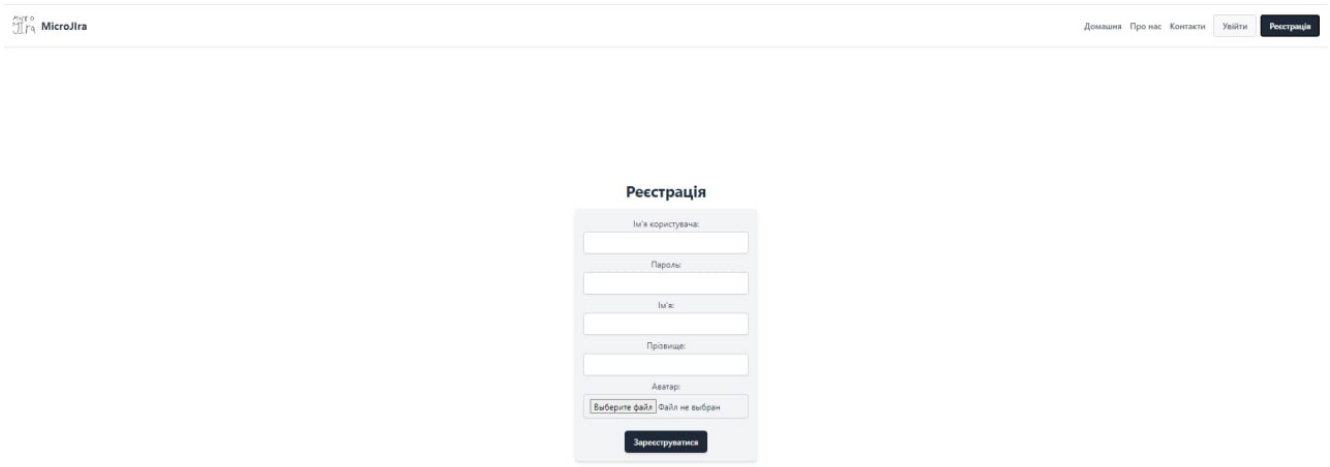

Рисунок 3.20 – Сторінка реєстрації

Форма авторизації містить поля для введення юзернейму користувача та паролю з кнопкою для входу. На сторінці реєстрації нові користувачі можуть ввести свій юзернейм, пароль, ім’я, прізвище та фотографію, щоб створити новий акаунт. Процес реєстрації простий та швидкий, що дозволяє новим користувачам легко приєднуватися до платформи.

Рисунок 3.21 – Верхня панель зареєстрованого користувача

На верхній панелі розташована кнопка «Вийти», завдяки якій зареєстрований користувач може вийти з системи і пройти авторизацію чи реєстрацію знову.

Таким чином, вебсайт для університетських стартапів є потужним інструментом, який забезпечує всебічну підтримку команд, сприяючи ефективній співпраці та управлінню проєктами в межах університету. Це інтегроване середовище дозволяє учасникам проєктів легко комунікувати, організовувати робочі процеси та відстежувати прогрес. Він об’єднує всі необхідні функції для

комунікації, організації робочих процесів, управління документами та відстеження прогресу, що в кінцевому підсумку сприяє успішній реалізації стартапів. Інтуїтивно зрозумілий інтерфейс, потужні інструменти та інтеграція з іншими сервісами роблять цей вебсайт незамінним для команд, які прагнуть досягти своїх цілей та розвивати свої проєкти на найвищому рівні.

Важливим аспектом такого вебсайту є його здатність адаптуватися до потреб різних користувачів, включаючи студентів, викладачів та адміністрацію університету. Завдяки цьому, він може стати основним інструментом для розвитку інновацій та сприяти створенню нових успішних стартапів в академічному середовищі.

Учасники проєктів отримують можливість ефективно співпрацювати, обмінюватися ідеями, ресурсами та інформацією, що значно підвищує їхню продуктивність та шанси на успіх. Завдяки комплексному підходу до управління проєктами, вебсайт допомагає зменшити ризики та покращити координацію між членами команди, забезпечуючи прозорість та контроль на всіх етапах реалізації проєкту.

Університети, які впроваджують такі платформи, можуть очікувати зростання кількості стартапів та підвищення якості їхньої реалізації. Це, в свою чергу, сприятиме зміцненню репутації університету як інноваційного центру та підвищенню його привабливості для талановитих студентів та викладачів з усього світу.

3.3 Програмна реалізація проєкту

Вебсистема складається з двох частин:

– бекенд-сервера, який відповідає за усю логіку роботи системи, взаємодію шляхом запитів через ORM (Object-Relational Mapping) з базою даних, відповідає на запити фронтенд-частини;

– фронтенд-сервера, який генерує SPA (single page application), яке виконується в браузері, динамічно змінює DOM-дерево, надсилає запити даних на

бекенд-сервер та відображає їх, тобто є інтерфейсом взаємодії з основною частиною системою.

Розглянемо окрему кожну з цих частин програмної реалізації вебсистеми.

3.3.1 Реалізація фронтенд-частини

Фронтенд складається з пакетного менеджера vite, який є рушієм системи та змушує працювати разом два фреймворки: фронтенд фреймворк Svelte і css фреймворк TailWInd CSS. Обидва є чудовими інструментами у розробці інструментами, бо динамічно оновлюють свої елементи у разі змін файлів проєкту.

Для ініціалізації проєкту було використано команди, які вказані на рисунку 3.22.

```
npm create vite@latest svelte-app -- --template svelte
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init tailwind.config.cjs -p
```

Рисунок 3.22 – Ініціалізація фреймворків Tailwind CSS

Для того, аби tailwind оновлював стилі, написані у .svelte файлах було змінено налаштування tailwind, що представлено на рисунку 3.23.

svelte-app > JS tailwind.config.js > ...

```
1  /** @type {import('tailwindcss').Config} */
2  export default {
3    plugins: [],
4      theme: {
5        extend: {},
6      },
7    purge: ["./index.html",'./src/**/*.{svelte,js,ts}'],
8    variants: {
9      extend: {},
10   },
11   darkmode: false,
12 }
```

Рисунок 3.23 – Налаштування Tailwind CSS

Тепер основні технології можуть взаємодіяти між собою. Структура проєкту має вигляд, який представлено на рисунку 3.24.

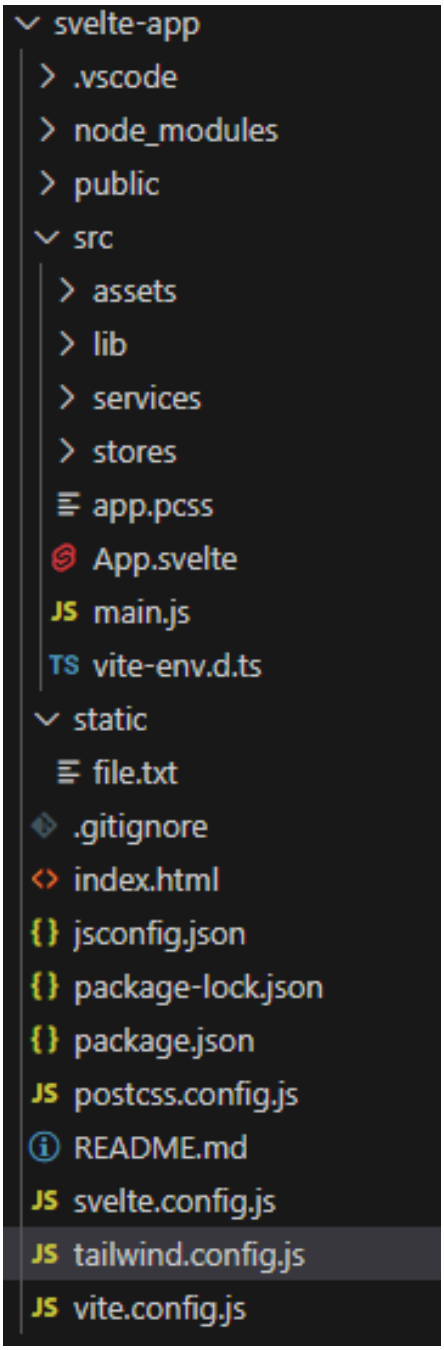

Рисунок 3.24 – Структура проєкту (фронтенд)

В папці lib зберігаються svelte-компоненти, які використовуються під час роботи сервера. Папка services налаштовує взаємодію з іншими сервісами (має

лише один файл апі.жс, бо взаємодіє лише з одним сервісом). Файли папки Stores служать для того, аби вміщувати в собі внутрішній стан програми.

На рисунку 3.25 розглянемо приклади коду кожної з цих директорій, аби на прикладах розібрати структуру програми.

```
import { get } from 'svelte/store';
import { authStore , setTokens} from '../stores/auth';
import { profileStore, setProfile } from '../stores/profile';

const BASE_URL = 'http://localhost:8000';

async function request(endpoint, method = 'GET', body = null, accessToken = null, isFormData = false) {
    const headers = {

    };
    if (!isFormData) {
        headers['Content-Type'] = 'application/json';
    }
    body.forEach((value, key) => {
        console.log(key, value);
    });
    if (accessToken) {
        headers['Authorization'] = `Bearer ${accessToken}`;
    }

    const options = {
        method,
        headers: headers,
    };

    if (body && method !== 'GET' && method !== 'HEAD') {
        options.body = isFormData ? body : JSON.stringify(body);
    }
    console.log(endpoint, options);
    const response = await fetch(`${BASE_URL}${endpoint}`, options);

    if (!response.ok) {
        const error = await response.json();

        throw new Error(`Request failed: ${JSON.stringify(error)}`);
    }

    return response.json();
}
```

Рисунок 3.25 – Функції-запити до бекенд-сервера

Базовою функцією є функція request, яка є шаблоном для всіх запитів проєкту. У ній вказується тип запиту, ендпоінт, тіло запиту і додаткові параметри, специфічні для деяких запитів.

Прикладами використання функції request є register та login, які представлені на рисунку 3.26.

```
export async function register(username, password, first_name, last_name, avatar) {
    console.log(username, password, first_name, last_name, avatar);
    const formData = new FormData();
    formData.append('username', username);
    formData.append('password', password);
    formData.append('first_name', first_name);
    formData.append('last_name', last_name);

    if (avatar) {
        formData.append('avatar', avatar);
    }
    formData.forEach((value, key) => {
        console.log(key, value);
    });
    const data = await request('/register/', 'POST', formData, null, true);
    setTokens(data.access, data.refresh);
    setProfile(data.username, data.avatar, data.first_name, data.last_name);
    return data;
}
export async function login(username, password) {
    const data = await request('/login/', 'POST', { username, password }, null, true);
    setTokens(data.access, data.refresh);
    setProfile(data.username, data.avatar, data.first_name, data.last_name);
    return data;
}
```

Рисунок 3.26 – Приклад використання функції request у запитах

Саме через такі функції, які готують дані для використання JS fetch API здійснюється вся взаємодія з бекнендом.

Наступними є сховища. Прикладом такого є сховище токенів, яке представлене на рисунку 3.27.

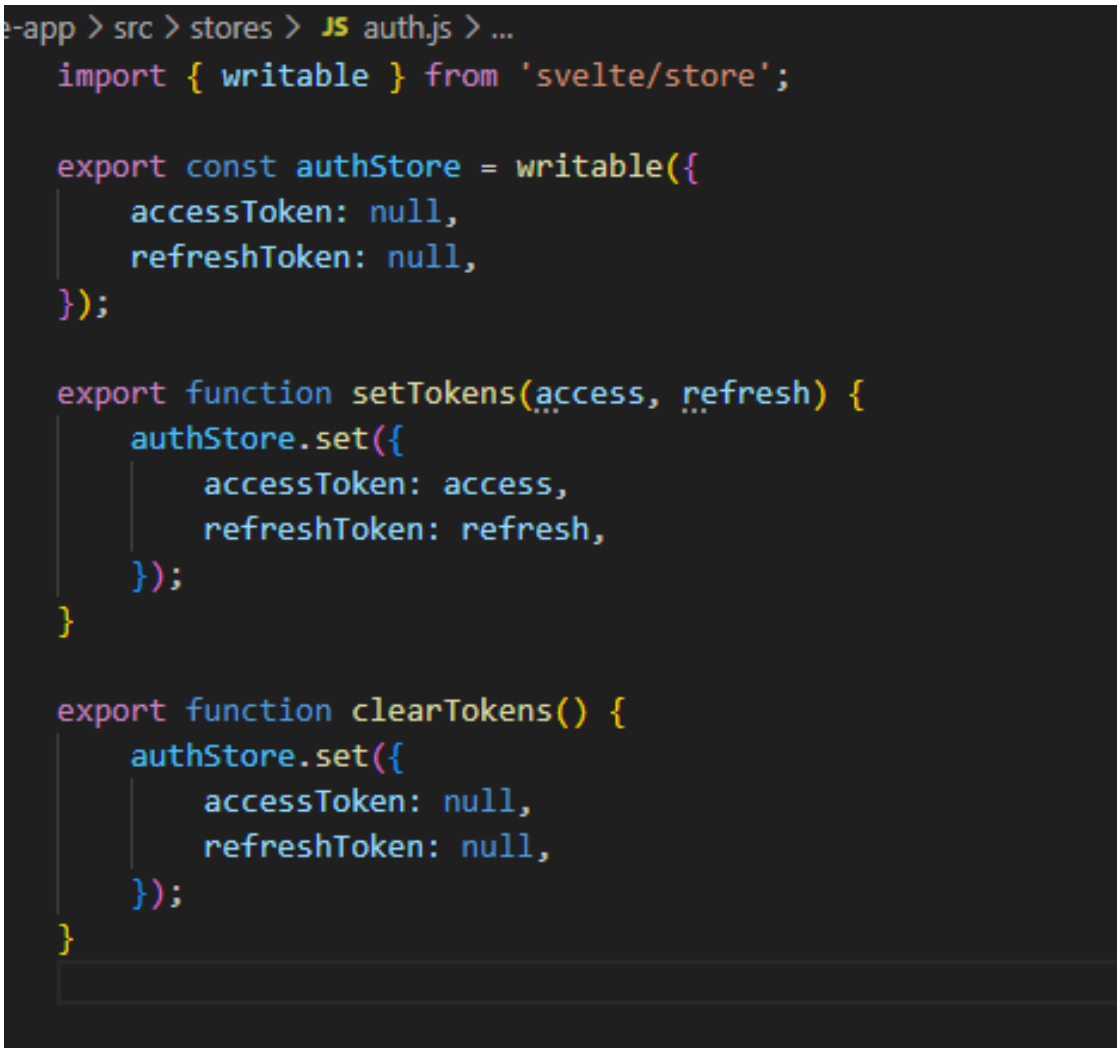

Рисунок 3.27 – Контейнери Svelte

Складається з самого сховища, які зберігаються у ~~спеціальних~~спеціальних об’єктах Svelte, та функцій для взаємодії з ним.

Прикладом Svelte-компонента є Header.svelte, в якому реалізовано header вебінтерфейсу системи на рисунку 3.28.

svelte-app > src > lib > Header.svelte > script

```
<header class="flex justify-between items-center p-4 bg-white text-gray-800 shadow w-full fix
    <div class="flex items-center space-x-4">

      <div class="flex space-x-2">
        {#if $authStore.refreshToken === null}
        <button
          class="px-4 py-2 border border-gray-300 text-gray-600 hover:bg-gray-100 rounded"
          on:click={()=>{windowState.update(u => ({ ...u, type: windowStateType.LOGIN_SCREEN}))

        }}
        >
          Login
        </button>

        <button
          class="px-4 py-2 bg-gray-800 text-white hover:bg-gray-900 rounded"
          on:click={()=>{windowState.update(u => ({ ...u, type: windowStateType.REGISTER_SCREEN})
        >
          Register
        </button>
        {:else }
        <div class="flex items-center space-x-2">

          <img src="{$profileStore.avatar !== null ? $profileStore.avatar : '/default-avatar.pn
          <div class="flex flex-col items-center">
          <span class="text-xs text-gray-600">{"@"+$profileStore.username}</span>
          <span class="text-xs text-gray-400">{$profileStore.first_name} {$profileStore.last_
        </div>
        <button
          class="px-4 py-2 bg-red-600 text-white hover:bg-red-700 rounded"
          on:click={clearTokens}
        >
          Logout
        </button>
      </div>
      {/if}
      <button
        class="px-4 py-2 bg-gray-800 text-white hover:bg-gray-900 rounded"
        on:click={()=>{
            if (get(windowState).type === windowStateType.TEST_SCREEN) {
              windowState.update(u => ({ ...u, type: windowStateType.MY_PROJECTS}))
            }
            else {
              windowState.update(u => ({ ...u, type: windowStateType.TEST_SCREEN}))
            }
            console.log(get(profileStore));
```

Рисунок 3.28 – Компонент Svelte та Tailwind CSS

В коді на рисунку 3.29 продемонстровано спільне використання специфічних для tailwind класів та специфічних для svelte синтаксичних конструкцій, які потрібні для створення динамічного заголовку інтерфейсу системи.

```
app > src > lib > Login.svelte > script
<script>
    import { writable } from 'svelte/store';
    import { login } from '../services/api.js';
    import { setTokens } from '../stores/auth';

    const user = writable(null);

    let username = '';
    let password = '';

    async function loginUser() {
      try {
        const result = await login(username, password);
        user.set(result.user);

      } catch (error) {
        console.error('Login error:', error);
        // Handle login error here, e.g., show error message
      }
    }
  </script>

  <main class="flex flex-col items-left">
    <h1 class="text-3xl font-bold mb-4">Login</h1>
    <div class="bg-gray-100 rounded-lg p-4 shadow-md w-full max-w-md">
      <label class="block mb-2">
        <span class="text-gray-700">Username:</span>
        <input type="text" class="mt-1 block w-full px-3 py-2 border bor
      </label>

      <label class="block mb-2">
        <span class="text-gray-700">Password:</span>
        <input type="password" class="mt-1 block w-full px-3 py-2 border
      </label>

      <button class="mt-4 px-4 py-2 bg-gray-800 text-white hover:bg-
        Login
      </button>
    </div>
  </main>
```

Рисунок 3.29 – Компонент Svelte та Tailwind CSS

Іншим прикладом Svelte-компонента є login.svelte, де продемонстровано головна філософія Svelte: елемент, код який його обслуговує та його стилі

зберігаються в одному й тому ж місці, тому код залишається простим і інтуїтивно зрозумілим навіть у випадку збільшення кодової бази у десятки разів.

3.2.5 Реалізація бекенд-частини

Бекенд-сервер написаний за допомогою технологій Django, Django Rest Framework, Django cors headers, Django Rest Framework Simple JWT, Swagger.

Для ініціалізації було використано команди, які представлено на рисунку 3.30.

```
django-admin startproject project_management
cd project_management
python manage.py startapp accounts
```

Рисунок 3.30 – Ініціалізація проєкту

Після виконання команди було створено проєкт з структурою, яка представлена на рисунку 3.31.

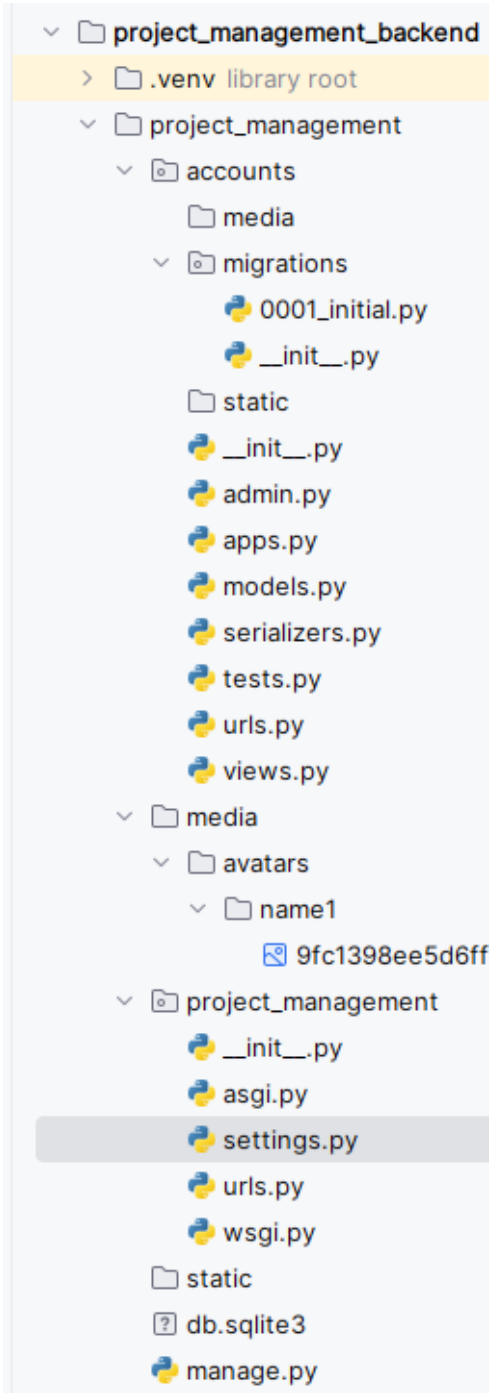

Рисунок 3.31 – Структура проєкту (бекенд)

В файлі models.py зберігаються представлення моделей класами Django. Після виконання команд з рисунку 3.32, у вибраній БД буде створено відповідні таблиці.

```
python manage.py makemigrations
python manage.py migrate
```

Рисунок 3.32 – Команди для міграції

Розглянемо приклад класу з файлу models.py, який представлено на рисунку 3.33.

models.py

```
30      class Meta:
31          unique_together = ['member', 'project']
32
33
        17 usages
34      class Profile(models.Model):
35          user = models.OneToOneField(settings.AUTH_USER_MODEL, related_name='profile',
36              on_delete=models.CASCADE,verbose_name='User',)
37          avatar = models.ImageField(upload_to=avatar_upload_path,blank=True,
38              null=True,verbose_name='Avatar',)
39          first_name = models.CharField(max_length=255,
40              verbose_name='First Name',)
41          last_name = models.CharField(max_length=255,
42              verbose_name='Last Name',)
43          projects = models.ManyToManyField( to: 'Project',
44              related_name='profiles',
45              blank=True,
46              verbose_name='Projects',
47          )
48          contacts = models.ManyToManyField(
49              to: 'self',
50              through=Contact,
51              through_fields=('from_profile', 'to_profile'),
52              verbose_name='Contacts',
53          )
54
55          class Meta:
56              verbose_name = 'Profile'
57              verbose_name_plural = 'Profiles'
58
59          def __str__(self):
60              return f"{self.first_name} {self.last_name}"
```

Рисунок 3.33 – Клас профілю

Для розширення вбудованого функціоналу профілів рекомендується створювати окремий клас, який буде однозначно посилатись (OneToOneField) на вбудовану модель користувача, в якому будуть вказані додаткові значення.

Також Django надає високорівневий функціонал, аби реалізовувати складні зв’язки між таблицями за допомогою ManyToManyField, ForeignKeyField та їх параметрам, які дозволяють створювати як симетричні, так і асиметричні зв’язки.

Наступним кроком є створення серіалізаторів за допомогою функціоналу DRF, який дозволяє перетворювати json-запити на об'єкти Django ORM і назад.

Для одного об’єкта можна створити довільну кількість серіалізаторів, чи писати свій власний серіалізатор, який представлено на рисунку 3.34.

```
3 usages
class UserProfileSerializer(serializers.Serializer):
    username = serializers.CharField(max_length=255)
    password = serializers.CharField(write_only=True, min_length=8)

    first_name = serializers.CharField(max_length=255)
    last_name = serializers.CharField(max_length=255)
    avatar = serializers.ImageField(required=False, allow_null=True)


    def validate_username(self, value):
        if User.objects.filter(username=value).exists():
            raise serializers.ValidationError("Username already exists.")
        return value

    def create(self, validated_data):
        username = validated_data.pop("username")
        password = validated_data.pop("password")
        user = User.objects.create_user(username=username, password=password)

        profile_data = {
            "first_name": validated_data.pop("first_name"),
            "last_name": validated_data.pop("last_name"),
            "avatar": validated_data.pop("avatar", None),
        }
        profile = Profile.objects.create(user=user, **profile_data)

        return profile
```

Рисунок 3.34 – Приклад користувацького серіалізатора

На рисунку зображено код складного серіалізатора, який спочатку створює об’єкт користувача, а потім прив’язує до цього користувача профіль.

Ці серіалізатори використовуються у класах представлень, які відповідають за логіку обробки запитів. Прикладом таких об’єктів є RegisterView, LoginView файлу views.py, які представлено на рисунку 3.35.

```
class RegisterView(generics.CreateAPIView):
    serializer_class = UserProfileSerializer
    permission_classes = [AllowAny]

    def post(self, request, *args, **kwargs):
        print(request.data)
        serializer = self.serializer_class(data=request.data)
        if serializer.is_valid():
            profile = serializer.save()

            username = request.data.get('username')
            password = request.data.get('password')

            user = authenticate(username=username, password=password)

            token_serializer = TokenSerializer()
            tokens = token_serializer.get_tokens_for_user(user)
            return Response(tokens|{

                    "username": profile.user.username,
                    "first_name": profile.first_name,
                    "last_name": profile.last_name,
                    "avatar": request.build_absolute_uri(profile.avatar.url) if profile.avatar else None,

            }, status=status.HTTP_201_CREATED)
        else:
            return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

Рисунок 3.35 – Представлення RegisterView

У цьому коді реалізується обробка POST-запиту і обробляється додавання користувача. Залишається визначити при запиті на який URL його буде обробляти дане представлення. Для цього існують файли urls.py нашого застосунку в рамках Django та urls.py всього проєкту. Розглянемо кожен з них на рисунку 3.36.

```
urlpatterns = [
    path('profile/', GetUserProfiles.as_view(), name='profile'),
    path('register/', RegisterView.as_view(), name='register'),
    path('login/', LoginView.as_view(), name='login'),
    path('', ProtectedResourceView.as_view(), name='resource'),

    path('projects/', ProjectListView.as_view(), name='project-list'),
    path('projects/<int:pk>/', ProjectDetailView.as_view(), name='project-detail'),
    path('projects/<int:project_id>/status/', StatusListView.as_view(), name='status-list'),
    path('projects/<int:project_id>/status/<int:pk>/', StatusDetailView.as_view(), name='status-detail'),
    path('projects/<int:project_id>/tasks/', TaskListView.as_view(), name='task-list'),
    path('projects/<int:project_id>/tasks/<int:pk>/', TaskDetailView.as_view(), name='task-detail'),
    path('contacts/', ContactListView.as_view(), name='contact-list'),
    path('contacts/<str:username>/', ContactCreateDeleteView.as_view(), name='contact-create-delete'),
    path('projects/<int:project_id>/members/', ProjectMemberListView.as_view(), name='project-member-list'),
    path('projects/<int:project_id>/messages/', ProjectMessageListView.as_view(), name='project-message-list'),
    path('projects/<int:project_id>/messages/<int:pk>/', ProjectMessageDetailView.as_view(), name='project-message-detail'),
    path('projects/<int:project_id>/files/', ProjectFileListView.as_view(), name='project-file-list'),
    path('projects/<int:project_id>/files/<int:pk>/', ProjectFileDetailView.as_view(), name='project-file-detail'),
    path('projects/<int:project_id>/tasks/<int:task_id>/messages/', TaskMessageListView.as_view(), name='task-message-list'),
    path('projects/<int:project_id>/tasks/<int:task_id>/messages/<int:pk>/', TaskMessageDetailView.as_view(), name='task-message-detail'),
    path('projects/<int:project_id>/tasks/<int:task_id>/files/', TaskFileListView.as_view(), name='task-file-list'),
    path('projects/<int:project_id>/tasks/<int:task_id>/files/<int:pk>/', TaskFileDetailView.as_view(), name='task-file-detail'),
]
```

Рисунок 3.36 – urls.py рівня застосунку

У цьому файлі кожному представленню у відповідність ставиться url, який він буде обробляти. Дані можна передавати у самій адресі запиту, тому існують рядки-шаблони по типу projects/<int:pk>/. Щодо urls.py рівня проєкту, то він виглядає як на рисунку 3.37.

urls.py

```python

import ...

schema_view = get_schema_view(
    openapi.Info(
        title="My API",
        default_version='v1',
        description="Описание вашего API",
    ),
    public=True,
    permission_classes=[AllowAny],
)


urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('accounts.urls')),
    re_path(r'^swagger(?P<format>\.json|\.yaml)$', schema_view.without_ui(), name='schema-json'),
    path('swagger/', schema_view.with_ui('swagger', cache_timeout=0), name='schema-swagger-ui'),
    path('redoc/', schema_view.with_ui('redoc', cache_timeout=0), name='schema-redoc'),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)

```

Рисунок 3.37 – Демонстрація urls.py рівня проєкту

У ньому зареєстровані усі Django-застосунки та вказані префікси, з якими вони будуть працювати, тобто структура загалом виглядатиме як адреса/префікс_застосунку/адреса_представлення_у_застосунку/.

Останній кроком залишається додавання у налаштування системи усіх застосунків і сторонніх бібліотек, які використані у розробці. Цей процес вказано на рисунку 3.38.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    'corsheaders',
    'rest_framework',
    'drf_yasg',

    'accounts',
]
```

Рисунок 3.38 – Встановлення застосунків

Як було сказано раніше, для спрощення процесу розробки, було встановлено Swagger, який ~~дозвоялє~~дозволяє легко слідкувати за реалізованими ендпоінтами та легко тестувати API. Цей процес показано на рисунку 3.39.

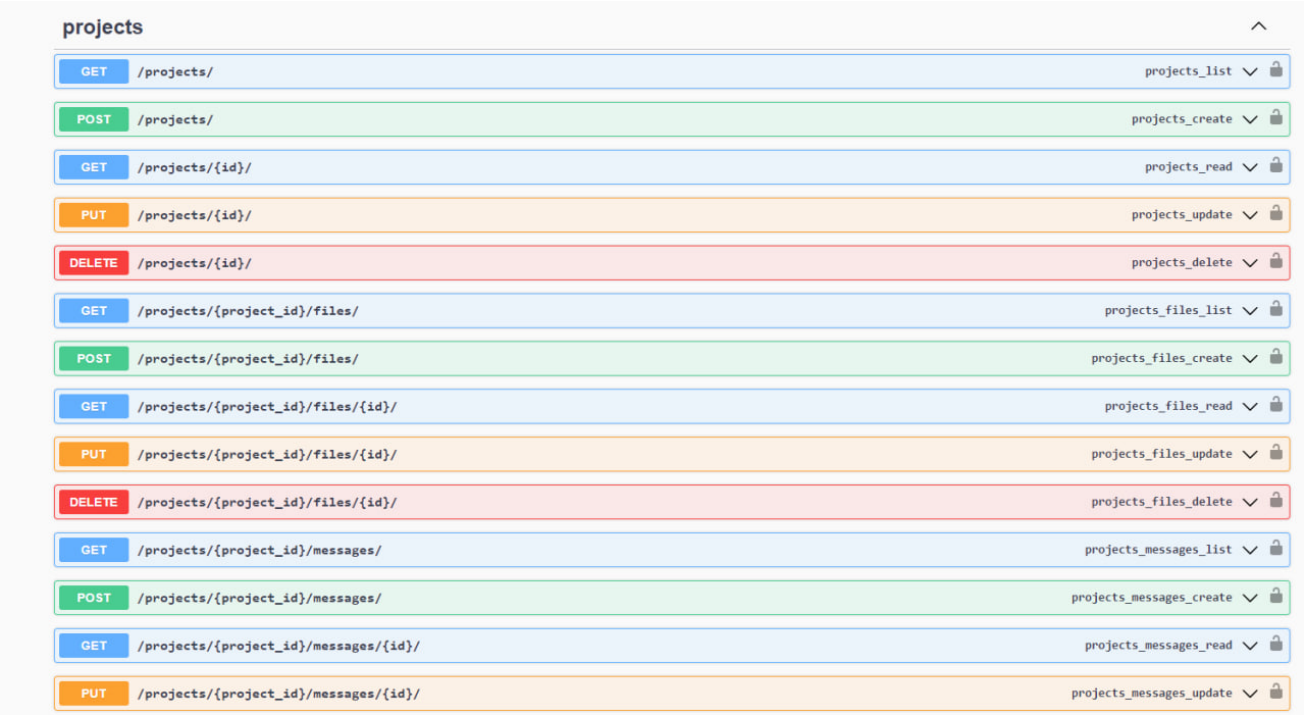

Рисунок 3.39 – Приклад використання Swagger

Висновки до третього розділу

Третій розділ кваліфікаційної роботи детально описує процес розробки застосунку, від моделювання системи до впровадження готового продукту.

На першому етапі було безпосереднє проектування застосунку, яке включало розробку архітектури системи, визначення технологічного стеку та створення детальних технічних завдань. Було обрано оптимальні технології та інструменти, які забезпечили стабільну та ефективну роботу програмного продукту. Особливу увагу приділено питанням безпеки даних та зручності користування, що є критично важливими аспектами для сучасних застосунків.

У процесі розробки були успішно реалізовані всі заплановані функціональні можливості, що включали інтуїтивно зрозумілий інтерфейс користувача, високий рівень продуктивності та надійності системи. Застосунок був протестований на різних етапах розробки, що дозволило виявити та виправити потенційні недоліки і забезпечити високий рівень якості кінцевого продукту.

Підсумовуючи, можна зазначити, що розроблений застосунок повністю відповідає поставленим вимогам і задачам. Він надає користувачам ефективний інструмент для вирішення конкретних завдань, покращуючи їх продуктивність та зручність роботи. Результати цього проекту демонструють значний потенціал для подальшого розвитку і вдосконалення, а також можуть бути використані як база для майбутніх досліджень та розробок у цій сфері.

Таким чином, розробка застосунку підтвердила актуальність обраного напрямку дослідження і зробила внесок у розвиток інформаційних технологій, забезпечуючи користувачів сучасними і надійними рішеннями для виконання завдань.

ВИСНОВКИ

У процесі написання цієї кваліфікаційної роботи було досліджено та впроваджено систему управління проєктами та контролю персоналу. Перш за все, проведений аналіз існуючих рішень на ринку, таких як Trello, Harvest, Asana, GanttPRO, Slack та Jira, дав змогу виявити їхні основні переваги та недоліки. Це дозволило створити власну вебсистему, яка поєднує в собі кращі риси аналізованих продуктів, але також враховує потреби і запити користувачів, що виникають у процесі управління проєктами.

Розглянуто технології та інструменти для розробки системи. А саме Python, Django, Django REST Framework, Swagger, Vite, Svelte та Tailwind CSS були обрані з огляду на їхні технічні можливості, продуктивність та зручність у використанні. Велика кількість переваг цих технологій дозволила створити необхідну систему швидко та легко.

Розроблена система включає функціонал для авторизації та реєстрації користувачів, створення та управління проєктами з використанням канбан-дошок, інтеграцію чатів та управління контактами. Такий комплексний підхід забезпечив високу функціональність та зручність у використанні, що дозволяє ефективно управляти проєктами та контролювати процес їх розробки.

На основі проведених досліджень та реалізації проєкту можна зробити висновок, що розроблена вебсистема значно спрощує процес роботи в команді. Вона поєднує в собі простоту використання, гнучкість та масштабованість, що робить її універсальним інструментом для різних типів користувачів, від студентів та викладачів до адміністраторів та менеджерів проєктів. Система також враховує сучасні вимоги до безпеки та продуктивності, що забезпечує її надійність та ефективність у довгостроковій перспективі.

Дана система є унікальною, що перекриває всі потреби, яку перед собою ставить користувач. Як приклад користувачів було обрано студентів працюючих над стартапом з необхідністю об’єднання легкого контролю за виконання

завданнями та комунікацією. І саме цей розроблений програмний продукт є ідеальним рішенням, яке поєднує в собі реалізацію всіх потреб.

Також розроблена система може бути успішно використана для автоматизації та оптимізації процесів управління проєктами та персоналом як над університетськими проєктами, так і в більш глобальних питаннях, що сприяє підвищенню продуктивності та якості роботи в організаціях. Цей проєкт демонструє можливості інтеграції різноманітних інструментів та технологій для досягнення високих результатів у сфері проєктного менеджменту.

Отже, спираючись на зроблено роботу, можна зробити висновок, що мета була досягнута і всі прославлені задачі були успішно виконані, а саме:

– було розглянуто та проаналізовано поняття менеджменту та основні етапами роботи над проєктом;

– проаналізовано сучасний стан задачі управління проєктами та персоналом;

– вивчено існуючі аналоги систем та визначено їхні переваги і недоліки;

– проаналізовано сучасні технології розробки вебсистем та обрано необхідні для розробки власного проєкту;

– розроблено архітектури системи та її компоненти;

– реалізовано функціонал системи відповідно до вимог користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Кеннет Мерчант, Вім Ван дер Стеде. Management Control Systems., Київ, 2016. — 928 с. (дата звернення: 13.05.2024).

2. Hoyer, V. Enterprise Mashups: Design Principles and Implementation Technologies. Springer, 2011. 330 p. (дата звернення: 14.06.2024).

3. Eisenberg, D. Svelte and Sapper in Action. Manning Publications, 2020. 440 p. (дата звернення: 15.05.2024).

4. Сагайдак Т. Ю., Поліщук В. В. Основи проєктного менеджменту. Київ: Наукова думка, 2020. 304 с. (дата звернення: 15.06.2024).

5. Проценко Д. В. Інструменти для проєктного менеджменту. Житомир: ЖДТУ, 2018. 198 с. (дата звернення: 13.05.2024).

6. Бондаренко І. Г., Коваленко Т. В. Управління проєктами: теорія і практика. Київ: КНЕУ, 2019. 256 с. (дата звернення: 13.06.2024).

7. Савчук О. Г. Програмні засоби для управління проєктами. Ужгород: УжНУ, 2018. 230 с. (дата звернення: 14.06.2024).

8. Грищенко О. В. Основи вебпрограмування. Київ: ВД "Київський університет", 2021. 256 с. (дата звернення: 15.06.2024).

9. Костюк Р. І. Сучасні фреймворки для розробки вебдодатків. Рівне: НУВГП, 2019. 182 с. (дата звернення: 17.05.2024).

10. Лисенко П. О. Управління проєктами в ІТ: теорія та практика. Чернівці: ЧНУ, 2018. 198 с. (дата звернення: 16.06.2024).

11. Науменко С. М. Вебсистеми для підтримки менеджменту. Запоріжжя: ЗНУ, 2017. 210 с. (дата звернення: 16.06.2024).

12. Сахно О. М. Програмні продукти для управління проєктами. Луцьк: ЛНТУ, 2018. 220 с. (дата звернення: 16.06.2024).

13. Жук І. О. Основи менеджменту в ІТ. Київ: ВД "Київський університет", 2017. 200 с. (дата звернення: 15.06.2024).

14. Сидоренко В. С. Управління проєктами в ІТ: методи та інструменти. Полтава: ПНТУ, 2019. 240 с. (дата звернення: 13.06.2024).

15. Тарасов О. Г. Вебтехнології для управління проєктами. Львів: ЛНУ, 2020. 300 с. (дата звернення: 15.05.2024).

16. Усенко Н. В. Розробка інформаційних систем для управління проєктами. Дніпро: ДНУ, 2018. 198 с. (дата звернення: 16.06.2024).

17. Шевченко В. В. Програмні засоби для веброзробки. Київ: КНЕУ, 2017. 290 с. (дата звернення: 15.06.2024).

18. Postman Learning Center. API Client and Design. URL: https://learning.postman.com/ (дата звернення: 22.06.2024).

19. The Best Project Management Software. URL: https://software.fish/project-management-software/best-project-management-software (дата звернення: 21.06.2024).

20. Trello. Офіційний сайт Trello. URL: https://trello.com/ (дата звернення: 18.06.2024).

21. Harvest. Офіційний веб-сайт Harvest. URL: https://www.getharvest.com/ (дата звернення: 18.05.2024).

22. Asana. Офіційний сайт Asana. URL: https://asana.com/ (дата звернення: 18.05.2024).

23. GanttPRO. Офіційний сайт GanttPRO URL: https://ganttpro.com/ (дата звернення: 18.05.2024).

24. Slack. Офіційний сайт Slack. URL: https://slack.com/ (дата звернення: 18.05.2024).

25. Jira. Офіційний сайт Jira. URL: https://www.atlassian.com/software/jira (дата звернення: 18.05.2024).

26. Django. Офіційна документація Django URL: https://docs.djangoproject.com/ (дата звернення: 20.05.2024).

27. Vite. Офіційна документація Vite. URL: https://vitejs.dev/ (дата звернення: 19.06.2024).

28. Svelte. Офіційна документація Svelte. URL: https://svelte.dev/ (дата звернення: 21.06.2024).

29. Tailwind CSS. Офіційна документація Tailwind CSS. URL: https://tailwindcss.com/ (дата звернення: 21.06.2024).

30. Swagger. Офіційна документація Swagger URL: https://swagger.io/ (дата звернення: 20.06.2024).