НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
(підпис)
"___" _____ 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 "Програмна інженерія"

на тему ВЕБ-СЕРВІС НАВЧАЛЬНИХ ПРОЕКТІВ В СФЕРІ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ З МОЖЛИВІСТЮ ВІДСТЕЖЕННЯ ПРОЦЕСУ РОЗРОБЛЕННЯ

Виконав:

студент 4 курсу, групи КП-51

Волощенко Олександр Євгенович _____
(підпис)

Керівник:

старший викладач кафедри ПЗКС Гадиняк Р.А. _____
(підпис)

Консультант з нормоконтролю:

доцент кафедри ПЗКС, к.т.н. Онай М.В. _____
(підпис)

Рецензент:

доц. каф. ММСА ІПСА, доц., к.т.н. Дідковська М.В. _____
(підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050103 «Програмна інженерія»

ЗАТВЕРДЖУЮ

Науковий керівник кафедри

_____ І.А. Дичка

«___» _____ 2018 р.

ЗАВДАННЯ

на дипломний проект студенту

Волощенку Олександру Євгеновичу

1. Тема проекту «Веб-сервіс навчальних проектів в сфері інформаційних технологій з можливістю відстеження процесу розроблення», керівник проекту Гадиняк Руслан Анатолійович, старший викладач, затверджені наказом по університету від «22» травня 2019 р. №1331-С.

2. Термін подання студентом проекту «__» червня 2019 р.

3. Вихідні дані до проекту: див. Технічне завдання.

4. Зміст:

- огляд існуючих рішень;
- обгрунтування вибору засобів реалізації;
- опис розроблених алгоритмів та підпрограм;
- аналіз розробленого рішення.

5. Перелік обов'язкового ілюстративного матеріалу:

- схема бази даних (креслення);
- діаграма прецедентів рішення (креслення);
- схема компонентів клієнтської частини (плакат);
- схема архітектури розробленого рішення (плакат).

6. Консультанти:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент, к.т.н.		

7. Дата видачі завдання «31» жовтня 2018 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Вивчення літератури за тематикою роботи	15.11.2018	
2.	Розроблення та узгодження технічного завдання	30.11.2018	
3.	Розроблення структури веб-сервісу	15.12.2018	
4.	Розроблення дизайну сторінок та графічних елементів	01.02.2019	
5.	Програмна реалізація веб-сервісу	15.03.2019	
6.	Тестування web-ресурсу	01.04.2019	
7.	Підготовка матеріалів текстової частини проекту	30.04.2019	
8.	Підготовка матеріалів графічної частини проекту	10.05.2019	
9.	Оформлення технічної документації проекту	25.05.2019	

Студент _____ О.Є. Волощенко

Керівник роботи _____ Р.А. Гадиняк

АНОТАЦІЯ

Дана дипломна робота присвячена розробці веб-сервісу навчальних проектів у сфері інформаційних технологій з можливістю відстеження процесу розробки.

Розроблене програмне забезпечення являє собою веб-сервіс, що складається з backend та frontend частин. Серверна частина складається з модулю маршрутизації, модулю моделей даних, модулю серіалізації та модулю з обробкою логіки запитів. Клієнтська частина складається з багатьох компонентів, що формують сторінки сервісу. Сторінки містять таку інформацію як список всіх проектів на сервісі, дані по конкретному проекту, список всіх технологій на сервісі та список всіх користувачів. В свою чергу сторінка проекту дозволяє переглядати інформацію про контрольні точки проекту, цілі контрольних точок, статуси та артефакти. Функціонал сервісу дозволяє авторизуватись на сервісі за допомогою стороннього сервісу Bitbucket, створювати нові проекти, формувати команди для проектів, створювати контрольні точки проектів та супутню інформацію.

У даному дипломному проекті розроблено: архітектуру сервісу, модуль взаємодії з базою даних, модуль з REST API, та frontend компоненти.

ABSTRACT

This diploma project is dedicated to the development of a web service of educational projects in the field of information technology with the ability to track the development process.

The developed software is a web service consisting of backend and frontend parts. The server part consists of a routing module, a data model module, a serialization module, and a query logic processing module. The client part consists of many components that form the service pages. Pages contain such information as a list of all projects on the service, data on a specific project, a list of all technologies on the service and a list of all users. In turn, the project page allows you to view information about project breakpoints, target points, statuses and artifacts. The functionality of the service allows you to log in to the service using a third-party Bitbucket service, create new projects, form teams for projects, create project points and related information.

In this diploma project developed: service architecture, the module of interaction with the database, the module with the REST API, and frontend components.

ДП.045440-01-90 Веб-сервіс навчальних проектів в сфері інформаційних технологій з можливістю відстеження процесу розроблення. Відомість проекту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проекту		
ДП.045440-02-91	Веб-сервіс навчальних	5	
	проектів в сфері		
	інформаційних технологій		
	з можливістю відстеження		
	процесу розроблення.		
	Технічне завдання		
ДП.045440-03-81	Веб-сервіс навчальних	59	
	проектів в сфері		
	інформаційних технологій		
	з можливістю відстеження		
	процесу розроблення.		
	Пояснювальна записка		
ДП.045440-04-51	Веб-сервіс навчальних	4	
	проектів в сфері		
	інформаційних технологій		
	з можливістю відстеження		
	процесу розроблення.		
	Програма та методика		
	тестування		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-05-34	Веб-сервіс навчальних	8	
	проектів в сфері		
	інформаційних технологій		
	з можливістю відстеження		
	процесу розроблення.		
	Керівництво користувача		
ДП.045440-06-99	Веб-сервіс навчальних	1	
	проектів в сфері		
	інформаційних технологій		
	з можливістю відстеження		
	процесу розроблення.		
	Функціональність		
	розробленої системи.		
	UML діаграма прецедентів		
ДП.045440-07-99	Веб-сервіс навчальних	1	
	проектів в сфері		
	інформаційних технологій		
	з можливістю відстеження		
	процесу розроблення.		
	Структура бази даних.		
	ERD діаграма		
ДП.045440-08-98	Веб-сервіс навчальних	1	
	проектів в сфері		
	інформаційних технологій		
	з можливістю відстеження		
	процесу розроблення.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___"_____ 2018 р.

ВЕБ-СЕРВІС НАВЧАЛЬНИХ ПРОЕКТІВ В СФЕРІ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ З МОЖЛИВІСТЮ ВІДСТЕЖЕННЯ ПРОЦЕСУ РОЗРОБЛЕННЯ

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проекту:

_____ Р.А. Гадиняк

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ О.Є. Волощенко

2018

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб-сервіс навчальних проектів в сфері інформаційних технологій з можливістю відстеження процесу розроблення.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання в якості інформаційного сховища даних навчальних проектів у сфері інформаційних технологій з метою збереження інформації про проекти та відстеження ходу розробки кожного з них.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Веб-сервіс повинен забезпечувати такі основні функції:

1. Сервіс має підтримувати наступні ролі: Адміністратор, Користувач, Гість.
2. Сервіс має підтримувати можливість реєстрації. Після успішної реєстрації користувач отримує роль Користувач. Незареєстровані користувачі мають роль Гість.
3. Адміністратори сервісу мають змогу змінювати ролі зареєстрованим користувачам.

4. Адміністратори мають право створювати проекти.
5. На проектах в команді є ролі Куратора і Розробника.
6. Куратори мають змогу додавати розробників до проекту.
7. Куратор проекту може видавати розробникам права на редагування проекту.
8. Проекти можуть мати наступні статуси: Ініціація, В процесі, Заморожено, Завершено.
9. Редагування артефактів проекту його учасниками.
10. Редагування логотипу та опису.
11. Редагування подій проекту (контрольних точок): майлстоунів, демонстрацій, релізів, тощо.
12. Надання користувачам можливість створення записів у блозі проекту.
13. Надання деяким користувачам статус “Помічник” за допомогу у розробці (порада, часткова реалізація, матеріальна допомога).
14. Підтримка доступу до бази IT проектів.
15. Прив’язка проектів до зовнішніх ресурсів (сайт, репозиторій, тощо).
16. Сервіс надає можливість матеріально підтримати проект.

Розробку виконати за допомогою мов програмування Python та JavaScript з використанням фреймворків Django на стороні серверу та Vue.js на стороні клієнта.

5. ВИМОГИ ДО ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

1. Пояснювальна записка.
2. Програма та методика тестування.
3. Керівництво користувача.
4. Креслення.

6. ЕТАПИ ПРОЕКТУВАННЯ

Вивчення літератури за тематикою роботи.....15.11.2018

Розроблення та узгодження технічного завдання 30.11.2018

Розроблення структури веб-сервісу 15.12.2018

Розроблення дизайну сторінок та графічних елементів 01.02.2019

Програмна реалізація веб-сервісу 15.03.2019

Тестування web-ресурсу 01.04.2019

Підготовка матеріалів текстової частини проекту.....30.04.2019

Підготовка матеріалів графічної частини проекту 10.05.2019

Оформлення технічної документації проекту.....25.05.2019

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ВЕБ-СЕРВІС НАВЧАЛЬНИХ ПРОЕКТІВ В СФЕРІ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ З МОЖЛИВІСТЮ ВІДСТЕЖЕННЯ ПРОЦЕСУ РОЗРОБЛЕННЯ

Пояснювальна записка

ДП.045440-03-81

"ПОГОДЖЕНО"

Керівник проекту:

_____ Р.А. Гадиняк

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ О.Є. Волощенко

2019

ЗМІСТ

ВСТУП

Галузь інформаційних технологій є однією з найперспективніших областей для розвитку в наш час. Кожного дня люди використовують ті чи інші застосунки в мобільних телефонах, на власних комп'ютерах чи навіть в мережі інтернет.

В сучасному світі інформаційних технологій успіх проекту в більшій мірі залежить від його планування та управління. Для успішного управління проектом було створено методології розробки програмного забезпечення. Загалом ці методології можна описати як засоби для структурування, планування та контролю процесу розробки програмного забезпечення. Таких методологій існує багато і кожна має свої характеристики. Набір цих характеристик визначає переваги та недоліки тої чи іншої методології розробки. Для ефективного управління проектами менеджер або команда розробників повинні знати декілька методологій аби вибрати ту яка найкраще підходить для розроблюваного проекту. Прикладами методологій розробки можуть бути XP, Kanban, тощо. Іноді для успішного контролю розробки команди можуть використовувати спеціальне програмне забезпечення.

Загалом таке ПЗ для командної розробки використовується як інструмент менеджменту і контролю виконуваної роботи в команді. Подібні інструменти завжди допомагають зрозуміти на якому етапі розробки зараз знаходиться той чи інший проект та допомагають поліпшити спільну роботу будь-яких команд над запланованими проектами.

Оскільки сфера IT стрімко розвивається – кожного дня може розроблюватись велика кількість проектів. Кожен з них вирішує якісь проблеми та переслідує певні цілі. Незалежно від методології розробки проекти завжди стикаються з певними проблемами, завжди діляться не певні контрольні точки та протягом процесу розробки створюють нові артефакти, що описують ту чи іншу частину проекту. Навіть

використовуючи однакові технології кожен окремий проект може використовувати їх по різному.

Виходячи з цього можна сказати, що рішення для командного розроблення програмного забезпечення є дуже актуальними в сучасному світі ІТ. Рішення, яке б надавало можливість збереження тих чи інших проектів, інформації про хід та планування їх розробки, тощо. Також важливим пунктом даного рішення є відстеження проектів та зв’язаної з ними інформацією. Наприклад відстеження статусів контрольних точок, нових цілей, нових артефактів чи записів у блозі, що може бути корисним як і команді проекту так і людям, яким цей проект сподобався.

Даний дипломний проект присвячений розробці web-сервісу, що призначений для зберігання та відстеження командних проектів у сфері інформаційних технологій та кроків їх розробки разом з супутньою інформацією.

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

4G – 4-те покоління мобільного зв'язку, яке дозволяє здійснювати передачу даних зі швидкістю, що перевищує 100 Мбіт / с;

ANSI – Американський національний інститут стандартів;

CSS – Cascading Style Sheets, каскадні таблиці стилів;

ECMAScript – стандарт мови програмування, затверджений міжнародною організацією ECMA;

FLOSS – Free/Libre and Open-Source Software, програмне забезпечення з відкритими вихідним кодом;

HTML – HyperText Markup Language, стандартизована мова розмітки документів в мережі;

JSON – JavaScript Object Notation, текстовий формат даних, заснований на JavaScript;

JSX – JavaScript XML, розширення синтаксису JavaScript, яке дозволяє використовувати схожий на HTML синтаксис для опису структури інтерфейсу;

MVC – Model-View-Controller, архітектурний шаблон, що використовується для розробки програмного забезпечення;

MVVM – Model-View-ViewModel, шаблон проектування архітектури додатку;

REST – REpresentational State Transfer, архітектурний стиль для розподілених систем в мережі, що для взаємодії використовує HTTP протокол;

SPA – Single Page Application, веб-додаток, який використовує єдиний HTML-документ як оболонку для всіх веб-сторінок і для взаємодії з користувачем динамічно підключає HTML, CSS, та JavaScript;

SQL – мова структурованих запитів, що застосовується для управління даними в реляційних базах даних;

TCP/IP – стек протоколів в мережі Інтернет;

UNIX – сімейство переносних, багатозадачних і багатокористувацьких операційних систем;

Wi-Fi – технологія бездротової локальної мережі;

XML – eXtensible Markup Language, мова розмітки;

XMLHttpRequests – API, який надає клієнту функціональність для отримання даних без перезавантаження сторінки. Використовується в AJAX запитах і особливо в single-page додатках.

Артефакти проекту – сукупність створених під час розробки програмного забезпечення елементів (документація, схеми, моделі, макети, тощо);

ІТ – інформаційні технології;

Майлстоун – контрольна точка в ході розробки проекту;

ООП – об’єктно орієнтоване програмування, методологія програмування, заснована на представленні програми у вигляді сукупності об'єктів;

ПЗ – програмне забезпечення;

СКБД – система керування базами даних.

1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1. Аналіз вимог до технологій командного розроблення програмного забезпечення

Сучасна ІТ індустрія розвивається шаленими темпами і кожного дня можуть створюватись нові команди, які як і інші команди плануються все нові і нові проекти, які потребують контролю та чіткої взаємодії між членами команд або і між командами в цілому. В залежності від методології у розробці програмного забезпечення можуть брати участь різні люди з різними ролями та повноваженнями, але у кожної з них можна простежити такі наступні ролі:

– замовник;

– інвестор;

– менеджер;

– команда розробників.

Кожен з перечислених зацікавлений в тому щоб завжди отримувати актуальну інформацію по програмному продукту і підтримувати її в актуальному вигляді. Окрім актуальної інформації команді також необхідно зберігати та відстежувати кроки розробки проекту разом з їх артефактами (наприклад схеми обміну даними, схеми баз даних, цілі та беклоги задач, тощо). Також командам потрібно зберігати та описувати особливості підходів вирішення проблем, що може бути корисним як для наступних розробників проекту, або для ІТ спільноти в цілому.

Загалом для подібного програмного забезпечення стандартною практикою в галузі ІТ виділяються наступні важливі вимоги:

– можливість створення проектів та їх поділ на задачі;

– виставлення пріоритетів проектам та задачам;

– можливість назначати виконавців проектів чи задач;

– виставлення часових рамок для проектів та задач;

– поділ задач на підзадачі;

– відстеження виконання проектів та задач;

– можливість прикріплювати до завдань артефакти, які прояснюють ціль або процес виконання.

Також користувачі подібних сервісів надають перевагу зручним та зрозумілим інтерфейсам, можливостям підключати сторонні сервіси (наприклад Git репозиторії команд чи календарі), гнучким способам розроблення структур проектів, присутність трекерів часу і тому подібне.

Виходячи з вимог IT галузі до подібного програмного забезпечення можна виділити наступні функціональні вимоги до розроблюваного дипломного проекту:

1. Сервіс має підтримувати наступні ролі: Адміністратор, Користувач, Гість.
2. Сервіс має підтримувати можливість реєстрації. Після успішної реєстрації користувач отримує роль Користувач. Незареєстровані користувачі мають роль Гість.
3. Адміністратори сервісу мають змогу змінювати ролі зареєстрованим користувачам.
4. Адміністратори мають право створювати проекти.
5. На проектах в команді є ролі Куратора і Розробника.
6. Куратори мають змогу додавати розробників до проекту.
7. Куратор проекту може видавати розробникам права на редагування проекту.
8. Проекти можуть мати наступні статуси: Ініціація, В процесі, Заморожено, Завершено.
9. Редагування артефактів проекту його учасниками.
10. Редагування логотипу та опису.
11. Редагування подій проекту (контрольних точок): майлстоунів, демонстрацій, релізів, тощо.

12. Надання користувачам можливість створення записів у блозі проекту.
13. Надання деяким користувачам статус “Помічник” за допомогу у розробці (порада, часткова реалізація, матеріальна допомога).
14. Підтримка доступу до бази IT проектів.
15. Прив’язка проектів до зовнішніх ресурсів (сайт, репозиторій, тощо).
16. Сервіс надає можливість матеріально підтримати проект.

1.2. Аналіз існуючих програмних рішень

Під час проведення дослідження літератури та огляду готових рішень було виділено кілька проектів які за своєю концепцією та функціональністю є схожими на заплановану програмну розробку цього дипломного проекту. Зважаючи на цей факт запропоновано переглянути ці готові рішення, адже деякі їх функціональні можливості реалізовані в розроблюваному дипломному проекті.

1.2.1. *Asana*

Одним із найпопулярніших і часто використовуваних сервісів для командної розробки програмного забезпечення є Asana. Asana це веб-сервіс для менеджменту та управління командними розробками, що дозволяє сконцентруватись на цілях, проектах та щоденних задачах, які роблять бізнес клієнта успішним. Платформа дозволяє планувати та структурувати хід роботи таким чином, яким це більш ефективно підходить бізнесу та створюваним проектам. Надається можливість створення задач, моніторингу їх виконання, призначення відповідних задач відповідальним членам команди з виставленими пріоритетами та встановленими дедлайнами [1]. На основі встановлених задачам дедлайнів створюється також календар який допомагає зрозуміти етап та час розробки потрібного програмного забезпечення. Описані вище дані дозволяють прогнозувати

стратегію розробки, поширення та розвитку програмного забезпечення, яке розробляється.

На рис. 1.1 зображено головну сторінку одного з робочих просторів в Asana. Тут можемо спостерігати список всіх завдань, їх дедлайни та аватари співробітників на яких ці завдання назначені. Панель зліва дозволяє перейти до домашньої директорії робочого простору, назначених на користувача завдань та повідомлень з завдань на які підписаний користувач.

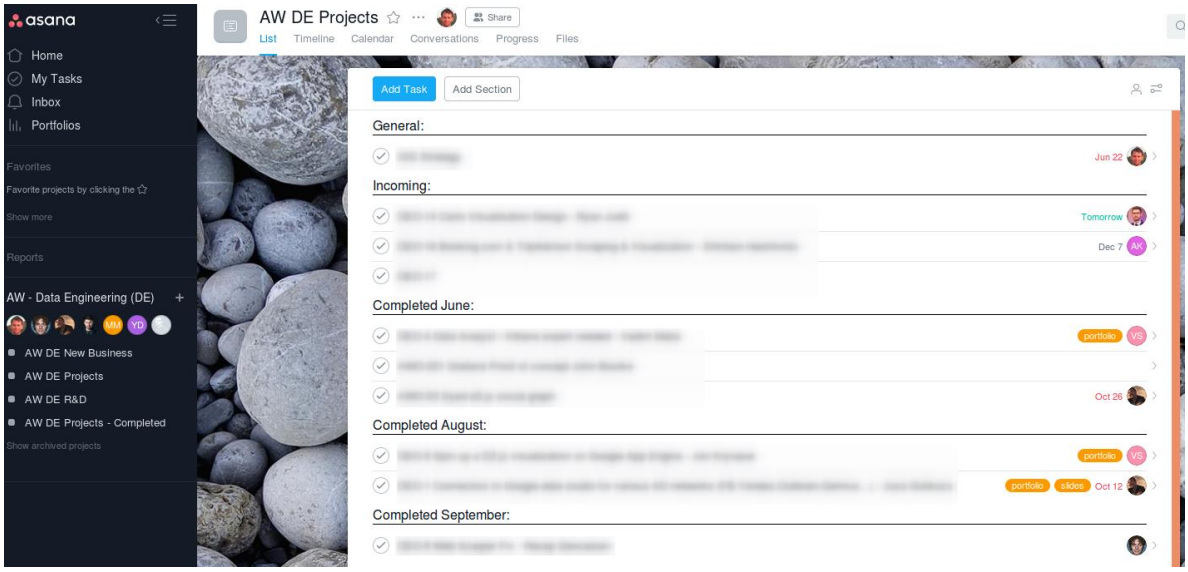


Рис. 1.1. Вигляд сторінки робочого простору у веб інтерфейсі Asana

Переваги та можливості [2]:

– створення завдань та необхідних для них параметрів;

– добре підходить для невеликих проектів;

– можливість довільно описувати задачі та прикріпити необхідні матеріали;

– відносно проста та зрозуміла функціональність;

– швидкодія;

– непоганий мобільний додаток.

Недоліки:

– при необхідності отримувати повідомлення щодо проектів на електронну пошту – сервіс може надсилати багато зайвих листів з непотрібною інформацією;
– іноді структура повідомлень в поштових листах не є зручною і може не відображати потрібну інформацію;
– неможливо побачити абсолютно всі призначені на користувача завдання в одному місці, для цього потрібно перемикатись між робочими просторами;
– не підходить для масштабних проектів.

Спираючись на переваги та недоліки Asana можна сказати, що розглянутий сервіс є одним із лідерів у сфері командного створення програмного забезпечення. Однак функціональність розглянутого сервісу заточена більше під опис та виконання конкретних задач, а не проектів, їх опису та контрольних точок.

1.2.2. *Projectplace*

Наступним наближено подібним до запланованого програмного забезпечення продуктом можна вважати Projectplace. Projectplace – програмне рішення для спільної роботи команд, який допомагає їм якісніше проводити комунікації, слідкувати за задачами, їх статусами та за проектами в цілому [3]. Загалом веб-сервіс має схожий з Asana функціонал. Основний інтерес для даної дипломної розробки полягає в понятті “проект” та можливість створення плану з контрольними точками для цього проекту. Користувач має можливість створювати майлстоуни для конкретного проекту, задавати їм часові рамки та опис. Крім цього також маємо можливість ділити контрольні точки на підзавдання які своєю структурою та функціями схожі на дерева.

На рис. 1.2 можемо спостерігати як після створення необхідних проміжних точок проекту створюється календар на якому чітко видно коли

заплановано початок і кінець завдань. Надається можливість команді коментувати та доповнювати вже існуючі завдання не виходячи з інтерфейсу календаря.

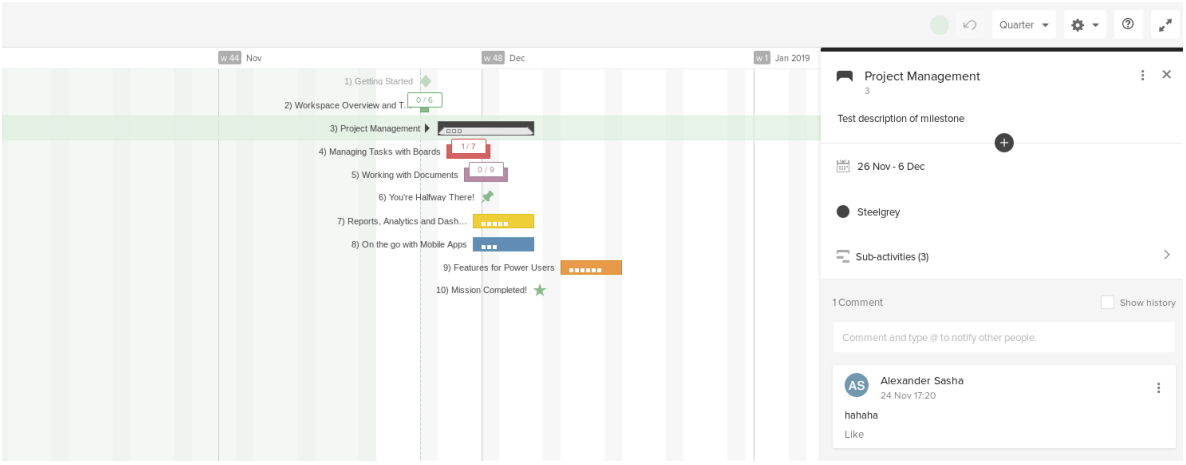


Рис. 1.2. Інтерактивний календар з контрольними точками проекту та їх описом у Projectplace

Можна виділити наступні переваги та функції сервісу [4], [5]:

– можливість створення різних робочих просторів та проектів;
– поділ проектів на контрольні точки;
– інтерактивний календар, що відображає дати запланованих на виконання або вже виконаних завдань/контрольних точок;
– можливість створення kanban дошок для робочих просторів чи проектів;
– можливість комунікацій між членами команди через чат на сервісі.

Недоліки:

– загромадження календарного плану на інтерфейсі при великій кількості завдань;
– неможливо прикріплювати до завдань потрібні матеріали, можливо лише для коментарів, що ускладнює пошук;

– не підходить для малих команд чи проектів, адже має багато функціональності яка не буде ними використовуватись, але за яку доведеться платити.

Отже, роблячи висновки щодо розглянутого веб-сервісу можна сказати, що Projectplace має необхідний функціонал для планування та розробки програмного забезпечення та був розглянутий як повноцінний аналог розроблюваного в даній дипломній роботі програмного забезпечення.

1.2.3. *Kickstarter*

Ще одним подібним до запланованого веб-сервісу можна вважати платформу Kickstarter. Kickstarter – платформа для фінансування творчих проектів людей по всьому світу. Кожен проект може відноситись до будь-якої сфери, будь то кіно, ігри, музика, творчість, дизайн чи навіть технології. На Kickstarter можна знайти дуже багато амбіціозних, творчих та інноваційних ідей кожна з яких може стати реальною завдяки іншим людям, яким можуть ці самі ідеї сподобатись. Кожна ідея на платформі має бути проектом з чітко поставленою ціллю, наприклад музичний альбом, книга чи якийсь програмний продукт. Проект розглядається як кінцева робота з чіткою ціллю, мета розміщення на сервісі цього проекту – зібрати необхідну кількість коштів для реалізації ідеї [6]. Інвестором може бути будь хто, і якщо проект буде успішно виконаний – цей дехто може отримати якийсь унікальний бонус від розробників.

Для запланованого програмного забезпечення в цьому дипломному проекті Kickstarter цікавий через підхід до роботи з проектами на сервісі. На рис. 1.3 можна спостерігати головну сторінку одного з проектів на платформі Kickstarter. Тут вказується сума яку проект вже зібрав, кількість людей яким ідея сподобалась та дні коли інвестиція в проект буде ще доступна. В будь-який момент можна задати питання розробнику проекта, подивитись плани та етапи розробки, також надається можливість

прокоментувати роботу автора. Саме ці функції платформи є важливими для розроблюваного програмного забезпечення.

Деякі переваги та функціонал сервісу [7]:

– розгорнутий опис до проектів;
– перелік виконаних та виконуваних контрольних точок на проекті;
– безкоштовне розміщення будь-яких проектів та інвестиції в них;
– вимоги до чіткого опису проектів та чітко поставлених цілей проекту.

Деякі недоліки:

– обмеження на створення проектів по країнам;
– неможливість гнучкості на проекті, розробник має виконати саме те що було зазначено та сплановано перед публікацією на платформі.

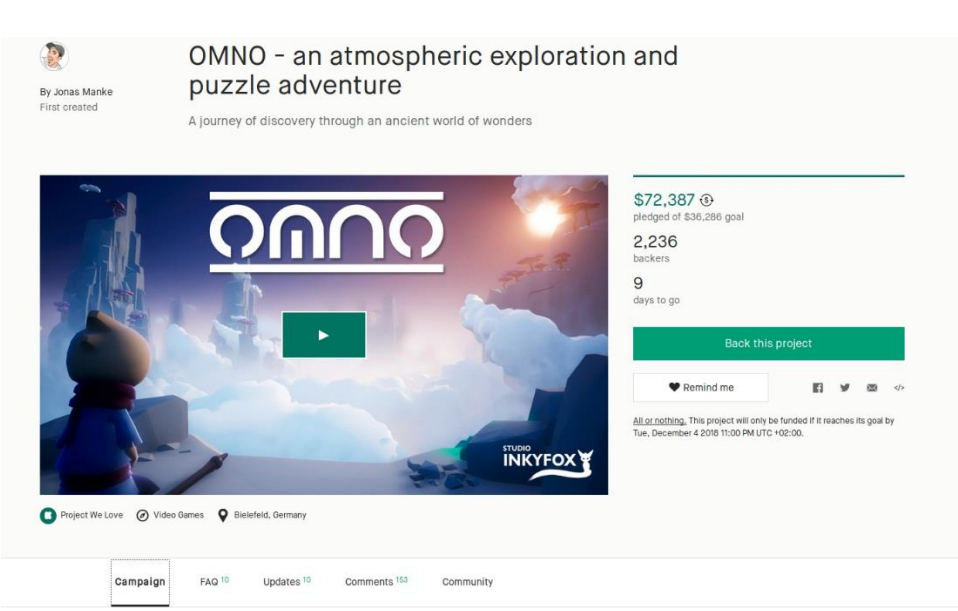


Рис. 1.3. Головна сторінка проекту на платформі Kickstarter

1.2.4. *Patreon*

Patreon – це онлайн сервіс, який дозволяє будь-кому фінансово підтримувати своїх улюблених творців. Це дозволяє так званим творцям

зв'язуватися зі своїми шанувальниками і отримувати дохід від своєї роботи. Patreon надає творцям більш стійке джерело доходу. Patreon не зосереджується на одноразовому проекті. На відміну від Kickstarter, Patreon прагне створити дохід у довгостроковій перспективі. Це відмінний спосіб монетизувати облікові записи YouTube, Instagram або інших соціальних медіа.

Patreon надає повноваження творцям, надаючи їм інструменти, необхідні для придбання, управління та активізації своїх патронів (підписників). За допомогою платіжної моделі підписки, фанати платять своїм улюбленим творцям щомісячну суму за свій вибір в обмін на ексклюзивний доступ, додатковий контент або ближчий погляд на їх творчу подорож. На думку розробників сервісу ця модель є безпрограшною; творці зберігають творчу свободу, отримуючи заробітну плату, яку вони заслуговують, і шанувальники спокійно відпочивають, знаючи, що їхні гроші йдуть безпосередньо на створення більше того контенту, що вони люблять.

Загалом Patreon для творців – це спосіб отримати гроші за створення речей, які вони вже створюють (наприклад відео, пісні, комікси, тощо). Шанувальники платять кілька доларів на місяць або за кожне повідомлення, яке випускають творці, а потім щомісяця творець отримує «зарплату», або кожен раз, коли випускає щось нове [9].

Patreon для патронів – це спосіб приєднатися до спільноти улюблених творців. Тепер користувачі можуть заплатити кілька доларів на місяць або за пост, який створює творець. Наприклад, якщо патрон сплачуєте 3 долари за відео, і творець випускає 4 відео у березні, то з картки патрона стягується загальна сума $12 цього місяця. Це означає, що творцю платять регулярно (кожен раз, коли він випускає щось нове) [9].

Переваги сервісу [10]:

– можливість матеріально підтримати творців, що розробляють контент який подобається шанувальникам;

– можливість патронів регулярно підтримувати творців;

– можливість відміни підписок з поверненням коштів.

Також користувачі сервісу виділяють ще і такі переваги як постійно зростаюче товариство платформи, якісна підтримка та сервіс.

Деякі недоліки:

– неможливо підписатись на того чи іншого творця безкоштовно просто для того щоб відстежувати діяльність, мова не йде про бонуси які надаються патронам;

– неможливо підтримати окремо взятий проект того чи іншого творця.

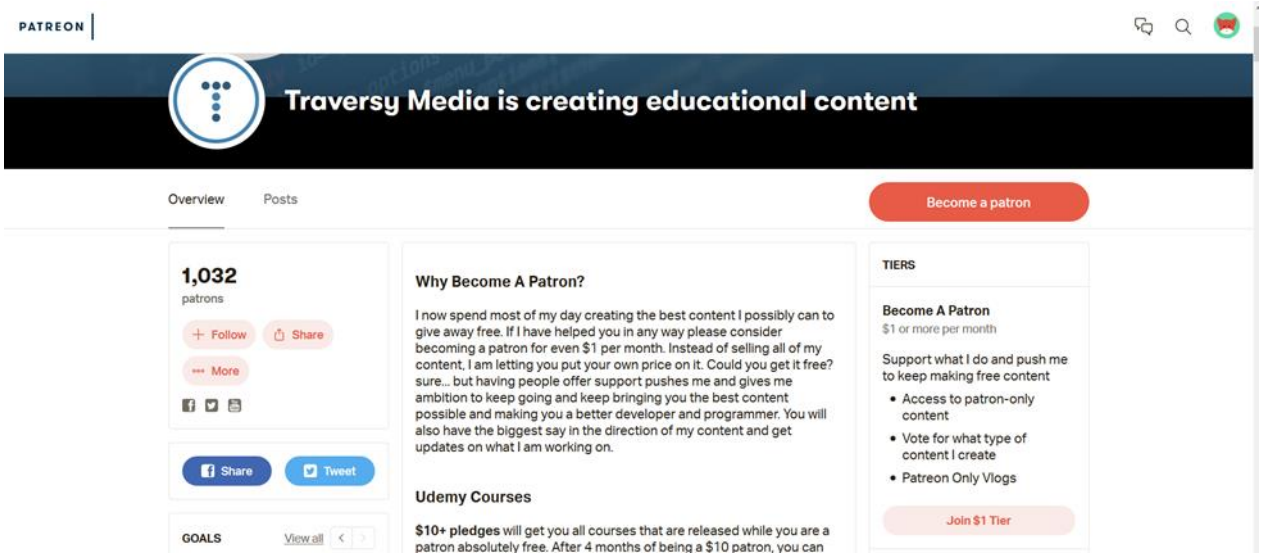


Рис. 1.4. Сторінка профілю творця на платформі Patreon

1.3. Результати аналізу

Звичайно також існує велика кількість подібних сервісів як і до перших двох, так і для третього. Для Asana та Projectplace такими сервісами можуть бути платформи Jira, Trello, Phabricator та інші. Всі вони схожі за функціоналом та цілями використання, але кожен може надати свої можливості які зроблять сервіс унікальним.

Розглянутий Kickstarter також має свої аналоги, наприклад Indiegogo або Fundly. Загалом ці платформи були створені та використовуються в якості краудфандингових платформ і до світу програмного забезпечення відносяться лише розміщуваними на цих сервісах продуктами які тісно пов’язані з IT індустрією. Для розробки програмного забезпечення даної дипломної роботи на цих сервісах досліджено функціоналіньсть для роботи з проектами. Насамперед важливо звернути увагу на підходи роботи та оформлення проектів, які розміщуються на цих платформах. Наприклад підхід для опису проекту, демонстрування його проміжних точок, можливість звичайних користувачів моніторити та задавати питання щодо проекту чи статусу його розробки.

В результаті порівняння вище описаних інструментів з необхідними параметрами отримано наступну таблицю (табл. 1.1).

Таблиця 1.1

Порівняння функціональностей можливих аналогів

Функціональність	Аналоги			
	Asana	Projectplace	Kickstarter	Patreon
1	2	3	4	5
Створення окремо взятих проектів	+	+	+	–
Можливість поділу проектів на події	+	+	+	–
Можливість формування команд	+	+	–	–
Наявність блогу проекту	–	–	+	–

Продовження табл. 1.1

1	2	3	4	5
Редагування артефактів проекту	+	+	–	–
Можливість надати допомогу розробникам	–	–	+	+
Відстеження статусів контрольних точок проектів	+	+	+	–

2. АНАЛІЗ МОВ ПРОГРАМУВАННЯ ТА ТЕХНОЛОГІЙ РОЗРОБЛЕННЯ WEB-САЙТІВ

2.1. Порівняння web-додатків та desktop-додатків

Найбільшою перевагою desktop додатків є те, що вони можуть бути добре оптимізовані та швидко взаємодіяти з апаратним забезпеченням машини на якій вони працюють, при цьому апаратне забезпечення є і основним мінусом настільних додатків. Якщо потрібно розробити додаток, що буде працювати на багатьох платформах – треба бути готовим до збільшення часу розробки. Такі додатки можуть мати суворі апаратні вимоги для нормального функціонування. Ця апаратна залежність, як правило, обмежує рівень складності інтерфейсів користувача для desktop додатків. Користувачі, які завантажують desktop додатки, найчастіше бувають активними користувачами цих програм. Це може пояснюватись тим, що такі додатки можуть підтримувати функціональність, яка може бути не доступна для web версій, або функціональність, яка може стабільно підтримуватись навіть у режимі офлайн [11]. До таких програм можна віднести Spotify, Slack та подібні.

У деяких випадках web додатки являють собою модель клієнт-сервер. Користувач звертається до програми за допомогою веб-браузера, який фактично і є клієнтом, і працює з ресурсами, доступними в Інтернеті, включаючи обчислювальну та процесорну потужність. Такий підхід дозволяє машинам з обмеженими апаратними можливостями надавати доступ до складних додатків, що надходять з централізованої інфраструктури. Крім того, використання існуючих веб-браузерів та їх мультимедійних можливостей дозволило розробникам створювати більш інтерактивні, багатофункціональні інтерфейси користувача. Деякі з цих можливостей також були відновлені для desktop додатків, але вони були в

значній мірі обумовлені повсюдністю Інтернету та способом, яким користувачі звикли взаємодіяти зі своїми комп'ютерами.

Хоча і desktop, і web застосунки мають свої плюси і мінуси, вони зрештою є лише інструментами, які люди використовують для вирішення проблем. Наприклад, ви можете працювати з електронними таблицями на своєму комп’ютері за допомогою програми Microsoft Excel або за допомогою веб-сервісу, як-от Google Documents. Обидва вони дозволяють здійснювати базове редагування електронних таблиць, але для роботи Google потрібне підключення до Інтернету. Щоб усунути цей недолік, деякі веб-програми розробили автономні можливості, які дозволяють почати роботу в Інтернеті, а потім продовжувати працювати пізніше, навіть якщо ви відключені від Інтернету. Деякі desktop програми також використовують технології, які спочатку використовувались для створення web додатків. Наприклад, розробники програмного забезпечення можуть використовувати HTML і JavaScript для розробки desktop додатків, а також web додатків. Це приклад того, як ці два типи додатків можуть запозичувати функції один одного. Можна бачити, що кожен тип програми має свої переваги і слабкості і може бути використаний найкраще у своїй області. Якщо взяти до уваги описані вище плюси та мінуси використання desktop та web додатків – для реалізації даного дипломного проекту вирішено використовувати web додатки. Оскільки в сучасному світі проблем з Інтернет підключенням майже відсутні завдяки технологіям 4G та Wi-Fi, то можна з впевненістю сказати, що незважаючи на те що головною реалізацією дипломного проекту буде веб додаток, доступність рішення не буде проблемою. Також таким чином ми позбавляємось необхідності готувати рішення для окремих платформ та підлаштовуватися під особливості апаратної реалізації кожної (Windows, Linux, Mac, …). Можливим недоліком може стати доступність та реалізація певної функціональності в різних браузерах. Загалом web додатки легше підтримувати та вони можуть бути сумісні навіть з мобільними пристроями.

2.2. Аналіз та порівняння систем керування базами даних

2.2.1. *Загальний огляд баз даних та вибір моделі*

У світі технологій баз даних існують дві найпопулярніших моделі баз даних: SQL та NoSQL бази даних. SQL бази даних також називають реляційними, а NoSQL відповідно не реляційними. Різниця між ними складається в тому як вони спроектовані, як зберігають інформацію та які типи даних вони підтримують.

Реляційні бази даних зберігають структуровані дані, що зберігаються у вигляді таблиць, формат яких визначений ще на початку проектування сховища. Кожна таблиця складається з стовбців та рядків. Записи в сховищі представляють собою рядки таблиці. В свою чергу атрибути даних являють собою стовпчики даних таблиць. Для того щоб взаємодіяти з реляційними базами даних існує спеціальна мова – SQL. Це стандартна мова для систем керування реляційними базами даних. Оператори SQL використовуються для виконання таких завдань, як оновлення даних у базі даних або вилучення даних з бази даних. Хоча більшість систем баз даних використовують SQL, більшість з них також мають свої власні додаткові розширення, які зазвичай використовуються тільки в їхній системі . Тим не менш, стандартні команди SQL є всюди і можуть бути використані для виконання майже всього, що потрібно робити з базою даних.

Переваги реляційних баз даних [12]:

– швидкість – швидкість, яку пропонує SQL, неймовірна та допомагає легко отримувати дані з записів бази даних;
– стандартизація – SQL відповідає стандартам ISI і ANSI, які затверджені в усьому світі;
– підтримка операцій на основі теорії множин – в основному реляційні бази базуються на реляційній теорії множин. Реляційна база даних підтримує реляційну алгебру, відповідно підтримуючи реляційні операції теорії множин, а саме об'єднання, перетину, різниця та декартові добутки, реляційні

бази даних також підтримують операції вибору, реляційного з'єднання та поділу;

– прості операції та визначені відносини – реляційні бази даних використовують SQL, який є простою і зрозумілою для людини мовою. Більш того, реляційні бази даних встановлюють визначені взаємозв'язки між таблицями, що дає користувачам повну картину збережених даних;
– безпека – реляційні бази підтримують права доступу, підтримують концепцію користувачів і їх прав;
– ACID – транзакції завжди атомні, послідовні, ізольовані і довговічні. Якщо виникає помилка, вся транзакція може бути повернута назад, відновлюючи попередній стан бази даних;
– вбудований інструментарій – тригери, процедури та індекси.

Недоліки реляційних баз даних:

– продуктивність – при нормалізованій схемі необхідно використовувати декілька об'єднань таблиць для збирання необхідних даних. Можна денормалізувати схему, тим самим зменшити кількість об'єднань і підвищити продуктивність. Однак це призводить до надмірності та небезпеки невідповідності, яка повинна оброблятися на рівні коду;
– масштабування – реляційні бази даних добре масштабуються вертикально. Але вертикальне масштабування обмежено фізично і бюджетом;
– “еволюція” схеми бази даних – при використанні реляційних баз даних необхідно визначити структуру моделі даних. Ця фіксована і сувора структура забезпечує безпеку, але потім цю структуру важко змінити;
– невідповідна модель даних для певних систем – модель розроблюваної системи може мати складну схему представлення даних, наприклад граф, обхід графу за допомогою інструментів

реляційних баз даних може стати проблемою через надзвичайну велику кількість з’єднань.

NoSQL є базами даних, які забезпечують механізм для зберігання і пошуку даних. Ці дані моделюються засобами, відмінними від табличних відносин, що використовуються в реляційних базах даних. NoSQL бази даних використовуються в web додатках, в big data і їх використання постійно зростає. Системи NoSQL також іноді можуть підтримувати SQL-подібні запити. Бази даних NoSQL включають в себе простоту проектування та більш просте горизонтальне масштабування для кластерів. Структури даних, що використовуються базами даних NoSQL, відрізняються від тих, що використовуються за замовчуванням у реляційних базах даних, іноді навіть гнучкішими, що робить деякі операції швидше в NoSQL. Використання баз даних NoSQL залежить від проблеми, яку вони повинні вирішувати. Перешкодами для більшого застосування NoSQL баз даних є використання мов запитів низького рівня, відсутність стандартизованих інтерфейсів і величезні попередні інвестиції в існуючі реляційні бази даних. Більшість NoSQL баз даних позбавлені справжніх транзакцій ACID, але кілька баз даних зробили їх центральними інструментами для своїх конструкцій.

Переваги NoSQL [13]:

– масштабованість – бази даних NoSQL використовують шардінг (sharding) для горизонтального масштабування, яке доступніше за вертикальне;
– доступність – функція автоматичної реплікації, у випадку будь-яких сигналів про відмову – відбувається реплікація до попереднього узгодженого стану.

Недоліки NoSQL:

– вузький фокус – мають дуже вузьку спрямованість та забезпечують дуже мало функціональних можливостей;

– open source – найчастіше NoSQL це бази даних з відкритим кодом, а отже немає надійного стандарту для NoSQL;
– недоступність графічного інтерфейсу (GUI) – інструменти GUI для доступу до NoSQL баз даних не є дуже доступними на ринку;
– великий розмір документів – деякі системи зберігають дані у форматі JSON, тому документи можуть бути досить великі.

Отже виходячи з вище описаних характеристик SQL та NoSQL баз даних можна скласти наступну таблицю (табл. 2.1) порівняння.

Таблиця 2.1

Порівняння SQL та NoSQL баз даних

SQL	NoSQL
Relational DBMS	DBMS
Мають фіксовану або статичну або попередньо визначену схему	Мають динамічну схему
Не підходять для ієрархічного зберігання даних	Найкраще підходять для ієрархічного зберігання даних.
Найкраще підходять для складних запитів	Не дуже підходять для складних запитів
Вертикальна масштабованість	Горизонтальна масштабованість

Виходячи з наведених вище фактів та з ймовірної схеми даних розроблюваного дипломного проекту прийнято рішення використовувати реляційні бази даних для зберігання інформації сервісу. Далі описано аналіз різних СКБД опираючись на їх переваги, недоліки та можливості, які найбільше підходять для розроблюваного програмного продукту.

2.2.2. *Аналіз СКБД MySQL*

MySQL – система управління реляційними базами даних з відкритим вихідним кодом, базується на SQL. Написана на C і C ++ і доступна у більш ніж 20 платформах, включаючи Mac, Windows, Linux і Unix. Може використовуватися в широкому діапазоні додатків, але найчастіше асоціюється з веб-додатками.

MySQL базується на моделі клієнт-сервер. Ядром MySQL є MySQL сервер, який оброблює всі інструкції (команди) бази даних. Сервер MySQL доступний як окрема програма для використання в клієнт-серверному мережевому середовищі і як бібліотека, яка може бути вбудована в окремі програми. MySQL працює разом з декількома утилітами, які підтримують адміністрування баз даних MySQL. Команди надсилаються MySQLServer через клієнт MySQL, який встановлюється на комп'ютері [14].

MySQL підтримує великі бази даних з мільйонами записів і підтримує безліч типів даних, включаючи цілі числа, числа з плаваючою крапкою, формати дат і тому подібні. Також підтримуються типи рядків з фіксованою і змінною довжиною.

Для забезпечення безпеки MySQL використовує привілеї доступу і систему зашифрованих паролів. Клієнти можуть підключатися до MySQL Server за допомогою декількох протоколів, включаючи TCP/IP сокети на будь-якій платформі.

MySQL підтримує розгортання в середовищах таких як Amazon RDS і Amazon Aurora для MySQL.

Переваги MySQL:

– популярність і простота використання – одна з найпопулярніших систем, немає дефіциту адміністраторів з досвідом. Існує велика кількість документації про те, як встановити та керувати базою даних MySQL;

– безпека – MySQL встановлюється за сценарієм, що встановлює рівень безпеки, визначивши пароль для користувача root. Крім

того MySQL підтримує управління користувачами і дозволяє надавати права доступу;

– швидкість – при розробці було опущено певні функції SQL, зо дало перевагу швидкості.

Недоліки MySQL:

– деякі обмеження – СКБД спочатку розроблялась для швидкості та простоти використання, а не для повної відповідності SQL, вона поставляється з певними функціональними обмеженнями;

– ліцензія – MySQL це ПЗ з двома ліцензіями, з відкритим вихідним кодом і декількома платними комерційними виданнями. Через це деякі функції та плагіни доступні лише для обмеженої кількості корпорацій;

– уповільнений розвиток – MySQL купувалась декілька разів різними компаніями, з'явилися скарги від користувачів, що процес розробки для значно сповільнився, оскільки співтовариство більше не має здатності швидко реагувати на проблеми і вносити зміни;

– втрата продуктивності при великих розмірах даних – коли дані зростають, тільки застосування індексів дають хорошу продуктивність.

2.2.3. *Аналіз СКБД PostgreSQL*

PostgreSQL – це об'єктно-реляційна система управління базами даних, є найбільш передовою системою баз даних з відкритим кодом. Розроблена для роботи на платформах UNIX, тим не менш, може бути портативною, так щоб мати змогу працювати на різних платформах, таких як Mac OS X, Solaris і Windows. PostgreSQL є безкоштовним і відкритим програмним забезпеченням, вихідний код доступний під ліцензією PostgreSQL, ліберальною ліцензією з відкритим кодом. Ця СКБД вважається дуже

стабільною системою, тому підтримка вимагає мінімальних зусиль. Розробка програмного забезпечення на основі PostgreSQL має низьку загальну вартість в порівнянні з іншими системами управління базами даних.

PostgreSQL – це універсальна, вона дозволяє додавати власні функції, розроблені з використанням різних мов програмування, таких як C/C++, Java тощо. PostgreSQL розроблений для розширення, можна визначити власні типи даних, типи індексів, тощо [15].

Багато компаній побудували продукти та рішення, використовуючи PostgreSQL. Деякі з цих компанії - Apple, Fujitsu, Red Hat, Cisco, Juniper Network тощо.

Переваги PostgreSQL:

– висока продуктивність – паралельні запити, механізми блокування, індекси (btree, hash, etc.), оптимізовані запити, розподілені таблиці;
– реплікації – синхронна і асинхронна реплікація, логічна і фізична реплікація, часткова реплікація, Point-in-Time-Recovery;
– безпека – підтримка користувачів та їх ролей, підтримка SSL;
– типи даних – окрім стандартних для більшості SQL СКБД типів, містить багато інших. Є можливість створювати власні типи;
– сумісність з різними платформами, що використовують всі основні мови програмування;
– гнучкий повнотекстовий пошук;
– open-source та community-driven – повністю відкритий проект, вихідний код PostgreSQL розроблений великою спільнотою, яка в документації та форумах описує, як працювати з СКБД.

Недоліки PostgreSQL:

– продуктивність в межах пам'яті – для кожного нового підключення клієнта PostgreSQL створює новий процес, якому виділяється 10 МБ пам’яті;

– популярність – PostgreSQL історично відставала від MySQL за популярністю. Як наслідок менше інструментів сторонніх виробників та не так багато адміністраторів.

2.2.4. *Результати аналізу СКБД*

На сьогоднішній день MySQL і PostgreSQL є двома найпопулярнішими реляційними системами керування базами даних з відкритим кодом у світі. Кожна з них має свої унікальні особливості та обмеження. Зважаючи на вище описані переваги та недоліки обох СКБД було прийнято рішення для розробки даного дипломного проекту використовувати СКБД PostgreSQL.

2.3. Аналіз та порівняння інструментів, для розроблення серверної частини web-додатків

2.3.1. *Аналіз мови програмування Python*

Python є інтерпретованою, об'єктно-орієнтованою мовою програмування високого рівня. Мова була створена на початку 1990-х роками Гвідо ван Россумом.

Простий, легкий у вивченні синтаксис Python підкреслює читабельність і, отже, знижує вартість технічного обслуговування. Мова підтримує модулі та пакети, що заохочує модульність програми та повторне використання коду. Інтерпретатор Python і велика стандартна бібліотека доступні в початковій або двійковій формі для всіх основних платформ і можуть вільно розповсюджуватися.

Python є прикладом FLOSS. Простіше кажучи, ви можете вільно поширювати копії цього програмного забезпечення, читати його вихідний код, вносити зміни до нього та використовувати його у нових програмах [16].

Завдяки відкритому вихідному коду – Python був перенесений на багато інших платформ. Розроблені на цій мові програми можуть працювати на будь-якій платформі без будь-яких змін взагалі, якщо бути досить обережним, щоб уникнути будь-яких системних функцій.

Інтерпретованість. Python не потребує компіляції в двійковій системі. Необхідно просто запустити програму безпосередньо з вихідного коду. Внутрішньо Python перетворює вихідний код у проміжну форму, яка називається bytecode, а потім переводить її на рідну мову комп'ютера, потім запускає її. Не потрібно турбуватися про компіляцію програми, переконавшись, що відповідні бібліотеки пов'язані і завантажені тощо. Це також робить Python програми набагато більш портативними.

Python підтримує процедурно-орієнтоване програмування, а також об'єктно-орієнтоване програмування. У процедурно-орієнтованих мовах програма будується навколо процедур або функцій. У об'єктно-орієнтованих мовах програма будується навколо об'єктів, які поєднують дані і функціональність. Python має дуже потужний, але спрощений спосіб виконання ООП, особливо в порівнянні з такими мовами, як C ++ або Java.

Стандартна бібліотека Python величезна. Вона може допомогти у виконанні різних речей, пов'язаних з регулярними виразами, генерацією документації, модульним тестуванням, потоками, базами даних, веб-браузерами, JSON, XML та іншими залежними від системи матеріалами.

2.3.2. *Аналіз мови програмування JavaScript*

JavaScript (часто просто JS) – це інтерпретована або JIT-компільована, об'єктно-орієнтована мова з функціями першого класу. Найбільш широке застосування знаходить як мова сценаріїв веб-сторінок, але також використовується і в інших програмних продуктах, наприклад, Node.js або Apache CouchDB. JavaScript це мова з динамічною типізацією, яка підтримує об'єктно-орієнтований, імперативний і декларативний стилі

програмування. JavaScript запускається на стороні клієнта. Стандартом мови JavaScript є ECMAScript.

JavaScript може функціонувати як процедурна, так і об'єктно-орієнтована мова. Об'єкти створюються програмно шляхом приєднання методів і властивостей до порожніх об'єктів під час виконання. Як тільки об'єкт був побудований, його можна використовувати як прототип для створення подібних об'єктів [17].

Синтаксис та функції мови не завжди підходять кожному користувачу, все залежить від проекту та вимог до нього. Так з’явилось декілька нових мов які перетворюються в JavaScript перед тим як виконатись браузером. Подібне перетворення коду у зрозумілий браузеру вигляд називають транспіляцією. Сучасні інструменти транспіляції виконують свою роботу досить швидко, одним із таких інструментів є Babel.

2.4. Аналіз та порівняння інструментів для розроблення клієнтської частини web-додатків

2.4.1. *Аналіз AngularJS*

AngularJS – це відкритий JavaScript фреймворк для клієнтської частини програмного застосунку, що підтримується Google та спільнотою розробників, який допомагає спростити розробку та підтримку клієнтської веб-додатків. Angular цілком та повністю написаний на JavaScript, тому фреймворк підтримується і працює на всьому, що сприймає цю мову програмування [19].

В першу чергу AngularJS спрямований на розробку SPA і підтримує схему проектування MVC. Фреймворк використовується для розробки великої кількості сайтів, включаючи Google і Virgin America.

Виділяють наступні ключові особливості фреймворку [19]:

– зв'язування даних – зменшує кількість циклів оновлення та скорочує кількість коду до менш ніж 80%;

– директиви – окремі елементи, які можна повторно використовувати для маніпуляцій DOM;
– AngularJS не використовує жодного механізму рендерингу сторінок;
– сервіси – AngularJS надається з кількома вбудованими сервісами, наприклад такими як $http, щоб мати змогу робити XMLHttpRequests;
– Model-View-Whatever – MVW це шаблон дизайну для поділу програми на різні частини.

Переваги AngularJS:

– надає можливість розроблювати односторінкові додатки (SPA) чистим та доступним для обслуговування шляхом;
– можливість прив'язки даних до HTML;
– код на AngularJS піддається Unit тестуванню;
– можливість використовувати багато разів один і той самий компонент;
– в AngularJS всі представлення є чистими HTML сторінками, а вся бізнес логіка – це контролери написані на JavaScript.

Недоліки AngularJS:

– Angular великий і складний. Існує багато способів зробити одне завдання, тому важко сказати який з шляхів буде найкращий;
– велика кількість “спостерігачів” (watchers) може спричинити уповільнення роботи застосунку;
– рішення на AngularJS з ростом проекту слабо піддаються до масштабування.

2.4.2. *Аналіз Vue.js*

Vue є прогресивним фреймворком для побудови інтерфейсів користувача. На відміну від інших – Vue був розроблений з нуля, щоб бути

максимально адаптивним. Ядро орієнтоване лише на шар представлення і легко інтегрується з іншими бібліотеками або існуючими проектами. З іншого боку, Vue також цілком здатний до SPA, використовуючи їх у поєднанні з сучасними інструментами та бібліотеками [19].

Творцем Vue.js є Evan You, колишній співробітник Google і Meteor Dev Group. Vue широко використовується серед таких компаній як Alibaba, Baidu, Xiaomi, Sina Weibo і ін. Також фреймворк входить в ядро Laravel і PageKit, система управління репозиторіями GitLab теж перейшла на Vue.js. Vue розроблений з акцентом на продуктивність і використовує віртуальний DOM, підтримується серверний рендеринг, можливість використовувати JSX і т.д. Зараз в основному фреймворк підтримується тільки співтовариством. Vue.js має досить невеликий розмір - не більше 20 кБ, і при цьому володіє хорошою продуктивністю в порівнянні з такими фреймворками як Angular або React.

Одним з ключових моментів в роботі Vue.js є віртуальний DOM. Структура веб-сторінки, як правило, описується за допомогою DOM. Для взаємодії з DOM застосовується JavaScript. При спробі маніпулювати html-елементами за допомогою JavaScript можна зіткнутися зі зниженням продуктивності. А операції над елементами можуть зайняти деякий час, що позначиться на досвіді користувача при роботі з сайтом. Однак якби ми працювали з коду js з об'єктами JavaScript, то операції проводилися б швидше. Для цього Vue.js використовує віртуальний DOM, який є легшою копією звичайного DOM. Завдяки віртуальному DOM підвищується продуктивність програми.

Vue.js підтримується всіма браузерами, які сумісні з ECMAScript 5. На даний момент це все сучасні браузери, в тому числі IE11.

Переваги Vue.js:

– легка інтеграція – дуже просто додати Vue.js до існуючого веб-проекту;

– дуже малий розмір – розмір фреймворку становить 18-21 Кб;

– детальна документація – Vue.js має вичерпну документацію;
– гнучкість – Vue.js дозволяє користувачеві писати свої компоненти у файлі HTML або файлі JavaScript;
– двостороння комунікація – завдяки архітектурі MVVM Vue.js полегшує двосторонній зв'язок.

Недоліки Vue.js

– брак ресурсів інформації – Vue.js тільки розвивається, отже інформації про ті чи інші можливості фреймворку може бути менше ніж у його конкурентів React та Angular;
– китайське коріння – велика частина товариства не англомовні люди, це часто впливає на появу сервісів з великим відсотком китайської мови. Також це впливає і на документацію деяких сервісів.

3. СТРУКТУРНО-АЛГОРИТМІЧНА ОРГАНІЗАЦІЯ

3.1. Структура програмного забезпечення

Загальна структура розробленого сервісу зображена на рис. 3.1.

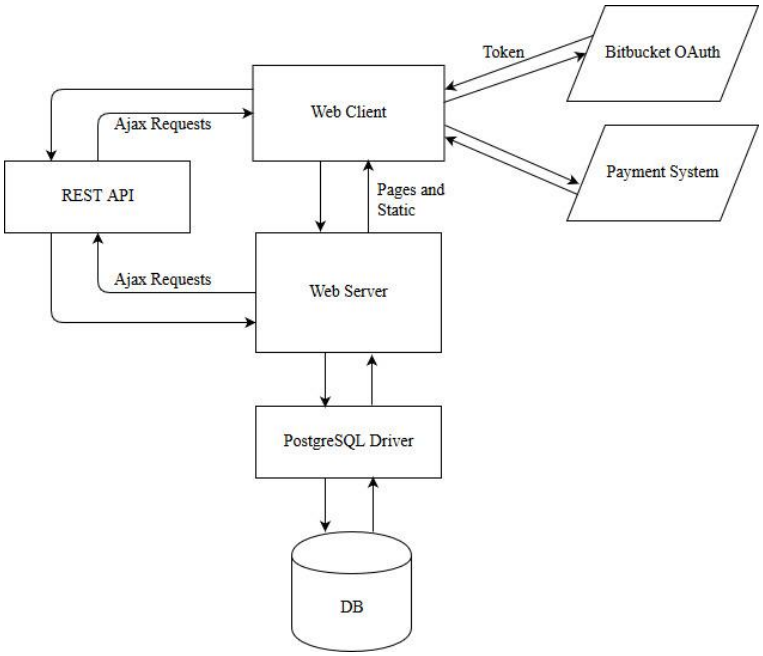


Рис. 3.1. Структурна схема сервісу

Розроблена система реалізована у вигляді веб-додатку та складається з веб-серверу, клієнтської частини та бази даних, які можна назвати основними компонентами цієї системи. Крім вказаних вище систем також задіяні сторонні сервіси, наприклад Bitbucket та Payment System.

PostgreSQL Driver та DB – база даних проекту, що отримує запити з веб-серверу, виконує вказані операції та повертає потрібні дані, якщо це передбачає надіслана раніше інструкція.

Web Server – веб-сервер розроблюваного сервісу. Підтримує комунікацію та об’єднує клієнт і базу даних через REST API. Сервер складається с модулів, що відповідають за представлення моделей з бази даних, за серіалізацію моделей з бази даних у JSON строки, та з модулю що об’єднує два останніх аби відповідати на потрібні запити. Також сервер

відповідає за відправку сторінок та статичних файлів для веб-клієнту сервісу.

REST API – вказані правила, яких повинен дотримуватись клієнт, аби отримувати бажані відповіді від серверу.

Web client – веб-клієнт програмного застосунку, відповідає за відображення інформації ресурсу, яку отримує з сервера за допомогою асинхронних запитів через API. Також відповідає за авторизацію користувачів на сервісі за допомогою стороннього сервісу Bitbucket. Та за комунікацію з якоюсь платіжною системою.

Bitbucket – рішення для управління Git репозиторіями. В нашому випадку використовується для авторизації на сервісі. Для того щоб користувач мав змогу на якісь дії на сервісі він обов’язково має бути авторизованим через Bitbucket. З сервісу ми отримуємо токени для користувача, який на даний момент користується розробленим програмним забезпечення та допомагає отримувати додаткову інформацію про користувача.

Payment System – платіжна система, яку потенціально може використовувати сервіс для проведення грошових операцій користувачами, які хочуть підтримати той чи інший проект. В даній реалізації платіжну систему буде імітувати одна сторінка да додаткова таблиця в базі даних проекту. Сторонню систему вирішено використовувати через їх спеціалізованість та підвищений захист для проведення подібних операцій.

3.2. Структура та опис бази даних

Враховуючи всі умови та вимоги до розроблюваного сервісу складено наступну схему бази даних системи (рис. 3.2).

Таблиця Projects – таблиця, яка описує проект на сервісі, складається з наступних полів:

– title – назва проекту;

– status – статус проекту;

– create_date – дата створення проекту;

– description – опис проекту;

– logo – логотип проекту;

– repository_url – посилання на репозиторій проекту.

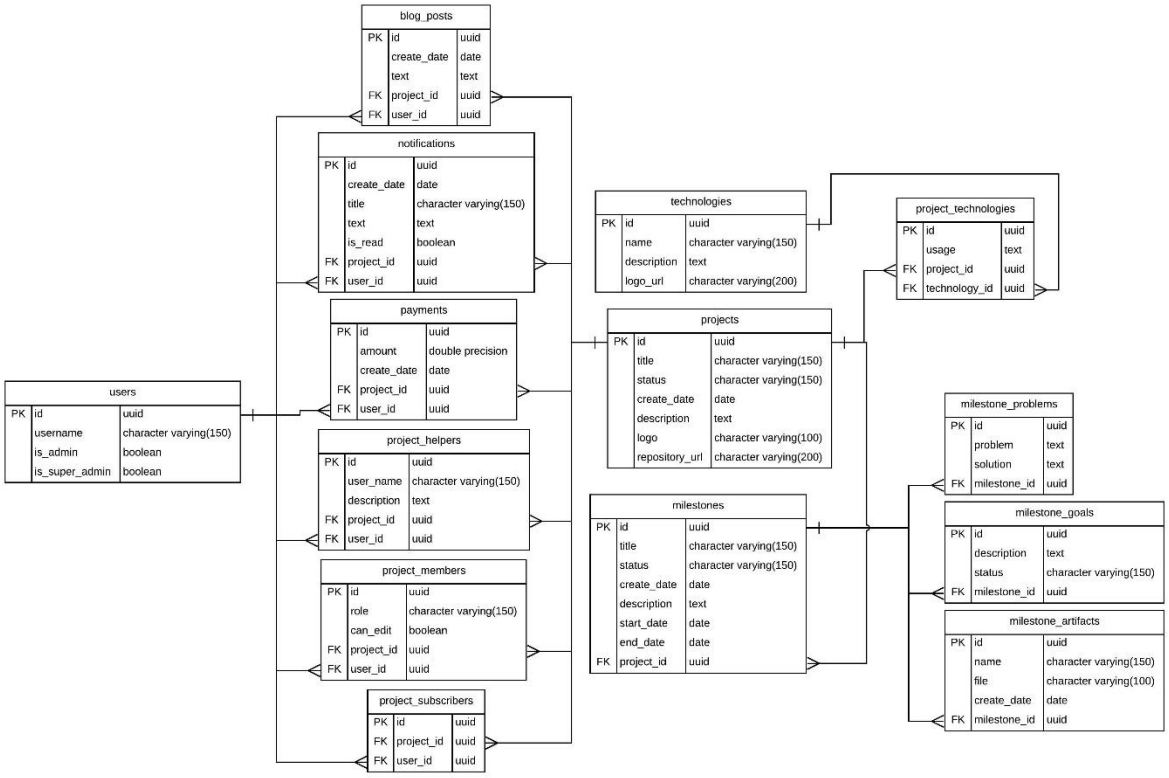


Рис. 3.2. Схема бази даних сервісу

Крім того з проектом зв’язані і інші сутності, наприклад Milestones, BlogPosts, ProjectMembers, ProjectTechnologies, ProjectHelpers, ProjectSubscribers, Notifications та Payments.

Таблиця Milestones – таблиця, яка описує контрольну точку будь-якого проекту. Зв’язок таблиці Projects з таблицею Milestones можна позначити як One-to-many, оскільки один проект може складатись з багатьох контрольних точок, а контрольна точка в свою чергу може належати тільки одному проекту.

Таблиця Milestones складається з наступних полів:

– title – поле, що показує назву контрольної точки;

– status – статус контрольної точки;

– create_date – дата створення контрольної точки;

– description – опис контрольної точки;

– start_date – дата старту контрольної точки;

– end_date – дата закінчення контрольної точки;

– project_id – унікальний ідентифікатор проекту до якого належить об’єкт з таблиці Milestones.

Як і таблиця Projects, Milestones теж має залежні сутності які його доповнюють і тим самим доповнюють сутність об’єкту з таблиці Projects. Такими сутностями на схемі є MilestoneGoals, MilestoneProblems та MilestoneArtifacts.

Таблиця MilestoneGoals – таблиця, що описує цілі контрольної точки. Відношення Milestones до MilestonesGoals можна встановити як One-to-many, оскільки одна контрольна точка може мати декілька цілей, а одна конкретна ціль може належати лише одній конкретній контрольній точці.

Таблиця складається з наступних полів:

– milestone_id – ідентифікатор контрольної точки, якій належить дана ціль;

– description – опис цілі;

– status – статус цілі.

Таблиця MilestoneProblems – таблиця, що описує проблеми та рішення, що виникли під час виконання контрольної точки. Відношення таблиці Milestones до таблиці MilestoneProblems є One-to-many.

Таблиця MilestoneProblems складається з наступних полів:

– milestone_id – унікальний ідентифікатор контрольної точки, яку описує об’єкт даної таблиці;

– problem – опис проблеми;

– solution – опис рішення проблеми.

Таблиця MilestoneArtefacts – таблиця, що описує артефакти контрольної точки. Таблиця MilestoneArtefacts відноситься до таблиці Milestones як One-to-many, оскільки сутність таблиці Milestones може мати декілька артефактів, а конкретний артефакт може відноситись лише до одної контрольної точки. Таблиця складається з наступних полів:

– milestone_id – ідентифікатор контрольної точки, до якої належить артефакт;
– name – ім’я артефакту;
– file – посилання на файл-артефакт в межах серверу;
– create_date – дата створення артефакту.

Таблиця Users – таблиця з інформацією про користувачів, що авторизувались на сервісі. Оскільки авторизація на сервісі проводиться через сторонній сервіс Bitbucket, то в таблиці зберігається мінімальна інформацію про користувача, а всі потрібні дані отримуються з його акаунту на Bitbucket.

Таблиця Users складається з наступних полів:

– username – ім’я користувача взяте з даних користувача на сервісі Bitbucket;
– is_admin – поле, що показує чи є користувач адміністратором сервісу;
– is_super_admin – поле, що показує чи є користувач супер адміністратором сервісу.

Таблиця BlogPosts – таблиця, що описує запис в блозі проекту. Відношення між таблицями Projects та BlogPosts встановлюється як One-to-many, оскільки у одного проекту може бути безліч записів у блозі, а один запис блогу може відноситись лише до одного проекту. Також варто зазначити, що таблиця пов’язана і з таблицею Users, оскільки необхідно мати посилання на автора, який залишив запис в блозі. Зв’язок між таблицями Users та BlogPosts встановлюється як One-to-many, оскільки у користувача може бути безліч записів у багатьох блогах, а у запису може

бути лише один автор. Таблиця BlogPosts також відноситься сама до себе у зв’язку One-to-many, оскільки запис може бути коментарем до іншого запису і батьків.

Таблиця BlogPosts складається з наступних полів:

– project_id – проект до якого відноситься даний запис блогу;

– user_username – унікальне ім’я користувача, який залишив запис;

– text – текстовий вміст запису;

– create_date – час створення запису.

Таблиця ProjectMembers – асоціативна таблиця у Many-to-many зв’язку між таблицею Projects та таблицею Users. Подібний зв’язок пояснюється тим що користувачі можуть мати багато проектів, а над проектом може працювати декілька користувачів. Також дана таблиця допомагає визначити роль того чи іншого користувача на проекті та його права на редагування.

Таблиця ProjectMembers містить наступні поля:

– user_username – унікальне ім’я користувача;

– project_id – унікальний ідентифікатор проекту, в якому бере участь користувач;

– role – роль користувача на проекті;

– can_edit – поле яке визначає чи може користувач редагувати інформацію, яка стосується того чи іншого проекту.

Таблиця Technologies – таблиця що містить всі зареєстровані на ресурсі технології за допомогою яких розроблювались ті чи інші проекти. Таблиця містить наступні поля:

– name – назва технології;

– description – опис технології;

– logo_url – посилання на офіційний логотип технології.

Таблиця ProjectTechnologies – асоціативна таблиця у зв’язку Many-to-many між таблицями Projects та Technologies. Такий зв’язок встановлений оскільки у проекті може застосовуватись безліч технологій, а одна

технологія може використовуватись у багатьох проектах. Таблиця показує не тільки взаємозв'язок між технологіями і проектами, а і можливий спосіб використання технології на тому чи іншому проекті. Таблиця складається з наступних полів:

– project_id – унікальний ідентифікатор проекту;

– technology_id – унікальний ідентифікатор технології на сервісі;

– usage – опис використання технології на проекті.

Таблиця ProjectSubscribers – ще одна асоціативна таблицями між таблицями Projects та Users в Many-to-many зв'язку. Показує який користувач на які проекти підписаний. Many-to-many зв'язок пояснюється тим, що один користувач може бути підписаний на декілька проектів, а оди проект може мати безліч підписників. Таблиця складається всього з двох полів:

– project_id – унікальний ідентифікатор проекту;

– user_username – унікальний ідентифікатор користувача.

Notifications – таблиця що представляє повідомлення в системі. Повідомлення використовуються, коли необхідно підписників проекту повідомити про якісь події на проекті (новий запис у блозі, зміна статусу контрольної точки, тощо). Складається з таких полів:

– user_username – унікальний ідентифікатор користувача якому призначено повідомлення;

– project_id – проект, зміни в якому привели до відправлення повідомлення користувачу;

– title – заголовок повідомлення;

– text – текст повідомлення;

– create_date – дата відправки повідомлення;

– is_read – статус, що дозволяє визначити, чи є повідомлення прочитане користувачем.

Таблиця ProjectHelpers – таблиця, що показує людей, які тим чи іншим способом допомогли проекту. В якомусь сенсі може вважатись ще однією

асоціативною таблицею між таблицями Projects та Users. Складається з наступних полів:

– user_username – унікальний ідентифікатор користувача, який допоміг проекту;
– username – ім’я людини, яка допомогла проекту (поле використовується, якщо користувач не зареєстрований на сервісі);
– project_id – унікальний ідентифікатор проекту, якому допоміг той чи інший користувач;
– description – опис того як саме користувач допоміг проекту.

Таблиця Payments – відображає те які користувачі підтримали той чи інший проект матеріально.

Складається з наступних полів:

– user_id – унікальний ідентифікатор користувача;
– project_id – унікальний ідентифікатор проекту;
– amount – сума пожертви.

3.3. Структура та опис REST модулю

Для комунікації клієнтської та серверної частини програмного застосунку було розроблено модуль з REST API.

Доступ до API ресурсів можна отримати виконуючи запити, які співпадають з наступним шаблоном: {domain}/api/1.0/{route}, де {domain} – доменне ім’я сервісу, наприклад www.example.com, а {route} – один з наперед зазначених шляхів доступу до ресурсів. Далі наведено різновиди можливих варіацій {route} частини запиту.

Першими для розгляду запитами будуть ті, які приймають лише методи GET та POST. При GET запитах на дані адреси будуть повертатись колекції даних певного типу. Тип можна визначити за складовою адреси запиту. Наприклад якщо зробити запит на адресу, де {route} є projects/, то у відповідь ми отримаємо набір даних типу Projects, а якщо зробити запит

з {route}, що являє собою строку projects/<project_pk>/technologies, то у відповідь отримаємо набір даних типу Technology, які зв’язані з проектом з ключем project_pk. При POST запитах в базі даних сервісу будуть створені нові об’єкти відповідного типу, а у відповідь на запит повернеться створений об’єкт.

До описаних адрес можна віднести наступні:

– projects/

– projects/<project_pk>/technologies/

– projects/<project_pk>/members/

– projects/<project_pk>/helpers/

– projects/<project_pk>/posts/

– projects/<project_pk>/payments/

– projects/<project_pk>/milestones/

– projects/<project_pk>/milestones/<milestones_pk>/problems/

– projects/<project_pk>/milestones/<milestones_pk>/goals/

– projects/<project_pk>/milestones/<milestones_pk>/artifacts/

– users/

– technologies/

Наступний набір адрес підтримує запити з методами GET, PUT та DELETE. При GET запитах на дані адреси у відповідь буде повертатись об’єкт вказаного типу ключ якого вказано в запиті. Наприклад на запит за адресою projects/<pk>/ у відповідь отримаємо об’єкт типу Projects з унікальним ключем pk. При запитах з методом PUT – об’єкт вказаного типу і з вказаним в запиті ключем буде оновлено згідно з тими даними, які надсилаються в тілі запиту. У відповідь на запит прийде оновлений об’єкт. При DELETE запитах на дані адреси – об’єкти за вказаними ключами в запитах будуть видалені з бази даних сервісу.

До описаних адрес належать наступні:

– projects/<pk>/

– projects/<project_pk>/technologies/<pk>/

– projects/<project_pk>/members/<pk>/

– projects/<project_pk>/helpers/<pk>/

– projects/<project_pk>/posts/<pk>/

– projects/<project_pk>/payments/<pk>/

– projects/<project_pk>/milestones/<pk>/

– projects/<project_pk>/milestones/<milestones_pk>/problems/<pk>/

– projects/<project_pk>/milestones/<milestones_pk>/goals/<pk>/

– projects/<project_pk>/milestones/<milestones_pk>/artifacts/<pk>/

– users/<pk>/

– technologies/<pk>/

3.4. Структура та опис Vue.js компонентів

Для реалізації клієнтської частини сервісу було обрано фреймворк Vue.js . Для більш зрозумілого та детального опису всіх створених під час розробки компонентів було розроблено наступну схему (див. Додаток 1), що показує створені компоненти та їх взаємодію один з одним.

App.vue – основний компонент, який інтегрується в html сторінку, в ньому відбувається основне відтворення всіх потрібних компонентів.

Layout.vue – компонент, який об’єднує наступні 4 компоненти: TagsView.vue, Sidebar.vue, AppMain.vue, Settings.vue та Navbar.vue.

Navbar.vue – компонент, що відповідає за відтворення та логіку хедеру сайту, що вміщає в себе невелике меню користувача, контролер зміни шрифту, контролер відображення сайту (повноекранний чи ні).

TagsView.vue – компонент, що відповідає за логіку відображення історії відвіданих сторінок з можливістю повернення на будь-яку з них.

Sidebar.vue – компонент, що відповідає за роботу, логіку та відтворення навігаційного меню сайту та його Sidebar.vue елементів, які містять інформацію про сторінку.

Settings.vue – компонент, що відповідає за роботу, логіку та зображення панелі з деякими загальними налаштуваннями сервісу.

AppMain.vue – основний компонент, що відповідає за зображення потрібних користувачу компонентів та за їх зміну один одним в залежності від адресної строки в браузері та потреб користувача.

HomePage.vue – компонент, що представляє собою домашню сторінку сервісу. Поки не виконує особливої логіки, окрім тієї, що відображає інші компоненти з інформацією про проекти користувача, найпопулярніші проекти чи нові проекти на сервісі.

UsersPage.vue – компонент, що відповідає за відтворення сторінки з користувачами сервісу. В даній реалізації сервісу ця сторінка доступна лише адміністратору сервісу.

TechnologiesPage.vue – компонент, що є відповідальним за сторінку з технологіми, які були зареєстровані на сервісі.

ProjectsPage.vue – компонент, що відповідає за показ сторінки з проектами сервісу.

ProfilePage.vue – компоненту, що відповідає за відтворення особистої сторінки користувача на якій відображається інформація про акаунт, яка стосується сервісу.

ProjectsPreviewCollection.vue – компонент, що відповідає за логіку запиту на сервер за даними про проекти для їх подальшого скороченого зображення та опису. В залежності від фільтру запиту будуть відображатись різні проекти.

ProjectPreview.vue – компонент, що відповідає за зображення короткого опису проектів, які приходять у відповідь на запит в компоненті ProjectsPreviewCollection.vue.

Project.vue – компонент, що показує інформацію по проекту, робить запит на сервер, щоб отримати подробиці про той чи інший проект. Компонує решту компонентів, які допомагають відобразити ту чи іншу інформацію, яка стосується проекту.

Helpers.vue – компонент, що відображає інформацію по користувачам, які тим чи іншим способом допомогли проекту. Також відображається інформація про те яку допомогу надав той чи інший користувач.

Blog.vue – компонент, що отримує список записів у блозі проекту, які залишають користувачі чи розробники проекту.

BlogPost.vue – компонент для показу інформації про запис у блозі, наприклад основний текст запису, ім’я користувача який залишив запис, дата створення запису.

Milestones.vue – компонент, що запитує дані з сервера про контрольні точки проекту та зображує ці точки проекту за допомогою компоненту Milestone.vue.

Milestone.vue – компонент для зображення інформації про контрольну точку проекта та для компонування інших компонентів, що допомагають відображати додаткову інформацію.

Problems.vue – компонент, що запитує сервер та показує на сторінці проблеми контрольної точки та їх рішення.

Goals.vue – компонент для показу цілей контрольної точки та їх поточних статусів.

Goal.vue – компонент для зображення об’єкту цілі проекту разом з її теперішнім статусом.

Artifacts.vue – компонент для отримання артефактів з серверу, можливістю їх завантаження та зображення.

Artifact.vue – компонент для показу завантажених контрольній точці артефактів.

Technologies.vue – компонент, що відповідає за запит на сервер за технологіями, що були зареєстровані на ресурсі, для подальшого їх зображення за допомогою компонента Technology.vue.

Technology.vue – компонент, що відповідає за зображення технологій, які в залежності від фільтру, надійшли до компоненту Technologies.vue. На

сторінці проекту до інформації про технологію також додається опис варіантів використання технології на проекті.

Users.vue – компонент, що відповідає за запит на сервер щодо користувачів, які хоч раз авторизувались на сервері. Також компонент використовується при запиті на учасників певного проекту.

User.vue – компонент для показу короткої інформацію про користувача з можливістю надання прав адміністратора, або прав на редагування проекту, якщо користувач відображається на сторінці проекту.

4. АНАЛІЗ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Дизайн і структура сторінок

Однією з основних частин розробленого програмного забезпечення є клієнтська частина. Кожна сторінка на сервісі підпадає під наступний шаблон, що показаний на рис. 4.1.

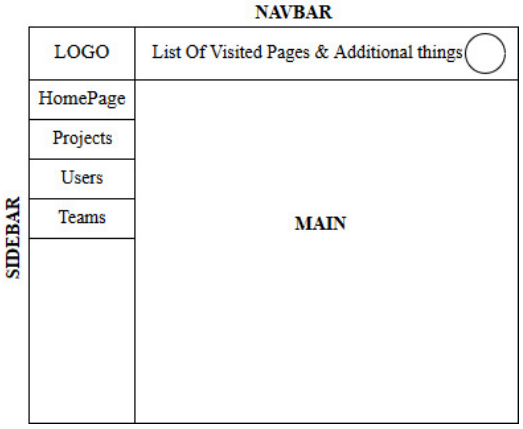


Рис. 4.1. Схема структури сторінок сервісу

Вся клієнтська частина розроблялась за допомогою інструментів HTML, SASS та фреймворку Vue.js, також для зручності і швидкості розробки використовувалась готова бібліотека компонентів для Vue під назвою Element UI.

Для того щоб потрапити на основну частину порталу необхідно пройти авторизацію. Дизайн сторінки авторизації зображено на рис. 4.2.

Форма завжди знаходиться в центрі сторінки авторизації. На цій формі зображено напис та три кнопки. Перша кнопка тематично стилізована під сервіс Bitbucket і дозволяє авторизуватись використовуючи акаунт на цьому сервісі. Друга кнопка дозволяє зайти на сервіс зі статусом Гість, при цьому буде обмежено деякий функціонал для цього користувача. Третя кнопка відкриває випадаючий список, в якому можна обрати мову перекладу сайту.

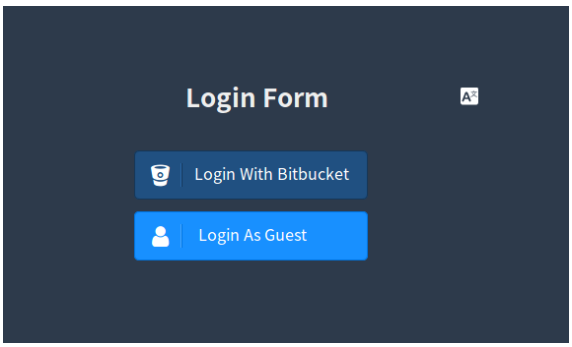


Рис. 4.2. Форма авторизації сервісу

В якості основного інструменту для навігації було розроблено меню навігації, яке зображено на рис. 4.4. В меню на даний момент присутні 4 вкладки: Home, Projects, Technologies та Users. Варто зазначити, що вкладка Users зображується лише якщо користувач, що користується сервісом в даний момент, є адміністратором. Також над меню зображено назву сервісу разом з можливим логотипом. Вкладка Home переводить поточного користувача но домашню сторінку сервісу, де відображаються найпопулярніші проекти, нові проекти та проекти поточного користувача, якщо такі є. Вкладка Projects перенаправляє користувача на сторінку з проектами, які зареєстровані на сервісі. Вкладка Technologies перенаправляє користувача на сторінку з технологіями зареєстрованими на сервісі. Вкладка Users – вкладка що перенаправляє користувача на сторінку з авторизованими на сервісі користувачами. Активна сторінка, тобто сторінка на якій зараз знаходиться користувач, в меню буде підсвічуватись синім кольором. Панель меню по ширині займає близько 13% всієї сторінки, а по висоті займає весь простір.

Також користувачу завжди відображається хедер сайту (рис. 4.5), що завжди заходиться згори сторінки. По ширині розтягується по всій сторінці, починаючи від меню і до правої сторони сторінки. По висоті блок займає приблизно 6% сторінки.

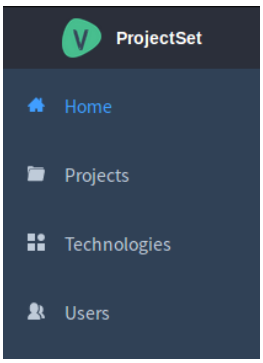


Рис. 4.4. Бокове меню сервісу

Блок поділяється на 2 частини: верхню та нижню. Верхня частина хедеру відображає куточок користувача, функціональні кнопки, шлях до поточної сторінки від домашньої сторінки та кнопка, що дозволяє згорнути меню у більш компактне. Вказані функціональні кнопки дозволяють змінити мову, збільшити шрифт чи перевести сайт в повноекранний режим. Куточок користувача, на якому зображено аватар поточного користувача, являє собою звичайний випадаючий список з посиланням на сторінку та функцією Log out.

Рис. 4.5. Хедер сервісу

Нижня частина хедеру простий блок на якому відображаються відвідані користувачем сторінки сервісу у вигляді невеличких блоків які в ширину підлаштовуються під текст. Також є додаткові кнопки для того щоб прибрати блок с панелі. Блок поточної сторінки виділяється зеленим кольором.

На деяких сторінках сервісу відображаються проекти у скороченому варіанті (рис. 4.6), лише логотип, назва, технології та опис. Блок складається з 4 частин. Перша частина займає 25% всього блоку з лівої сторони і там відображається логотип проекту. Другий блок можна назвати контейнером

для назви проекту та відображення кількості підписників. Назва проекту виділяється жирним шрифтом, а кількість підписників виділяється іншим кольором. Третій блок являє собою контейнер під використовувані проектом технології. Кожна технологія представляє собою блакитний блок який по розмірам підлаштовується до назви технології. Четвертий блок займає собою решту простору і вміщує обмежену кількість символів, являє собою контейнер для короткого опису того чи іншого проекту.

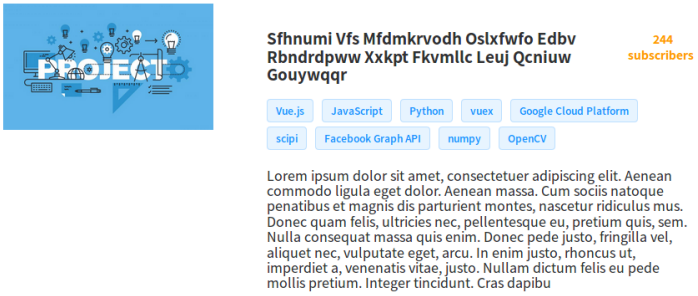


Рис. 4.6. Представлення проекту на сторінці проектів

Однією з основних частин сторінки саме проекту є частина з описом будь-якої контрольної точки цього проекту. Частина дизайну блоку з контрольною точкою зображено на рис. 4.7 . Блок з контрольними точками представляє собою часову лінію з точками. Кожна точка – дата початку тої чи іншої контрольної точки разом з карткою контрольної точки на якій описується все що її стосується. Колір точки на часовій лінії визначається статусом її контрольної точки. Якщо статус “не розпочато” – колір сірий, статус “в процесі” – колір синій, статус “завершено” – зелений, статус “закрито” – червоний. Всі картки на часовій лінії відсортовані по даті старту контрольної точки, яку описує дана картка

Блок контрольної точки складається з декількох частин. Перша частина показує загальну інформацію. В лівому куті жирним шрифтом наведена назва контрольної точки.. В правому куті відображається статус контрольної точки в кольоровому блоці, колір залежить від статусу.

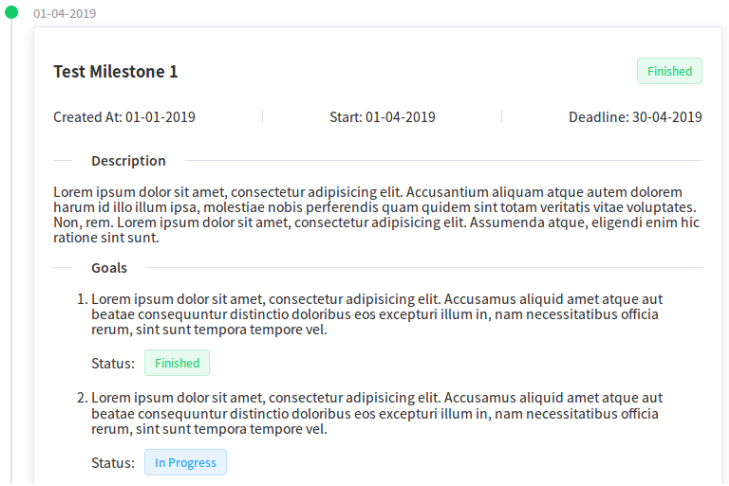


Рис. 4.7. Представлення контрольної точки на сторінці проекту

Нижче розташовані ключові дати контрольної точки, а саме дата її створення, дата початку контрольної точки та дата дедлайну. Далі у відповідній секції відображається короткий опис контрольної точки. В наступній секції у вигляді списку користувачу наводяться цілі даної контрольної точки разом з статусами. Логіка та дизайн статусів аналогічні тим, що були описані вище при описі інформації про контрольну точку. Наступним блоком є опис проблем та рішень у вигляді таблиці з двома стовпцями з відповідними назвами. І останній блок відповідає за артефакти з відповідним віконечком для завантаження та підблоком для перегляду та редагування. Кожна описана частина картки контрольної точки відокремлена від іншої горизонтальною лінією з написом, який дає зрозуміти який блок зараз переглядає користувач.

4.2. Тестування системи

Після завершення розробки програмного забезпечення було проведено тестування на відповідність поставленим у технічному завданні вимогам. Було проведено димне тестування та було застосовано тестування критичного шляху. При даному виді тестування перевіряється 20% функціоналу системи, якими точно будуть користуватись 80% користувачів.

Для тестування було складено наступні сценарії перевірки:

1. Авторизуватись в системі через Bitbucket.
2. В ролі адміністратора створити проект на сервісі.
3. В ролі адміністратора додати на проект куратора проекту.
4. В ролі куратора проекту відредагувати інформацію про проект.
5. В ролі куратору проекту додати до команди розробника.
6. В ролі куратора проекту створити проекту контрольну точку.
7. В ролі розробника без прав редагування спробувати змінити інформацію про проект.
8. В ролі розробника з правами редагування змінити інформацію про проект.

В результаті тестування за вказаними сценаріями було отримано наступні результати.

Таблиця 4.1

Результати тестування

Номер сценарію	Опис	Результат
1	2	3
1	1. Користувач на сторінці авторизації натискає “Auth with Bitbucket”. 2. Авторизується на сторінці Bitbucket	Після вводу правильних авторизаційних даних користувача з Bitbucket перенаправляє на головну сторінку.
2	1. Користувач авторизується як адміністратор. 2. Переходить на сторінку з проектами. 3. Натискає кнопку “Add Project”. 4. Заповнює необхідні поля.	Після збереження даних з форми на сторінці проектів з’являється новий проект.

Продовження табл. 4.1

1	2	3
3	1. Користувач авторизується на сервісі в ролі адміністратора. 2. Переходить на сторінку проекту. 3. Переходить на вкладку команди. 4. Натискає кнопку, що дозволяє назначити куратора. 5. Користувач обирає користувача на роль куратора проекту.	На сторінці проекту на вкладці команди відображається новообраний куратор.
4	1. Користувач авторизується на акаунт куратора. 2. Переходить на сторінку проекту. 3. Натискає кнопку редагування. 4. Користувач редагує інформацію на відповідній формі. 5. Користувач відправляє форму.	Після закриття форми на сторінці проекту можна бачити оновлену інформацію.
5	1. Користувач авторизується на акаунт куратора. 2. Переходить на сторінку проекту. 3. Переходить на вкладку команди. 4. Користувач натискає кнопку “Add Member”. 5. Користувач обирає потрібного розробника.	На вкладці команди проекту відображається запрошений учасник.

Продовження табл. 4.1

1	2	3
6	1. Користувач авторизується на акаунт куратора. 2. Переходить на сторінку проекту. 3. Переходить на вкладку майлстоунів. 4. Натискає кнопку “Add Milestone”. 5. На формі, що відобразилась, вводить потрібні поля. 6. Зберігає дані.	На вкладці майлстоунів відображається новостворена контрольна точка.
7	1. Користувач авторизується на акаунт розробника. 2. Авторизований користувач переходить на сторінку проекту.	Користувачу не відображається кнопка для редагування інформації.
8	1. Користувач авторизується на акаунт розробника. 2. Переходить на сторінку проекту. 3. Натискає кнопку редагування. 4. Редагує інформацію на відповідній формі. 5. Відправляє форму.	Після закриття форми на сторінці проекту можна бачити оновлену інформацію.

4.3. Рекомендації по використанню і доопрацюванню

Щодо доопрацювання сервісу можна сказати наступне. На даній версії програмного продукту значна увага приділяється саме моменту розробки того чи іншого продукту. Описані цілі, проблеми, артефакти і статуси, тому є необхідність також надати більше функціоналу для того щоб проекти можна було достойно описати та презентувати. Наприклад додати функціонал для демонстрацій проектів у вигляді слайд шоу фотографій,

демонстрацій презентацій, а ще доцільніше – можливість надання відеоматеріалів всім проектам. Також можливим поліпшенням сервісу було б доопрацювання блогу. Наприклад надати можливість користувачам, які залишають записи, додавати до своїх текстів і додаткові матеріали такі як фото чи відео. Також значним поліпшенням блогу можна вважати і функціонал оцінок тих чи інших записів аби розробники проекту бачили прогресивні ідеї своїх фанатів та могли втілювати їх у життя вже з наступними версіями продукту. Система коментування записів у блозі також була б доречною в одній із наступних версій даного програмного забезпечення.

Окремим важливим пунктом доопрацювання є обов’язкова інтеграція спеціалізованої платіжної системи, яка б опрацьовувала і контролювала кошти користувачів, які хочуть підтримати той чи інший проект на сервісі.

Щодо рекомендацій використання, то можна сказати, що продукт доцільно використовувати невеликим командам у яких є ідеї для проектів та бажання навчитись їх реалізовувати в команді. Також при деяких поліпшеннях сервісу є можливість використання сервісу компаніями, що займаються розробкою програмного забезпечення, як бази даних проектів, які були реалізовані компанією колись, або знаходяться в процесі розробки зараз. Також сервіс можна застосовувати і командам, які мають на меті вивести свої ідеї та продукти у світ, але яким не вистачає деякої матеріальної бази, а оскільки при інтеграції спеціалізованих систем кошти будуть захищені – вірогідність таких команд досягти успіху збільшується.

ВИСНОВКИ

Метою дипломного проекту було розроблення рішення для збереження навчальних проектів у сфері інформаційних технологій з можливістю відстеження процесу розробки.

Після аналізу аналогічних рішень та інструментів розробки програмного забезпечення було вирішено розробити рішення у вигляді веб додатку.

Рішення було розроблено на мові Python з використанням фреймворку Django та інструменту Django REST Framework. Клієнтська частина сервісу розроблена на мові JavaScript за допомогою фреймворку Vue.js. Системою керування базами даних було обрано PostgreSQL.

Розроблений додаток надає можливості:

- можливість авторизації на сервісі за допомогою сервісу Bitbucket;
- можливість управління проектами, їх контрольними точками, командами та інформацією про них;
- можливість відстеження тих чи інших проектів;
- можливість матеріального винагородження проектів.

Розробка виконана у повному обсязі та відповідає вимогам, що зазначені у технічному завданні дипломного проекту. Тестування додатку виконано згідно з методикою тестування.

Розроблений додаток допоможе зацікавленим в галузі IT людям відстежувати проекти, хід, методи та засоби їх розробки, а при бажанні навіть підтримати проект матеріально.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

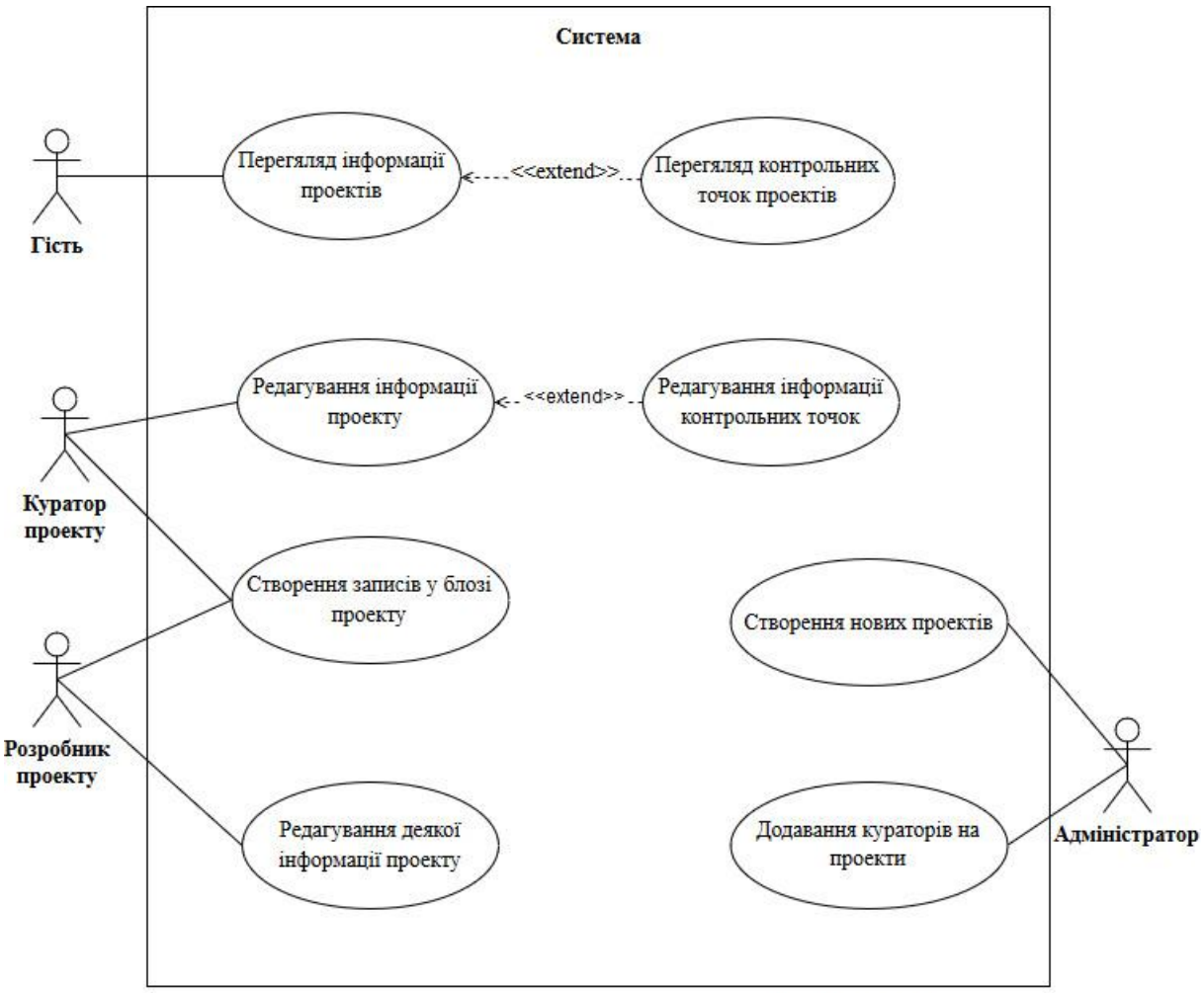
1. Asana About [Електронний ресурс]. — Режим доступу до ресурсу: https://asana.com/
2. Asana Reviews [Електронний ресурс]. — Режим доступу до ресурсу: http://tiny.cc/6wh37y
3. Projectplace About [Електронний ресурс]. — Режим доступу до ресурсу: https://www.projectplace.com/about-us/
4. Projectplace features [Електронний ресурс]. — Режим доступу до ресурсу: https://www.projectplace.com/features/
5. Projectplace: Pros & Cons [Електронний ресурс]. — Режим доступу до ресурсу: http://tiny.cc/5dj37y
6. Kickstarter About [Електронний ресурс]. — Режим доступу до ресурсу: https://www.kickstarter.com/about
7. Kickstarter: Pros & Cons [Електронний ресурс]. — Режим доступу до ресурсу: https://brandongaille.com/14-pros-and-cons-of-kickstarter/
8. Patreon About [Електронний ресурс]. — Режим доступу до ресурсу: https://www.patreon.com/about
9. What Is Patreon [Електронний ресурс]. — Режим доступу до ресурсу: https://support.patreon.com/hc/en-us/articles/204606315-What-is-Patreon-
10. What are the pros and cons of using Patreon [Електронний ресурс]. — Режим доступу до ресурсу: http://tiny.cc/9xm37y
11. Desktop vs Web Applications [Електронний ресурс]. — Режим доступу до ресурсу: https://www.seguetech.com/desktop-vs-web-applications/
12. Advantages And Disadvantages of RDBMS [Електронний ресурс]. — Режим доступу до ресурсу: http://tiny.cc/0lo37y
13. An Overview of SQL and NoSQL with it’s Pros and Cons [Електронний ресурс]. — Режим доступу до ресурсу: http://tiny.cc/4mo37y

14. MySQL [Електронний ресурс]. — Режим доступу до ресурсу: https://searchoracle.techtarget.com/definition/MySQL
15. SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems [Електронний ресурс]. — Режим доступу до ресурсу: http://tiny.cc/ptp37y
16. About Python [Електронний ресурс]. — Режим доступу до ресурсу: https://python.swaroopch.com/about_python.html
17. About JavaScript [Електронний ресурс]. — Режим доступу до ресурсу: http://tiny.cc/d8p37y
18. AngularJS Overview [Електронний ресурс]. — Режим доступу до ресурсу: https://www.tutorialspoint.com/angularjs/angularjs_overview
19. Vue.js Docs [Електронний ресурс]. — Режим доступу до ресурсу: https://vuejs.org/v2/guide/

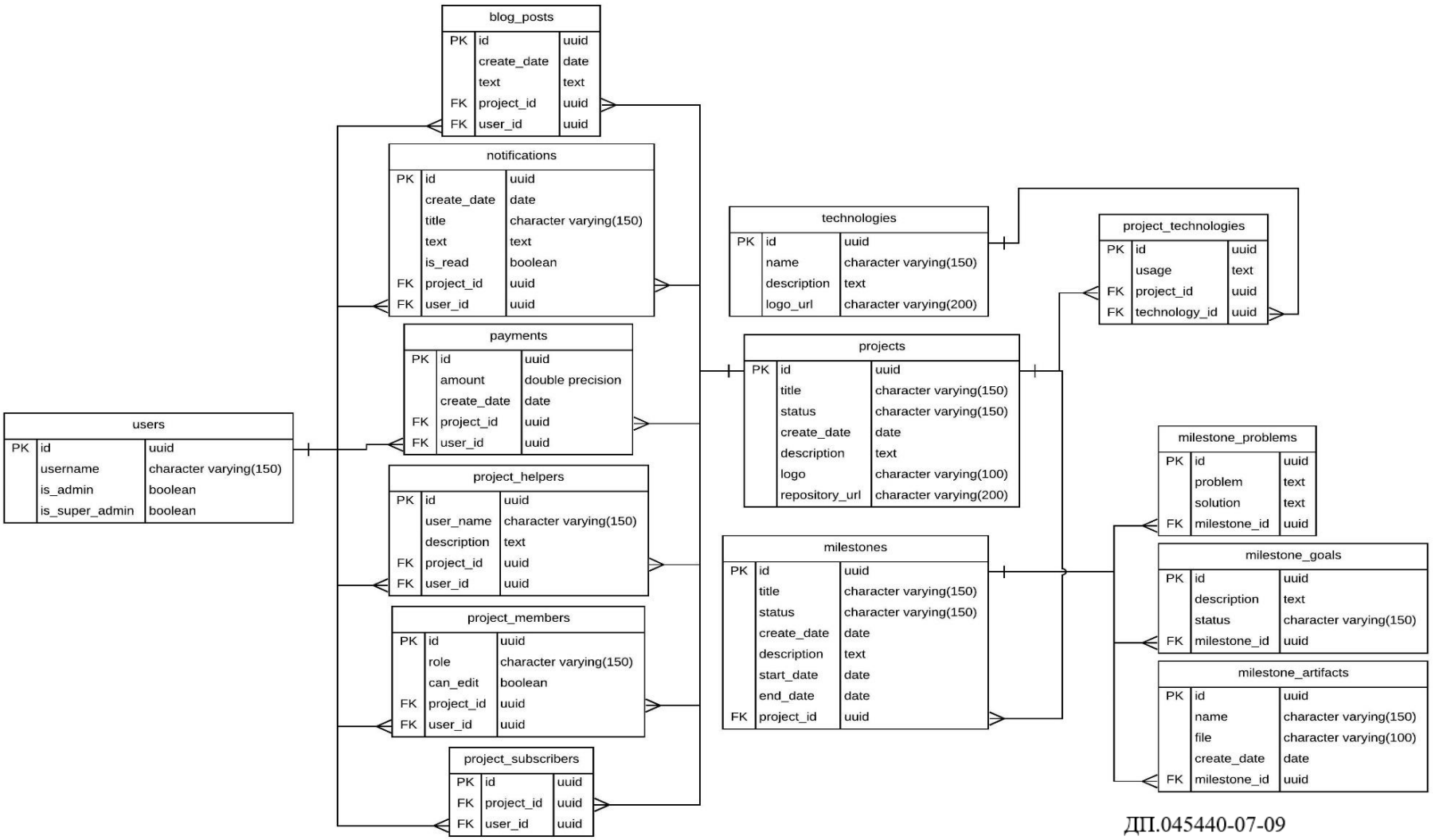
ДОДАТКИ

Додаток 1

Копії графічних матеріалів

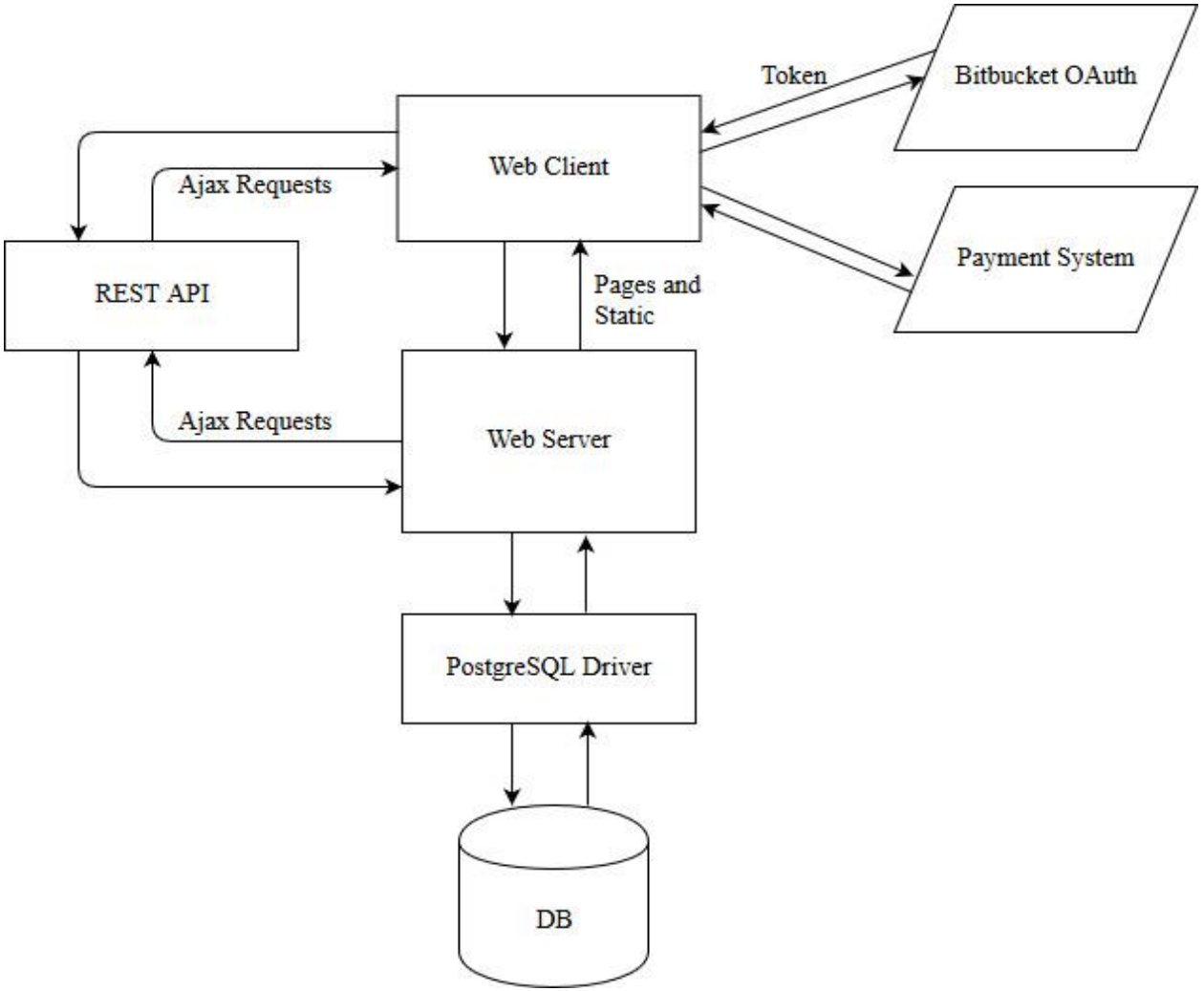


ДП.045440-06-99
Веб-сервіс навчальних проектів в сфері інформаційних технологій з можливістю відстеження процесу розроблення. Функціональність розробленої системи. UML діаграма прецедентів



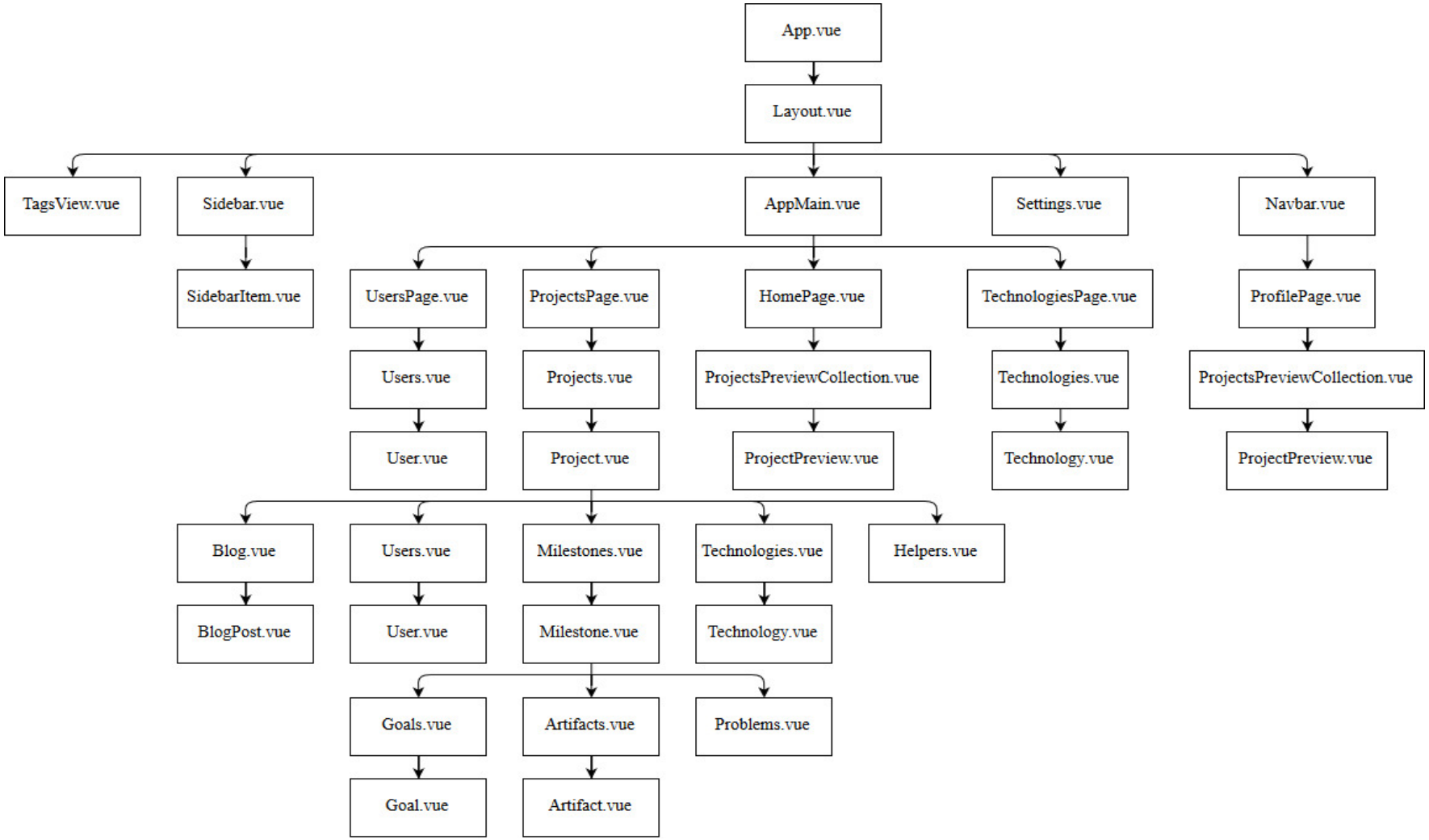
ДП.045440-07-09
Веб-сервіс навчальних проектів в сфері інформаційних технологій з можливістю відстеження процесу розроблення. Структура бази даних. ERD діаграма

СТРУКТУРНА СХЕМА СИСТЕМИ



Волощенко О.Є., група КП-51

СХЕМА VUE.JS КОМПОНЕНТІВ



Волощенко О.Є., група КП-51

Додаток 2

Копія презентації

Веб-сервіс навчальних проектів у сфері інформаційних технологій з можливістю відстеження процесу розробки

Виконав: студент 4 курсу, групи КП-51
Волощенко Олександр Євгенович

Керівник: старший викладач кафедри ПЗКС
Гадиняк Руслан Анатолійович

Постановка задачі

Мета дипломного проекту: розроблення сховища даних навчальних проектів у сфері інформаційних технологій з можливістю відстеження процесу розробки.

Завдання:

- огляд та аналіз готових програмних рішень та технологій
- вибір архітектури розроблюваної системи
- вибір засобів реалізації програмної частини проекту
- реалізація програмної частини
- тестування та аналіз отриманих результатів

Актуальність

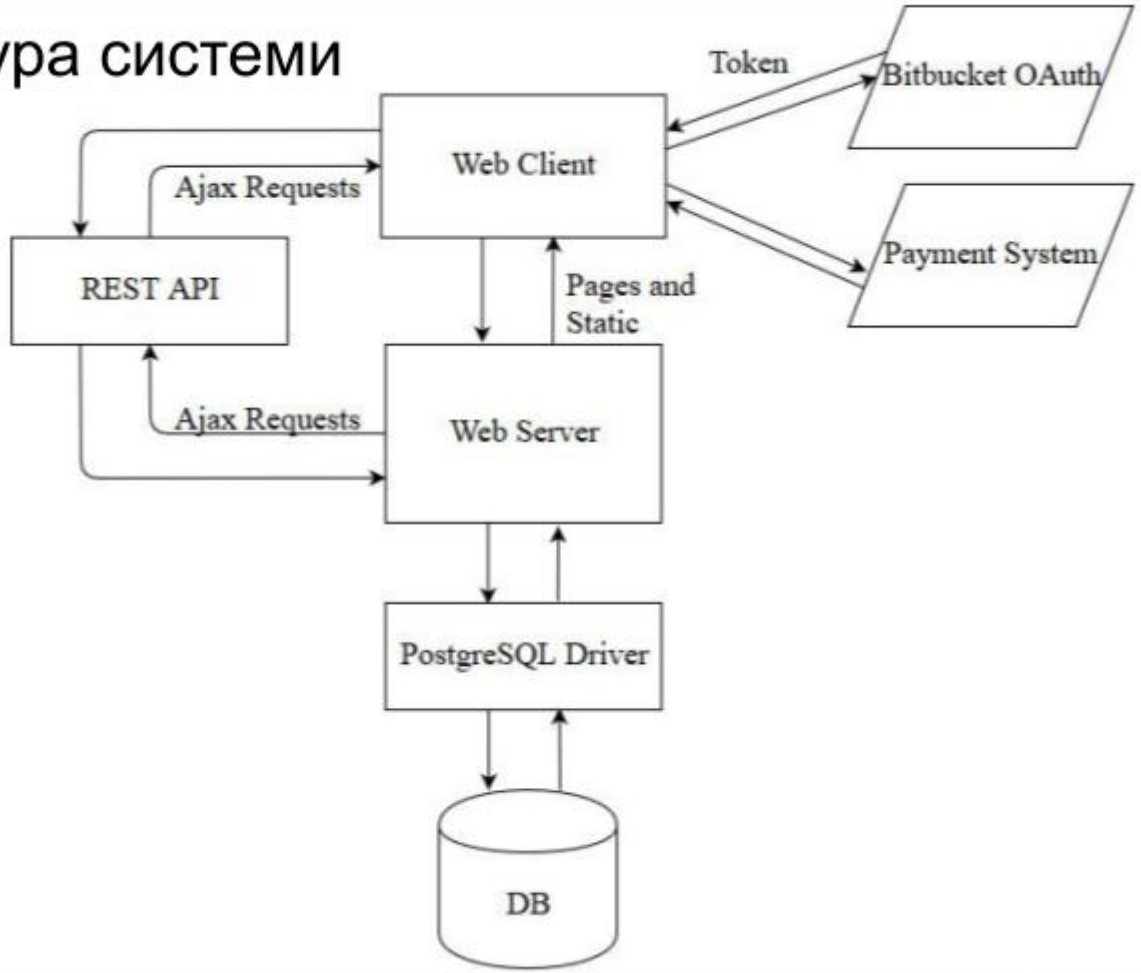
- Проблема відстеження проектів
  - Відстеження статусів
  - Відстеження процесу розроблення
  - Відстеження інструментів та технологій
- Проблема відстеження контрольних точок
  - Відстеження цілей
  - Відстеження проблем та їх рішень
  - Відстеження артефактів

Розглянуті аналоги

Asana

Kickstarter

Архітектура системи



Засоби реалізації

Мова програмування та технології розробки веб-застосунків:

Для back-end частини було обрано мову програмування Python, фреймворк Django та інструмент Django REST Framework. Для frontend частини було обрано мову JavaScript та фреймворк Vue.js.

Система керування базою даних:

Для реалізації дипломного проекту було обрано СКБД PostgreSQL.

Схема бази даних

Схема веб-серверу

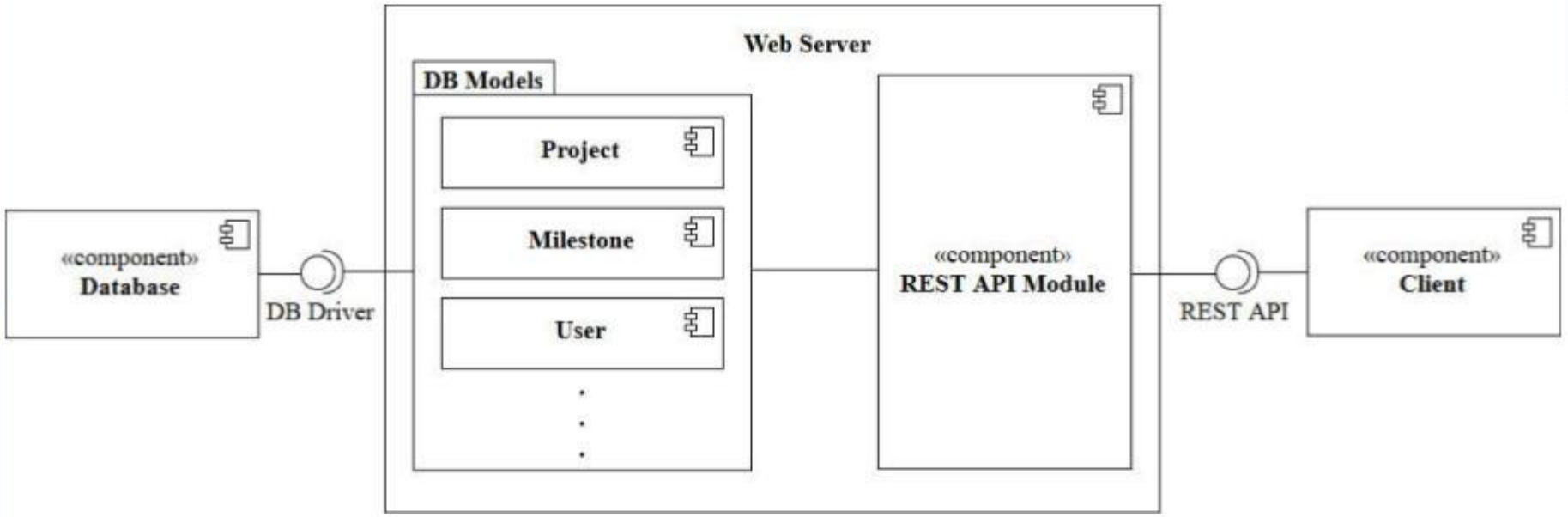
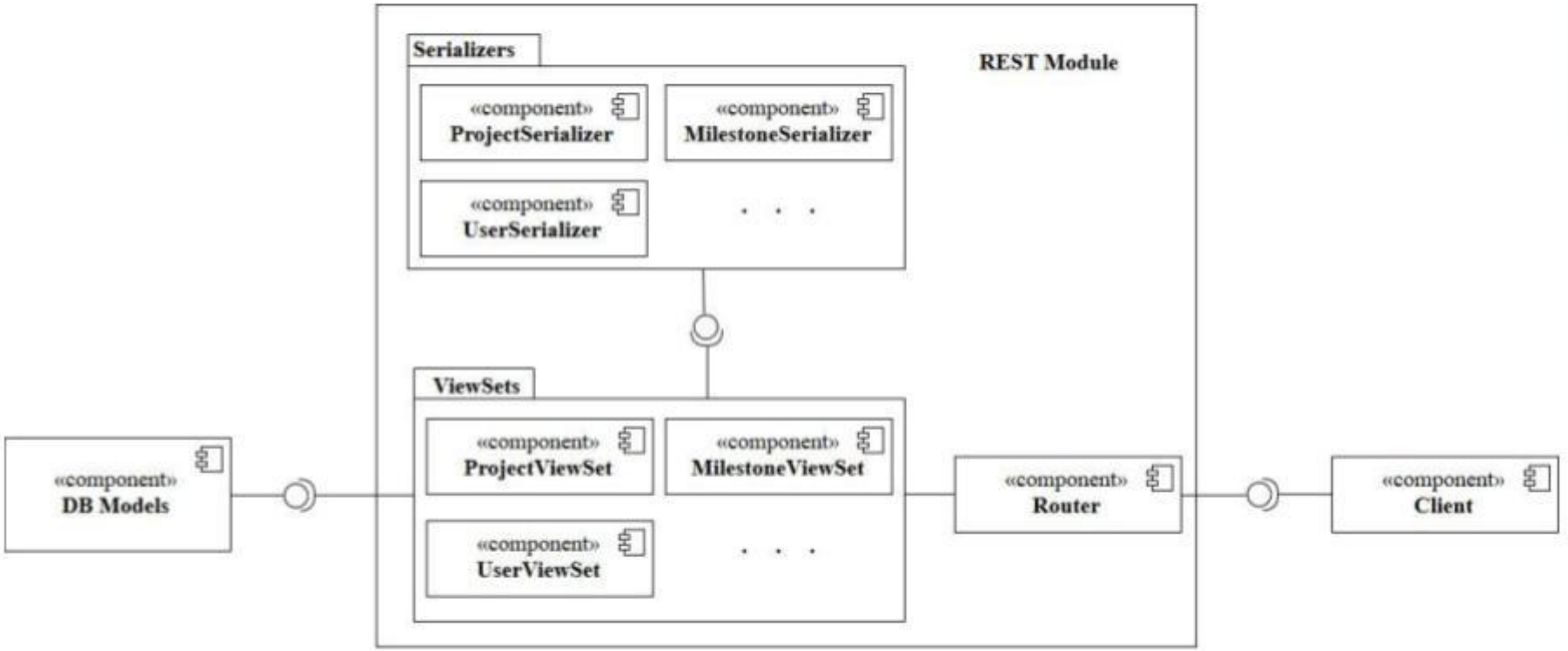
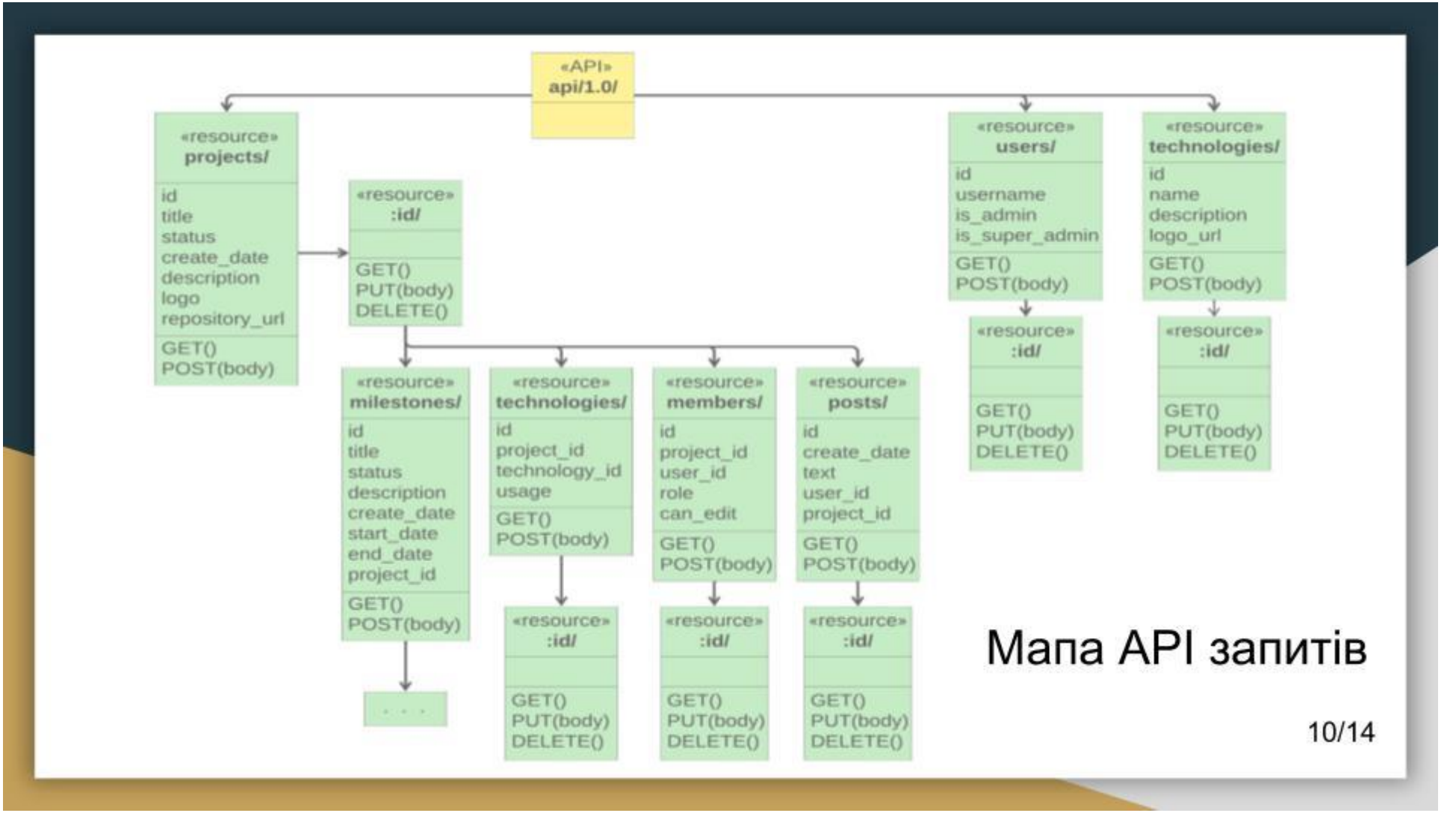


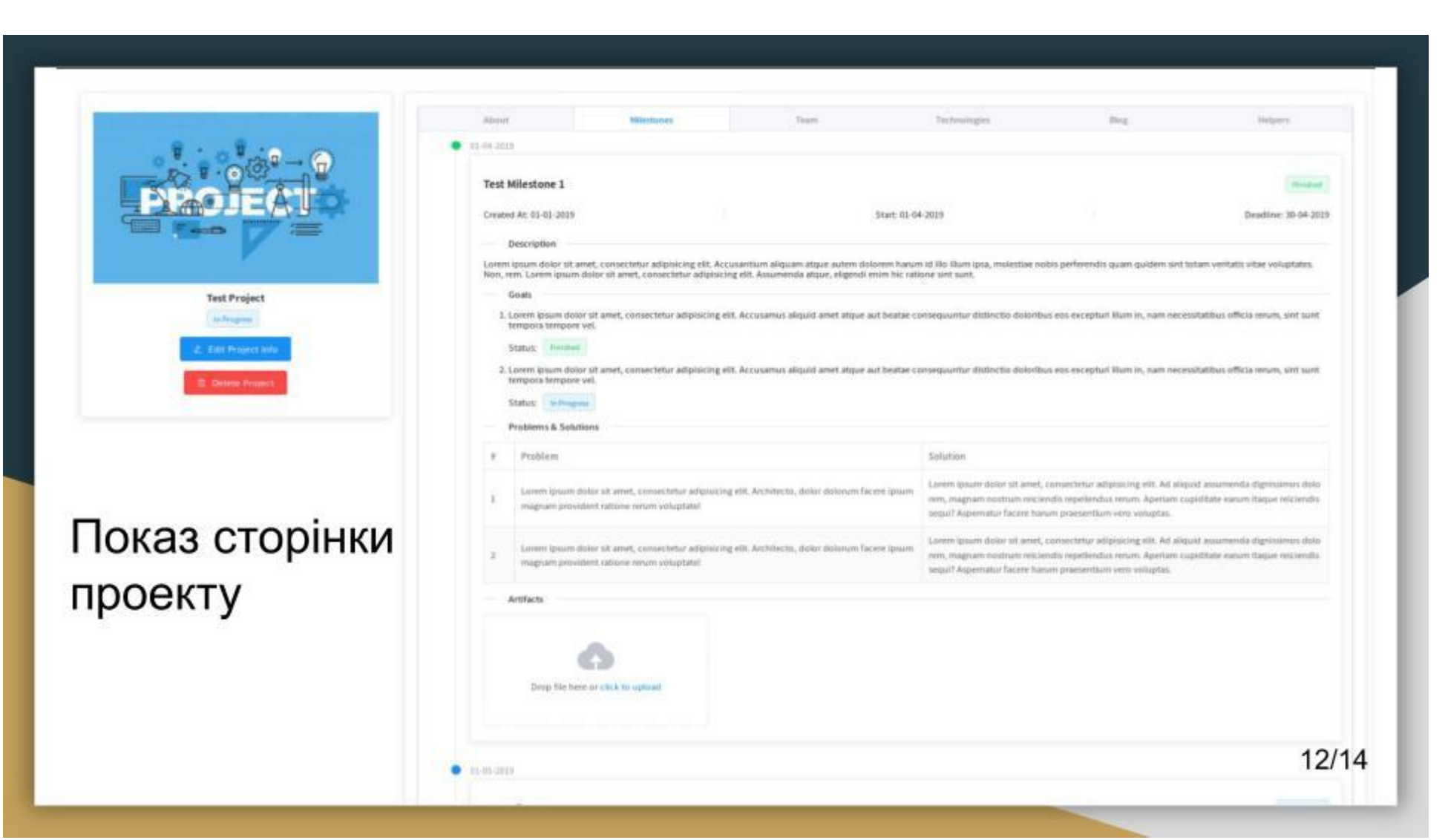
Схема REST модулю





Мапа API запитів

Схема Vue.js компонентів



Показ сторінки проекту

Тестування системи

Деякі з сценаріїв тестування:

1. Перевірка авторизації через сторонній сервіс.
2. Перевірка створення нових проектів.
3. Перевірка додавання кураторів та розробників на проекти.
4. Перевірка оновлення інформацію про проект, контрольні точки проекту та інформацію про них
5. Перевірка створення записів у блозі проекту
6. ….

Результати

- ✓ Проаналізовано аналоги системи
- ✓ Створено архітектуру системи
- ✓ Створено базу даних проектів системи
- ✓ Реалізовано модуль взаємодії з БД та модуль REST API.
- ✓ Реалізовано клієнтську частину системи
- ✓ Проведено тестування веб-додатку

Дякую за увагу!

Додаток 3

Лістинг програми

```
import uuid
from django.db import models

# region statuses

NOT_STARTED = 'not_started'
IN_PROGRESS = 'in_progress'
FINISHED = 'finished'
CLOSED = 'closed'

STATUSES = [
    (NOT_STARTED, 'Not Started'),
    (IN_PROGRESS, 'In Progress'),
    (FINISHED, 'Finished'),
    (CLOSED, 'Closed')
]

# endregion

# region roles

CURATOR = 'curator'
DEVELOPER = 'developer'

ROLES = [
    (CURATOR, 'Curator'),
    (DEVELOPER, 'Developer'),
]

# endregion


def artifact_directory_path(instance, filename):
    return 'artifacts/{0}/{1}'.format(instance.id, filename)
```

```
class Milestone(models.Model):

    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)

    title = models.CharField(max_length=150)

    status = models.CharField(max_length=150, choices=STATUSES,
default=NOT_STARTED)

    create_date = models.DateField(auto_now_add=True)

    description = models.TextField()

    start_date = models.DateField()

    end_date = models.DateField()


    project = models.ForeignKey('Project', on_delete=models.CASCADE,
db_column='project_id')


    def __str__(self):

        return 'Milestone "{0}" from "{1}" project'.format(self.title,
self.project.title)


    class Meta:

        db_table = 'milestones'

        ordering = ['start_date']

        app_label = 'projectset_app'



class MilestoneArtifact(models.Model):

    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)

    name = models.CharField(max_length=150)

    file = models.FileField(upload_to=artifact_directory_path)

    create_date = models.DateField(auto_now_add=True)


    milestone = models.ForeignKey(Milestone, on_delete=models.CASCADE,
db_column='milestone_id')


    def __str__(self):

        return 'Artifact {} from {} milestone'.format(self.name,
self.milestone.title)


    class Meta:

        db_table = 'milestone_artifacts'
```

```
        ordering = ['create_date']

        app_label = 'projectset_app'

class MilestoneGoal(models.Model):

    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)

    description = models.TextField()

    status = models.CharField(max_length=150, choices=STATUSES,
default=NOT_STARTED)

    milestone = models.ForeignKey(Milestone, on_delete=models.CASCADE,
db_column='milestone_id')


    def __str__(self):

        return 'Goal # {} from {} milestone'.format(self.id,
self.milestone.title)


    class Meta:

        db_table = 'milestone_goals'

        app_label = 'projectset_app'



class MilestoneProblem(models.Model):

    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)

    problem = models.TextField()

    solution = models.TextField()

    milestone = models.ForeignKey(Milestone, on_delete=models.CASCADE,
db_column='milestone_id')


    def __str__(self):

        return 'Problem # {} from {} milestone'.format(self.id,
self.milestone.title)


    class Meta:

        db_table = 'milestone_problems'

        app_label = 'projectset_app'



def logo_directory_path(instance, filename):

    return 'logos/{0}/{1}'.format(instance.id, filename)
```

```
class Project(models.Model):

    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)

    title = models.CharField(max_length=150)

    status = models.CharField(max_length=150, choices=STATUSES,
default=NOT_STARTED)

    create_date = models.DateField(auto_now_add=True)

    description = models.TextField()

    logo = models.ImageField(upload_to=logo_directory_path, blank=True)

    repository_url = models.URLField(blank=True)


    members = models.ManyToManyField('User', through='ProjectMember',
related_name='members')

    blog_posts = models.ManyToManyField('User', through='BlogPost',
related_name='blog_posts')

    technologies = models.ManyToManyField('Technology',
through='ProjectTechnology')

    helpers = models.ManyToManyField('User', through='ProjectHelper',
related_name='helpers')

    subscribers = models.ManyToManyField('User', through='ProjectSubscriber',
related_name='subscribers')


    def __str__(self):

        return 'Project "{}"'.format(self.title)


    class Meta:

        db_table = 'projects'

        ordering = ['-create_date']

        app_label = 'projectset_app'



class Payment(models.Model):

    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)

    amount = models.FloatField()

    create_date = models.DateField(auto_now_add=True)


    project = models.ForeignKey('Project', on_delete=models.CASCADE,
db_column='project_id')
```

```
    user = models.ForeignKey('User', on_delete=models.CASCADE,
db_column='user_id')


    def __str__(self):
        return 'Payment from {} to "{}" project'.format(self.user.username,
self.project.title)


    class Meta:
        db_table = 'payments'
        ordering = ['-create_date']
        app_label = 'projectset_app'



class Technology(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)
    name = models.CharField(max_length=150)
    description = models.TextField()
    logo_url = models.URLField()


    def __str__(self):
        return '{}'.format(self.name)


    class Meta:
        db_table = 'technologies'
        app_label = 'projectset_app'



class ProjectTechnology(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)
    usage = models.TextField()


    project = models.ForeignKey('Project', on_delete=models.CASCADE,
db_column='project_id')
    technology = models.ForeignKey('Technology', on_delete=models.CASCADE,
db_column='technology_id')


    def __str__(self):
```

```
        return '{} in "{}" project'.format(self.technology.name,
self.project.title)


    class Meta:
        db_table = 'project_technologies'
        app_label = 'projectset_app'




class User(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)
    username = models.CharField(max_length=150)
    is_admin = models.BooleanField(default=False)
    is_super_admin = models.BooleanField(default=False)

    notifications = models.ManyToManyField('Project', through='Notification')

    def __str__(self):
        return self.username

    class Meta:
        db_table = 'users'
        app_label = 'projectset_app'



class ProjectMember(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)
    role = models.CharField(max_length=150, choices=ROLES, default=DEVELOPER)
    can_edit = models.BooleanField(default=False)

    project = models.ForeignKey('Project', on_delete=models.CASCADE,
db_column='project_id')
    user = models.ForeignKey('User', on_delete=models.CASCADE,
db_column='user_id')

    def __str__(self):
        return '{} in "{}" project'.format(self.user.username,
self.project.title)
```

```
    class Meta:
        db_table = 'project_members'
        app_label = 'projectset_app'


class ProjectHelper(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)
    user_name = models.CharField(max_length=150)
    description = models.TextField()

    project = models.ForeignKey('Project', on_delete=models.CASCADE,
db_column='project_id')
    user = models.ForeignKey('User', on_delete=models.CASCADE,
db_column='user_id', null=True)

    def __str__(self):
        return '{} helps "{}" project'.format(self.user.username if self.user
else self.user_name, self.project.title)

    class Meta:
        db_table = 'project_helpers'
        app_label = 'projectset_app'


class ProjectSubscriber(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)

    project = models.ForeignKey('Project', on_delete=models.CASCADE,
db_column='project_id')
    user = models.ForeignKey('User', on_delete=models.CASCADE,
db_column='user_id')

    def __str__(self):
        return '{} subscription on "{}"'.format(self.user.username,
self.project.title)

    class Meta:
```

```
        db_table = 'project_subscribers'
        app_label = 'projectset_app'


class Notification(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid.uuid4, editable=False)
    create_date = models.DateField(auto_now_add=True)
    title = models.CharField(max_length=150)
    text = models.TextField()
    is_read = models.BooleanField(default=False)

    user = models.ForeignKey('User', on_delete=models.CASCADE,
db_column='user_id')
    project = models.ForeignKey('Project', on_delete=models.CASCADE,
db_column='project_id')

    def __str__(self):
        return 'Notification "{}" to {}'.format(self.title, self.user.username)

    class Meta:
        db_table = 'notifications'
        ordering = ['-create_date']
        app_label = 'projectset_app'
```

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

ВЕБ-СЕРВІС НАВЧАЛЬНИХ ПРОЕКТІВ В СФЕРІ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ З МОЖЛИВІСТЮ ВІДСТЕЖЕННЯ ПРОЦЕСУ РОЗРОБЛЕННЯ

Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проекту:

_____ Р.А. Гадиняк

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ О.Є. Волощенко

2018

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Веб-сервіс навчальних проектів у сфері інформаційних технологій з можливістю відстеження процесу розробки. Рішення розроблено як веб-додаток на мові Python з використанням інструментів Django та Django REST Framework. Клієнтська частина розроблена за допомогою JavaScript та фреймворку Vue.js.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1. Функціональна працездатність елементів сторінок веб-застосунку.
2. Наявність доступу до бази даних проектів.
3. Коректна робота створення проектів, контрольних точок та супутніх даних.
4. Забезпечення коректної обробки REST запитів та відповідей на них.
5. Забезпечення належного рівня безпеки даних.
6. Зручність роботи з веб-застосунком.
7. Відповідність вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом White Box Testing. Таке тестування передбачає випадок, коли внутрішня структура застосунку відома. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

1. Функціональне тестування, зокрема на рівні Critical path test (тестування 20% програмного продукту, які будуть використовувати 80% користувачів).
2. Тестування продуктивності програмного забезпечення, зокрема Performance testing (тестування стабільності).
3. Тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність web-додатку перевіряється шляхом:

1. Динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати.
2. Динамічного ручного тестування на відповідність функціональним вимогам.
3. Статичного тестування коду.
4. Тестування web-ресурсу в різних web-браузерах.
5. Тестування при максимальному навантаженні.
6. Тестування стабільності роботи при різних умовах.
7. Тестування зручності використання.
8. Тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ВЕБ-СЕРВІС НАВЧАЛЬНИХ ПРОЕКТІВ В СФЕРІ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ З МОЖЛИВІСТЮ ВІДСТЕЖЕННЯ ПРОЦЕСУ РОЗРОБЛЕННЯ

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проекту:

_____ Р.А. Гадиняк

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ О.Є. Волощенко

2019

ЗМІСТ

1. Опис структури додатку

Даний застосунок для навчальних проектів у сфері інформаційних технологій з можливістю відстеження процесу розробки являє собою веб-сервіс, який складається з динамічних веб-сторінок. Основна мова веб сервісу є англійська, але можна додати мультиязичність для елементів веб-сервісу, але не для користувацьких записів.

Загалом веб-сервіс складається з наступних сторінок:

- сторінка авторизації;
- головна сторінка;
- сторінка з проектами;
- сторінка з технологіями;
- сторінка з користувачами;
- сторінка проекту.

Всі сторінки, крім сторінки авторизації, наслідують один шаблон. Згори кожної сторінки міститься header. В цьому елементі відображаються вкладки відвіданих сторінок, шлях до поточної сторінки від головної та ще додаткові елементи для зручності. Також є елмент зліва сторінки який являє собою навігаційне меню сайту.

2. Сторінка авторизації

Сторінка авторизації (рис. 1) відкривається завжди першою, якщо користувач відвідує сайт вперше, або його попередня сесія закінчилась.

Сторінка складається лише з одного елементу – форми авторизації. На формі можна побачити дві кнопки. Перша відповідає за авторизацію через сервіс Bitbucket. Друга кнопка дозволяє зайти на сервіс у ролі гостя.

Під час авторизації через Bitbucket користувача перенаправить на сам сервіс, де необхідно буде ввести логін та пароль. Після успішної авторизації на Bitbucket користувач буде перенаправлений на головну сторінку сервісу і буде вважатись авторизований.

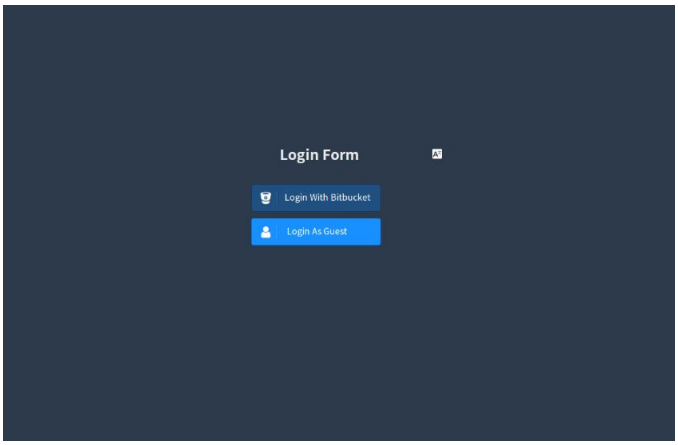


Рис. 1. Сторінка авторизації

3. Головна сторінка

Головна сторінка (рис. 2) складається з декількох частин:

– блок проектів в яких приймає участь авторизований користувач;

– блок проектів з найбільшою кількістю підписників;

– блок проектів, що були зареєстровані на сервісі нещодавно.

Для того щоб потрапити на головну сторінку сервісу необхідно в лівому меню навігації клікнути на посилання з назвою “Home”. Також на сторінку можна потрапити якщо натиснути на логотип веб-сервісу в лівому верхньому куті сайту.

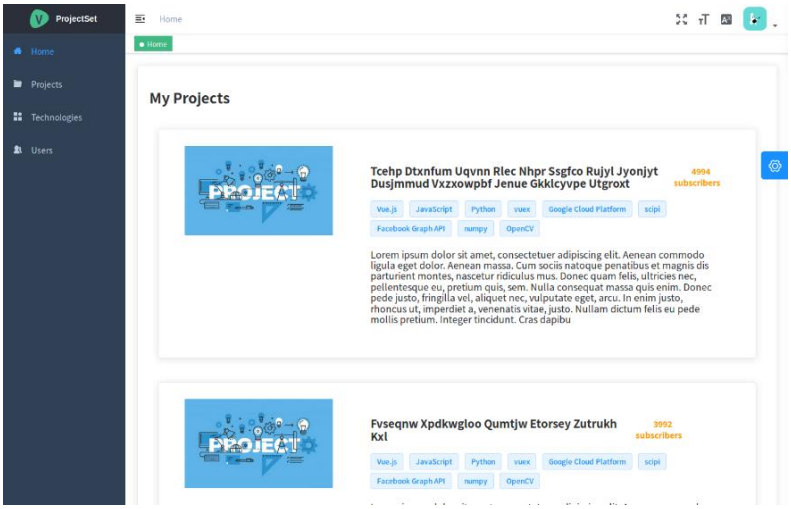


Рис. 2. Головна сторінка сервісу

4. Сторінка проектів

Сторінка проектів призначена для того щоб показати авторизованому користувачу всі проекти, які були зареєстровані на порталі. Для того щоб потрапити на сторінку необхідно в панелі меню сторінок обрати посилання з назвою “Projects”.

Проекти оформлені у вигляді невеликих блоків, які містять логотип проекту, назву проекту, короткий опис проекту, використані технології та кількість підписників даного проекту.

Щоб перейти на сторінку якогось конкретного проекту необхідно натиснути на назву проекту.

В правому верхньому куті сторінки є кнопка яка дозволить адміністратору створювати нові проекти.

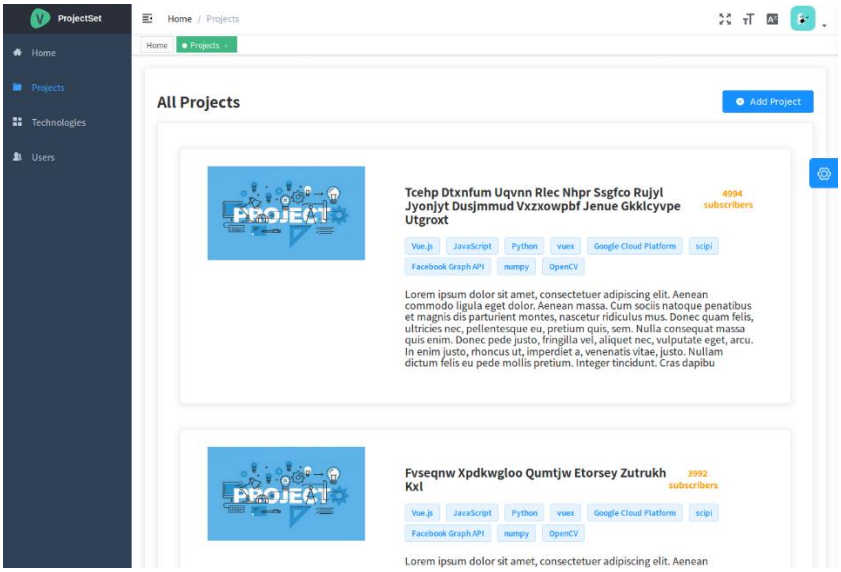


Рис. 3. Сторінка проектів

Після натискання кнопки “Add Project” перед користувачем з’явиться наступне вікно (рис. 4). Для створення проекту необхідно заповнити всі необхідні поля, а саме назва, опис та посилання на репозиторій нового проекту.

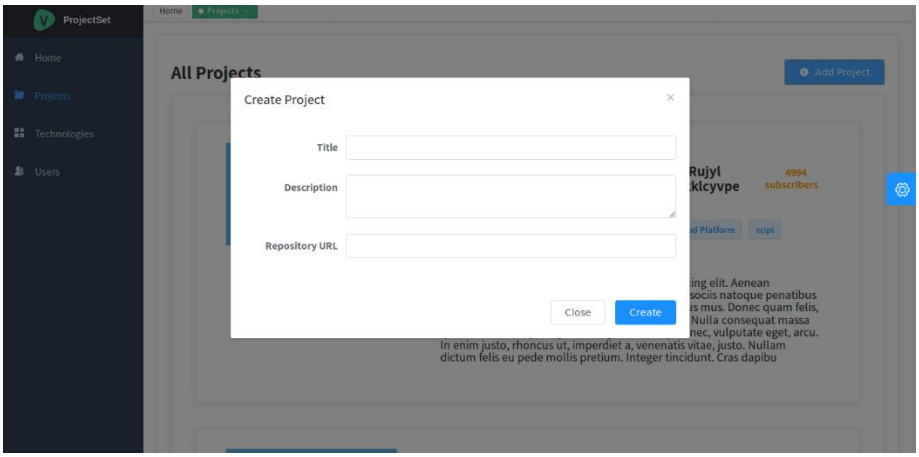


Рис. 4. Вікно створення проекту

5. Сторінка технологій

Сторінка технологій (рис. 5) призначена для того щоб відображати всі зареєстровані на сервісі технології. Для того щоб потрапити на сторінку технологій необхідно в меню навігації натиснути на посилання “Technologies”

Проекти зображені у вигляді невеликих карток, які показують логотип тої чи іншої технології, її назву та короткий опис.

Система не містить окремих сторінок під кожну окрему технологію.

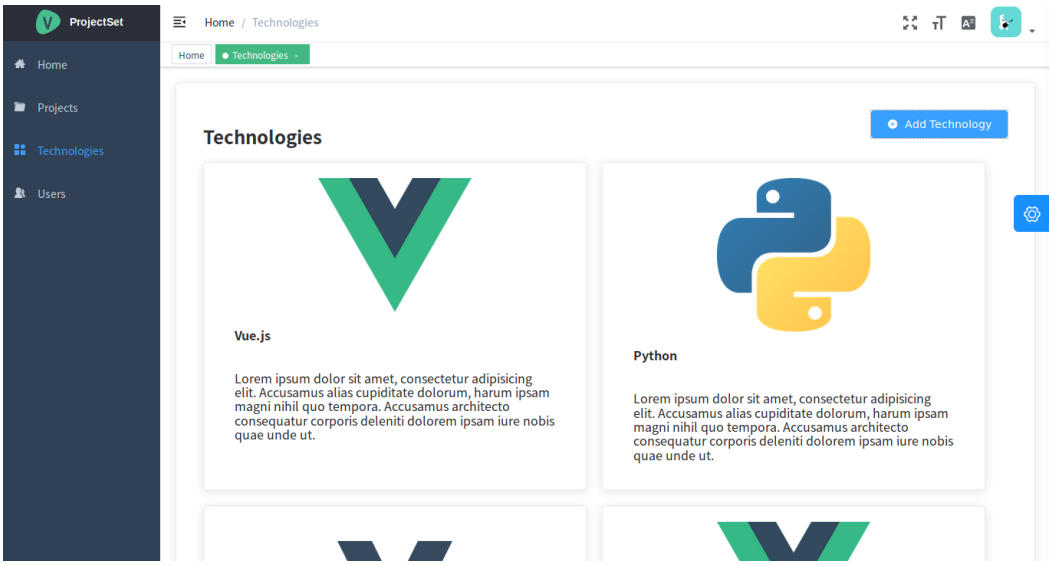


Рис. 5. Сторінка технологій

Для того щоб додати нову технологію до баз даних сервісу необхідно на сторінці натиснути на праву верхню кнопку з написом “Add Tehcnology”. Після цього перед користувачем з’явиться вікно створення технології (рис. 6) на якому йому запропонують заповнити поля назви технології, опису та посилання на логотип технологій.

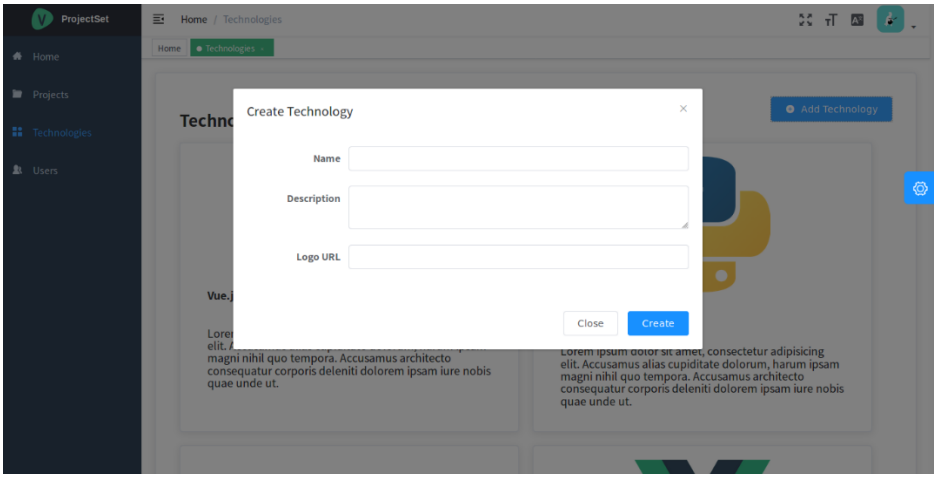


Рис. 6. Вікно створення технології

6. Сторінка проекту

Сторінка проекту (рис. 7) призначена для того щоб показати користувачу всю інформацію по проекту, яка міститься в системі.

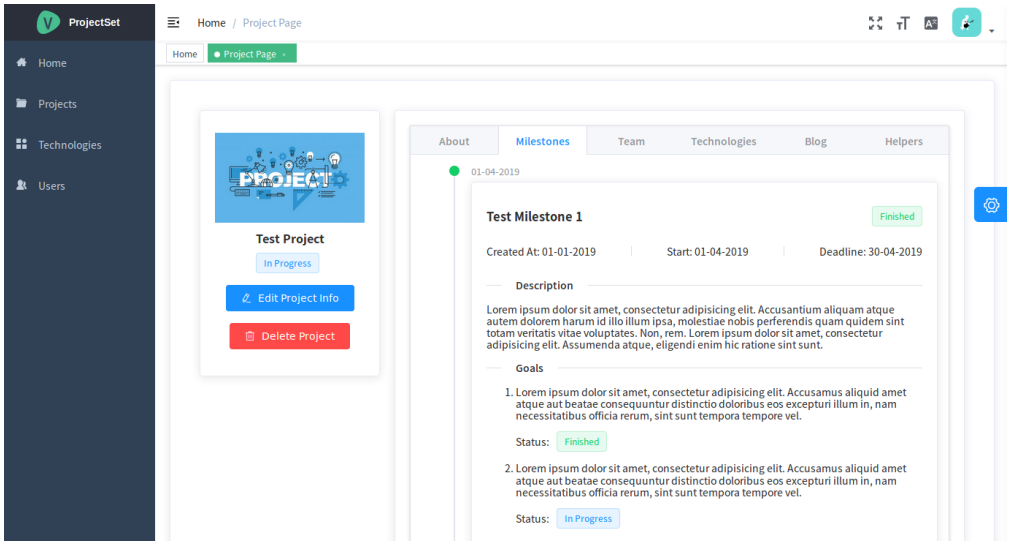


Рис. 7. Сторінка проекту

Сторінка поділяється на дві частини. В першій частині відображається логотип проекту, назва та статус. Також є додаткові кнопки, які допомагають керувати проектом. В другій частині відображається вся додаткова інформація по проекту, наприклад опис проекту, контрольні точки, команда, блог та помічники. Щоб переглянути ту чи іншу інформацію необхідно переключатись між вкладками в блоці.