КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота
на здобуття освітнього рівня бакалавра

за спеціальністю 121 Інженерія програмного забезпечення
на тему:

РОЗРОБКА ВЕБ-ДОДАТКУ ДЛЯ ПЛАНУВАННЯ ЗАДАЧ ТА УПРАВЛІННЯ ПРОЕКТАМИ

Виконала студентка 4-го курсу
Марина КУХТА _____

Науковий керівник:
доцент, кандидат фіз.-мат. наук
Максим ВЕРЕС _____

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Марина КУХТА _____

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем
«____»_____2021р.,
протокол № ___
Завідувач кафедри
Олександр ПРОВОТАР _____

Київ – 2021

РЕФЕРАТ

Обсяг роботи 74 сторінок, 41 ілюстрацій, 3 таблиці, 20 джерел посилань

ВЕБ-ДОДАТОК, УПРАВЛІННЯ ПРОЕКТАМИ, ПЛАНУВАННЯ ЗАДАЧ, JAVA, SPRING FRAMEWORK, ANGULAR, POSTGRESQL.

Об'єктом розроблення є веб-додаток для планування задач та управління проектами.

Мета дипломної роботи полягає в ознайомленні зі сферою управління проектами, аналізі сучасних технологій веб-розробки, отриманні знань та практичних навичок у роботі з технологіями для розробки веб-додатків та розробки веб-додатку для планування задач та управління проектами.

Для розробки даного додатку використовувалися такі інструменти розробки: мова програмування Java, редактор коду IntelliJ IDEA, Java Spring фреймворк, Angular фреймворк, об'єктно-реляційна система керування базами даних PostgreSQL та браузер Google Chrome.

Результат роботи: в процесі виконання дипломної роботи був проведений аналіз важливості управління проектами та розглянуті популярні існуючі на сьогоднішній день системи для вирішення цієї задачі. Були розглянуті сучасні інструменти для розробки веб-додатків, їх переваги та особливості. Сформовані функціональні вимоги до власного додатку. На основі даних вимог був розроблений додаток для планування задач та управління проектами.

Новизна результатів: програмний продукт, реалізований у ході написання дипломної роботи, відповідає критеріям сучасних веб-додатків для планування задач та управління проектами.

Результат виконання дипломної роботи - а саме веб-додаток може використовуватися як для повсякденного та і для комерційного застосування у багатьох сферах які потребують інструмент для вирішення проблеми планування.

Розроблений програмний продукт може використовуватися будь-ким, хто бажає приділити увагу плануванню та управлінню проектами як в команді так і самостійно.

Сфера застосування: фінансові послуги, торгівля, програмне забезпечення, високі технології, комерційна та некомерційна діяльність тощо.

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

MVC - Модель-вид-Контроллер,

БД - база даних,

СУБД - система управління базами даних,

ПЗ - програмне забезпечення,

DOM - Document Object Model,

IDE - інтегроване середовище розробки,

IT - інформаційні технології,

SQL - Structured query language

HTTP - Hypertext Transfer Protocol,

EOM - електронно-обчислювальна машина,

API - прикладний програмний інтерфейс

DI - Dependency-Injection

HTML - HyperText Markup Language

XML - розширювана мові розмітки

SEO - пошукова оптимізація сайту

UX - дизайн інтерфейсу користувача

UI - принцип побудови дизайну інтерфейсу користувача

JS - мова програмування JavaScript

REST - Representational State Transfer,

XHTML - Extensible Hypertext Markup Language,

HDFS - Hadoop Distributed File System,

TTL - максимальний період часу або кількість ітерацій або переходів, за який набір даних може існувати до свого зникнення.

ВСТУП

Оцінка сучасного стану об’єкта розробки. За останні роки дуже зросла популярність на діджиталізацію всіх сфер нашого життя, це безумовно вносить значні зміни у повсякденне життя користувачів. Таким чином більшість людей та компаній переходить від звичайного планування на папері до сучасних інструментів, що доступні на будь-якій платформі та пропонують безліч корисних інструментів, що покращують та прискорюють процес роботи.

Такі інструменти допомагають грамотно побудувати план роботи, мати чітке уявлення про подальші кроки для отримання необхідних результатів та поточний успіх у справах. Інструменти, що на сьогоднішній день представлені на ринку користуються популярністю серед компаній-гігантів та звичайних користувачів.

Рейтинги та відгуки кажуть про те, що з допомогою даних інструментів зростає продуктивність, зменшується ймовірність невдалого завершення роботи над проектами та сприяють покращенню умови роботи. Різноманіття даних інструментів часто з'являються на ринку через великий попит.

Актуальність роботи та підстави для її виконання. Все більше користувачів замислюються над пошуком ідеального додатка для планування задач та проектів в рамках власного життя чи роботи. Якщо раніше будь-яке планування робилося в паперовому форматі, то на сьогоднішній день дана методика відійшла над дальній план. Таким чином попит дуже зростає як і вимоги до таких систем.

Хоча наразі існує багато аналогів, що вирішували б дану проблему, всі вони мають свою недоліки, такі як платний функціонал, англомовний інтерфейс, перегруженість інструментами або навпаки не придатні для повноцінного планування всіх сфер свого життя. Тому створення даного додатку може бути корисним для користувачів, що ще не знайшли свій ідеальний інструмент. Крім того існує не багато україномовних аналогів, що могли б задовольнити багатьох користувачів.

Мета й завдання роботи. Метою даної роботи є розробка веб-додатка для планування задач та управління проектами.

Для реалізації необхідно вирішити такі задачі:

- Проаналізувати важливість управління проектами через погляд на проблему в певній галузі;
- Проаналізувати існуючі на сьогоднішній день аналоги;
- Проаналізувати сучасні технології для розробки веб-додатків;
- Проаналізувати принципи побудови веб-додатків;
- Визначити вимоги до системи;
- Розробити додаток на основі сформованих вимог .

Об'єкт, методи й засоби розроблення. Об'єктом розробки є веб-додаток для планування задач та управління проектами. Методи й засоби розробки: використання мови програмування Java, редактору коду IntelliJ IDEA, Java Spring фреймворку, Angular фреймворку, об'єктно-реляційної система керування базами даних PostgreSQL та браузеру Google Chrome

Можливі сфери застосування. Практичне застосування розробленого застосунку можливе в повсякденному житті та будь-якій комерційній та некомерційній сфері, оскільки всі вони зіштовхуються з плануванням задач та управлінням власними проектами.

РОЗДІЛ 1
ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ В СФЕРІ УПРАВЛІННЯ ПРОЕКТАМИ

1.1 Цілі та обґрунтування важливості управління проектами на прикладі сфери IT

Вперше розробники програмного забезпечення замислилися про управління програмними проектами в шістдесятих роках, підставою стала велика статистика краху проектів, що займалися розробкою програмних продуктів. Невдачі були пов’язані з затримками в реалізації ПЗ, часові та фінансові витрати в декілька разів перевищували початкові підрахунки, розробники невміло витрачали свій робочий час а менеджери часто невдало підбирали необхідну стратегію роботи команди. Всі ці провали можна було б оминути використовуючи правильний підхід в управлінні проектами. Але на жаль в ті часи не були так популярні та розвинені застосунки для розв'язання даних проблем.

Необхідність управління проектами в індустрії IT, а саме в розробці ПЗ полягає в тому, що процес завжди має бути підпорядкований бюджетній політиці компанії в якій воно розробляється та завжди має часові обмеження які не можна порушувати. Тому відповідальні керівники мають гарантувати виконання всі вищенаведених аспектів розробки в чому їм на сьогоднішній день допомагають багатофункціональні додатки.

Контроль за ходом виконання поставлених задач допомагає відслідковувати весь процес розробки і порівнювати та аналізувати показники активності команди. Моніторинг часто допомагає уникнути потенційних проблем та показати наскільки успішно розроблена частина ПЗ відповідає поставленим цілям. Таким чином, навіть якщо щось іде не по плану, керівник може змінити підхід до роботи та запобігти появі проблем в майбутньому. Хоча

добре спланована стратегія управління проектом не завжди гарантує успішне завершення проекту, та погана стратегія завжди призведе до його провалу.

Даний підхід буде корисним у будь-якій сфері.

1.2 Аналіз існуючих рішень присутніх на ринку.

На сьогоднішній день існує багато додатків які вирішують проблему планування та дозволяються спростити та пришвидшити цей процес. Всі вони мають як переваги так і недоліки для користувачів кожної галузі. Додатки пропонують різноманітні набори інструментів що більше підходять тим чи іншим користувачам.

Для подальшого аналізу далі будуть розглянуті найпопулярнішу рішення на ринку на 2021 рік.

1.2.1 Jira Software

Програмне забезпечення Jira являється одним із лідерів серед інструментів для планування задач та управління проектами для команд, що мають перед собою ціль правильно організувати свою роботу для отримання найкращих результатів.

Jira пропонує своїм клієнтам широкий ряд інструментів, що допомагають залишатися ефективними командам у таких сферах як: фінансові послуги, торгівля, програмне забезпечення, високі технології, некомерційна діяльність та багато інших. Продукт не є безкоштовним та інтегрований як для смартфонів так і для комп'ютерів. Для розширення можливостей Jira дозволяє інтеграцію з додатковими інструментами такими як: GitHub, MySQL, Oracle та інші для команд з різних галузей. Однак функціонал являється дуже складним для звичайного користувача та потребує багато часу, для того, щоб розібратися з усіма функціональними можливостями та налаштуваннями.[1]

За допомогою наданих інструментів користувачі можуть планувати роботу команди, ефективно розподіляючи завдання між учасниками, відслідковувати прогрес, відстежувати аналітику у вигляді схем та таблиць по різним критеріям та переглядати звіти спираючись на надані програмою візуальні дані та багато іншого (рис. 1.1).

Рисунок 1.1 – Інструменти для відображення діаграм в додатку Jira

Головний екран представлений на рис. 1.2. Структура Jira складається з трьох елементів: проект, завдання і під задачі. Проект - основний елемент платформи, в якому зберігаються завдання і інформація по роботі над програмою. Користувачі мають можливість створювати проекти з нуля або використовуючи готові шаблони. Для відстеження ходу роботи над проектом автоматично створюється дорожня карта. Дорожня карта проекту являє собою ієрархічну структуру, що дозволяє планувати робочий процес в різних часових перспективах, відстежувати процес виконання завдань і систематизувати роботу декількох команд над одним проектом. [2]

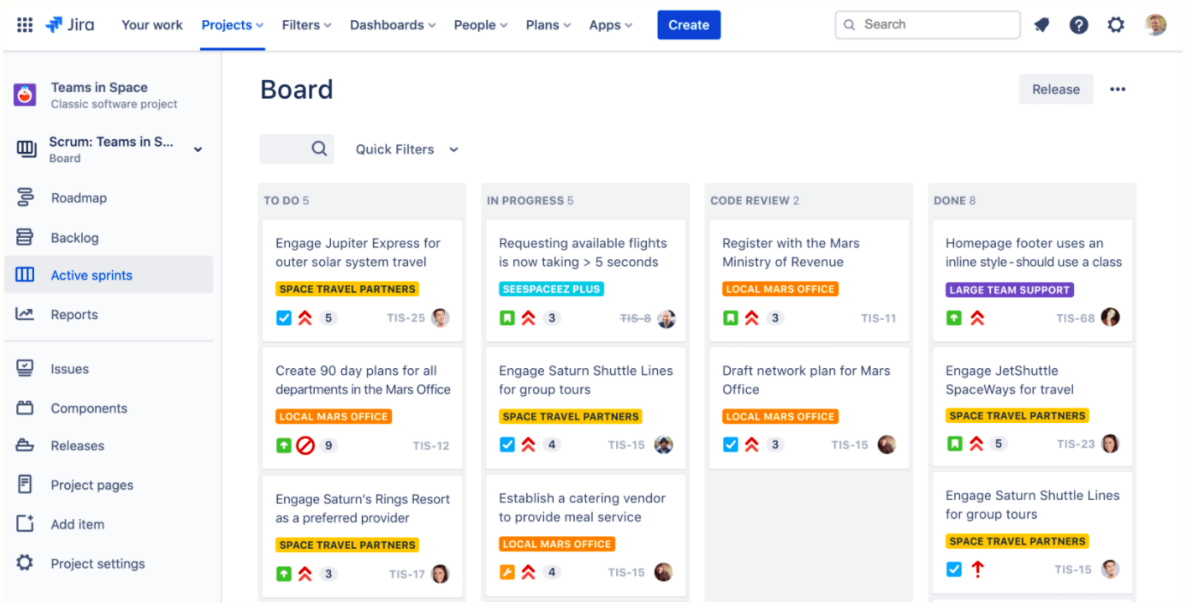

Рисунок 1.2 – Головний екран додатку Jira

Jira складається із декількох компонентів які можна налаштувати: робочий процес, типи задач, користувацький робочий простір, вікна, налаштування користувацького робочого простору та інші. Jira підтримує доски Scrum та Kanban на яких зображені поточні задачі для виконання та демонструють етап на якому знаходиться та чи інша з них. Існує два типи задач - стандартна система організації та Angile Scrum (рис. 1.3). В стандартній система організації типів задач всі нові створені задачі одразу відображаються у схемі. Задачі та проекти створені в системі Angile Scrum - використовують цю систему. Також є можливість створювати власні типи задач та налаштовувати їх для своїх потреб.

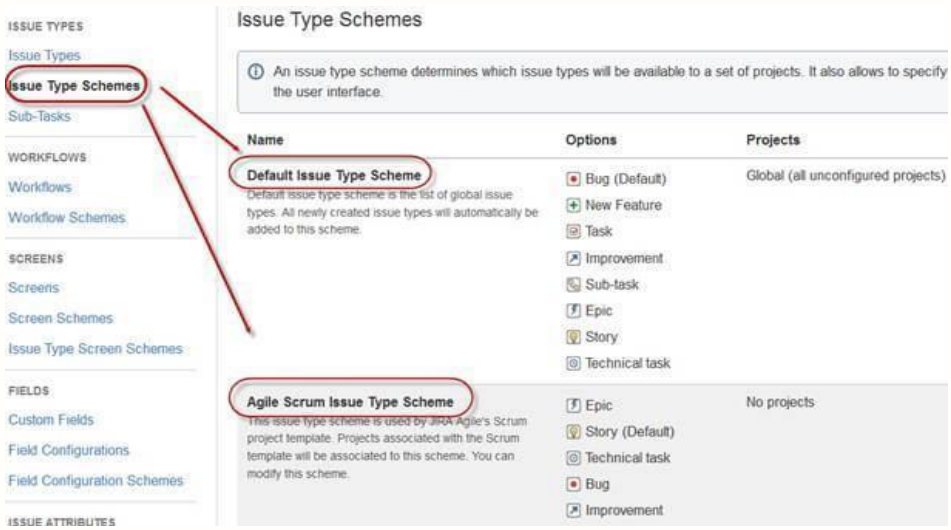

Рисунок 1.3 – Типи завдань в Jira

Компоненти створені для поділу задач проекту на розділи, це робить проект більш структурованим розділяючи його на модулі, під проекти та інше.

Робочий процес в JIRA - це набір статусів і переходів, через які проходить завдання, проблема або помилка під час свого життєвого циклу, тобто від створення завдання до його завершення. Він складається з п'яти основних стадій, а саме: відкрита задача, вирішена, в процесі, пере відкрита та закрита. Таблиця робочого процесу дозволяє переглянути, через які стадії пройшло завдання з моменту створення.

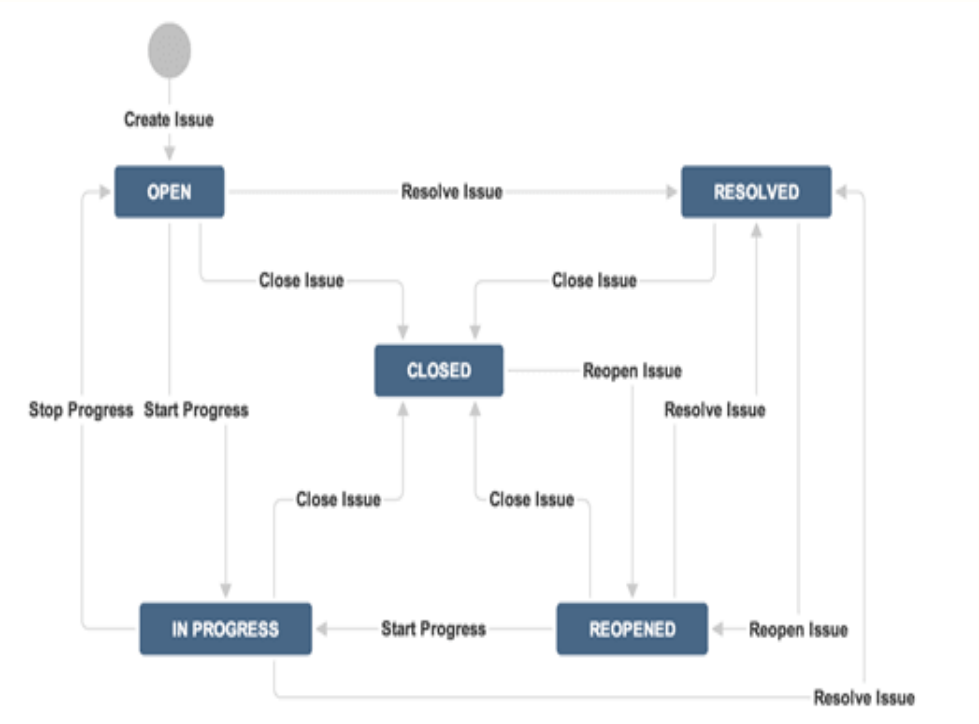

Рисунок 1.4 – Типи процесів в Jira

Розглянемо основні функціональні можливості Jira:

1) Створення проектів та задач на scrum та kanban дошках;
2) Створення власних шаблонів проектів;
3) Велика кількість налаштувань для всіх інструментів;
4) Створення дорожньої карти проекту;
5) Понад десяток встроєних agile-звітів по проектах;
6) Інтеграція з безліччю інструментів ;
7) Пошуку та фільтрації по проектам;

8) Потужна система відслідковування помилок;
9) Настроюваний життєвий цикл завдання;

1.2.2. Trello

Trello - інструмент з простим та зрозумілим функціоналом для управління проектами. Простота полягає у досить обмеженій кількості інструментів на відміну від Jira та інтуїтивно зрозумілій архітектурі, що не перевантажує користувача надмірним функціоналом та не викликає складнощів навіть у новачків.

Основна задача сервісу - підвищити продуктивність роботи команди та надати можливість швидко та зручно налаштовувати робочий процес у різних сферах. Областям застосування Trello не має обмежень, він підходить як для користування у повсякденному житті для особистого планування так і для різноманітних компаній починаючи від соціальних та закінчуючи IT. Система легко інтегрується з багатьма необхідними сторонніми сервісами: Gmail, Dropbox, Google Диск, SurveyMonkey, Slack, Evernote, Google Календар та іншими. Можливість встановити на смартфон, комп’ютер та планшет. Доступний на таких платформах як iPhone, Android, Windows, MacOS.

Зовнішній вигляд інтерфейсу головної сторінки можна побачити на рис. 1.5.

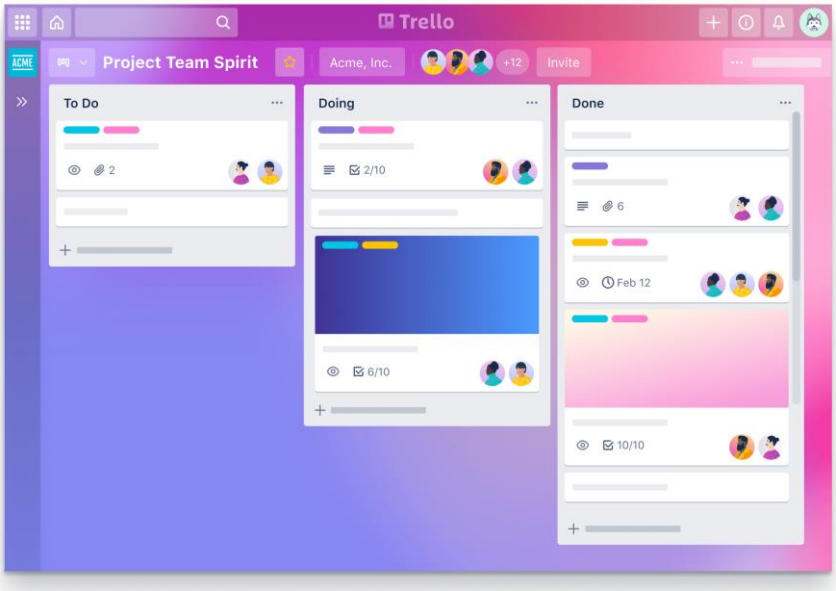

Рисунок 1.5 - Головна сторінка Trello

Основна робота зосереджена саме на ній. Інтерфейс програми побудований на канбан дошках, які використовуються для організації задач. Кожен проект - це окрема дошка на якій розміщуються картки - задачі. Дошки мають декілька видів статусів за замовчуванням -зробити, в процесі, зроблені, користувач за бажанням може доповнити цей список. Керування ними здійснюється через переміщення їх між статусами, видалення або копіювання. Також користувач може змінити рівень доступу до дошки, доступні такі варіанти конфіденційності:

1)Приватні дошки - надають можливість редагувати та видаляти дошку тільки користувачам яким наданий доступ;

2)Командні дошки - редагувати та видаляти дошку можуть тільки користувачам які додані до команди проекту;

3)Публічні дошки - доступ до змісту матимуть всі у кого є посилання а редагувати можуть тільки ті користувачі яким наданий доступ;

4)Дошка організації - приватні дошки з розширеним функціоналом, що доступний у платній версії.

Картки в більшості випадків містять у собі тіло задачі, її опис, додаткові медіа матеріали та інше або ж інформацію про об'єкт роботи. Вони мають безліч можливостей налаштувань - додавати текстові та кольорові мітки, учасників робочого простору, виконавців, змінювати обкладинку та встановлювати дедлайн. Важливо зауважити, що всі зміни які відбуваються на робочому просторі команда може відслідковувати свою активність та інших у реальному часі. Також до можливостей карток слід віднести - створення обговорень та голосувань, оскільки команда часто потребує швидкого вирішення термінових питань. Разом з тим будь-які зміни будуть відображатися у історії, що дозволяє відслідковувати прогрес та включення в роботу всіх учасників команди.[3]

Підсумовуючи виділимо такі можливості Trello:

1) Створення канбан дощок;
2) Зручне візуальне представлення завдань;
3) Можливість прикріпити до завдань медіа файли;
4) Кросплатформеність;
5) Інтегрованість зі сторонніми програмами;
6) Можливість працювати у команді;
7) Гнучкі налаштування;

Щодо мінусів системи можна виділити:

1) Не підходить для використання у великих компаніях через обмеженість функціоналу;
2) Система стає не зручною при роботі над складними проектами, оскільки важко орієнтуватися у великій кількості карток;
3) Відсутність української мови;
4) Відсутність діаграми для оцінки прогресу.

1.2.3. Any.do

Any.do - це онлайн-сервіс, що допомагає планувати робочі процеси, бізнес та особисті справи в межах одного інтерфейсу. Доступний на комп'ютері, має

десктопну та онлайн версію та мобільних простоях з операційними системами Android та Apple. Сервіс відрізняється естетичним інтерфейсом та зрозумілістю у користуванні навіть для тих хто раніше не користувався подібним програмним забезпеченням. Головний екран зображений на рис. 1.6. [4]

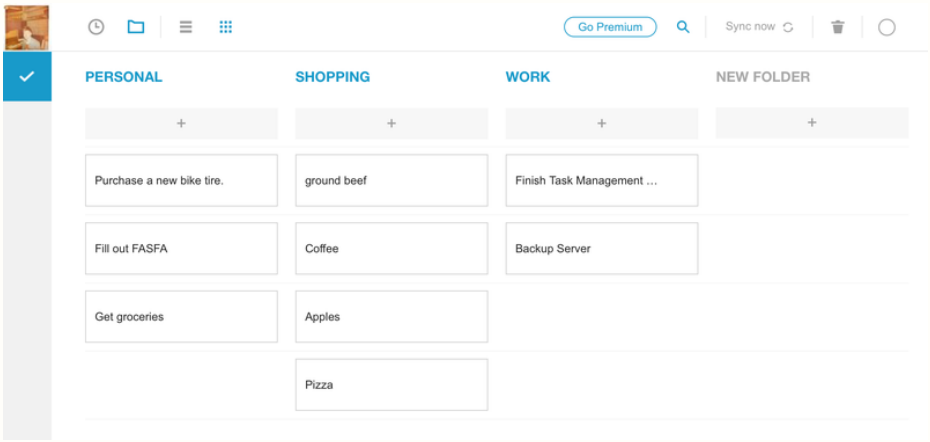

Рисунок 1.6 – Головна сторінка Any.do

Серед ключових функціональних особливостей слід відмітити синхронізацію інформації з Apple та Google календарями, Google Tasks та хмарним сховищем для полегшення доступу до сервіса з будь-яких пристроїв, розпізнавання мови за допомогою Siri або Bixby та підтримку жестів.

Планувати справи можна за допомогою списків і проектів відмічаючи їх різнокольоровими маркерами, розподіляючи по категоріям, розставляючи пріоритетність та сортуючи її.. Також можна налаштувати нагадування, встановити дедлайни та ділитися списками з іншими користувачами сервісу надаючи їм доступ для перегляду або редагування. Можливість додати віджети та розширення на головний екран девайсу робить сервіс інтерактивним в управлінні. Всі ці функції допоможуть ефективно вести роботу команди чи сімейні проекти без додаткових витрат. Приклад коллаборації з командою представлений на рис 1.7.

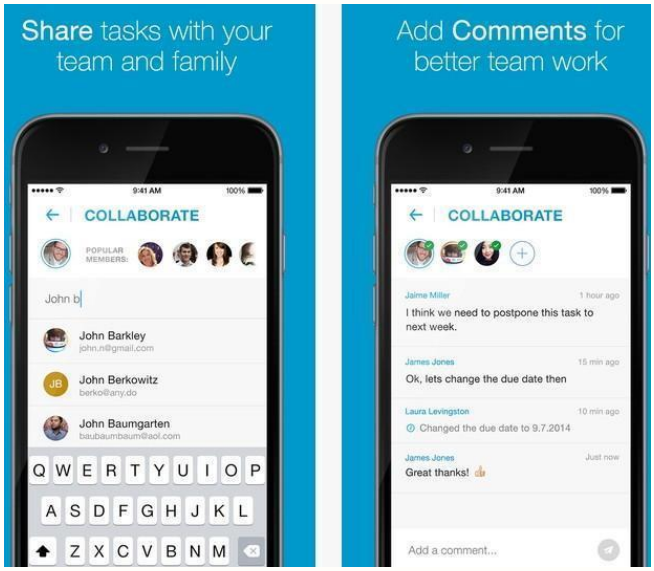

Рисунок 1.7 – Коллаборація проекту Trello

Цікавим інструментом управління списками, що доступний на мобільних пристроях є управління жестами. Так, наприклад, щоб відмітити задачу виконаною необхідно всього лиш потрясти пристрій. Для людей, яким важлива швидкість у управлінні створена функція розпізнавання мови, за допомогою якої розпізнаний текст конвертується в текст задачі. Даний функціонал зображений на рис. 1.8.[5]

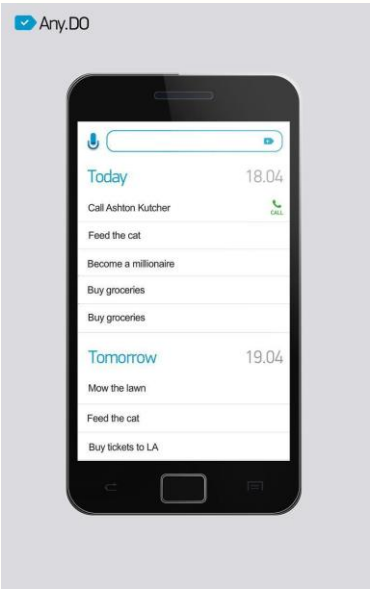

Рисунок 1.8 – Голосове розпізнавання в Any.do

Підводячи підсумки виділимо можливості сервісу:

1) Створення задач та списків;
2) Кросплатформеність;
3) Функція сортування задач;
4) Організація робочих матеріалів;
5) Синхронізація з календарями та хмарним сховищем;
6) Налаштування дедлайнів до задач, сповіщень та інше;
7) Можливість роботи в команді.

1.3 Висновки з аналізу існуючих рішень

В цьому розділі були розглянуті цілі та обґрунтована важливість управління проектами на прикладі сфери IT. Дійсно, з розвитком IT-сфери у користувачів з'явилася потреба у появі нових додатків, що допомагають їм в різних сферах справлятися з труднощами планування проектів які виникають в процесі роботи. Тому виникає необхідність розробляти такі системи і робити їх більш підлаштованими під вимоги клієнтів для широкого використання.

Було розглянуто декілька програмних продуктів, які вирішують проблему планування, усі вони мають свої відмінні риси у вигляді передбачених інструментів, архітектури та дизайну, але що у них спільне - мета. Метою є досягнення бажаних результатів в процесі роботи швидше, якісніше і з меншою кількістю помилок в результаті чітко спланованого плану дій для кожного учасника команди і постійного відстеження прогресу.

Вище були проаналізовані такі аналоги систем: Trello, Jira та Any.do. Порівняльні характеристики наведені в таблиці 1.1

Таблиця 1.1 - Порівняння між собою додатків “Jira”, “Trello” та “Any.do”

Основна функціональність	Аналоги присутні на ринку		
	Jira	Trello	Any.do
Створення проектів та задач	+	+	+

Push-up сповіщення	+	+	+
Синхронізація з календарем	+	+	+
Відстеження статусів проектів	+	+	-
Можливість швидкого освоєння системи	-	+	+
Типізація системи організації задач	+	-	-
Кросплатформеність	+	+	+
Можливість використання для роботи над великими проектами	+	+	-
Наявність інструментів для розробки ПЗ	+	-	-

Проаналізувавши всі вище розглянуті в таблиці додатки можемо дійти висновку, що вони схожі набором інструментів як і більшість на ринку, але все ж таки мають специфічні відмінності. Кожен з додатків має специфічний дизайн та підходять більше певному колу людей в залежності від цілей використання. Оскільки неможливо зробити додаток який підійде всім користувачам без обмежень, не перевантаживши його надмірною кількістю інструментів, як наприклад це зроблено в Jira, важливо не робити продукт надто складним.

З розглянутих систем найкращими виявилися Jira та Trello. Кожен з цих додатків спрямований на управління проектами в будь-якій галузі але і одночасно мають різні неперетинні інструменти з допомогою яких система стає більш придатною для певного кола користувачів.

Таким чином, при проектуванні та розробці власного додатку в рамках дипломної роботи було вирішено орієнтуватися на такі системи Jira та Trello. Наділивши свій додаток стандартним набором інструментів за допомогою яких можна буде вдало управляти проектами та планувати задачі як і в системах з яких було обрано брати приклад, не перевантаживши її надмірним функціоналом. Хоча ринок пропонує безліч додатків, що не поступаються в популярності серед користувачів обраним, слід обмежитися двома, враховуючи власне враження від даних систем.

РОЗДІЛ 2
ОПИС ТЕХНОЛОГІЙ ТА МОВ ПРОГРАМУВАННЯ ДЛЯ РІШЕННЯ ЗАДАЧІ

2.1 Опис завдання

На сьогоднішній день сфера IT розвивається у багатьох галузях і для роботи кожної з них використовується свій набір технологій та мов програмування. Кожна з технологій має свої правила застосування та використання для вирішення задач у певній галузі застосування.

Основною ідеєю побудови веб-додатків - це побудова клієнт-серверної архітектури, а саме web API. Приклад такої архітектури зображено на рис. 2.1.

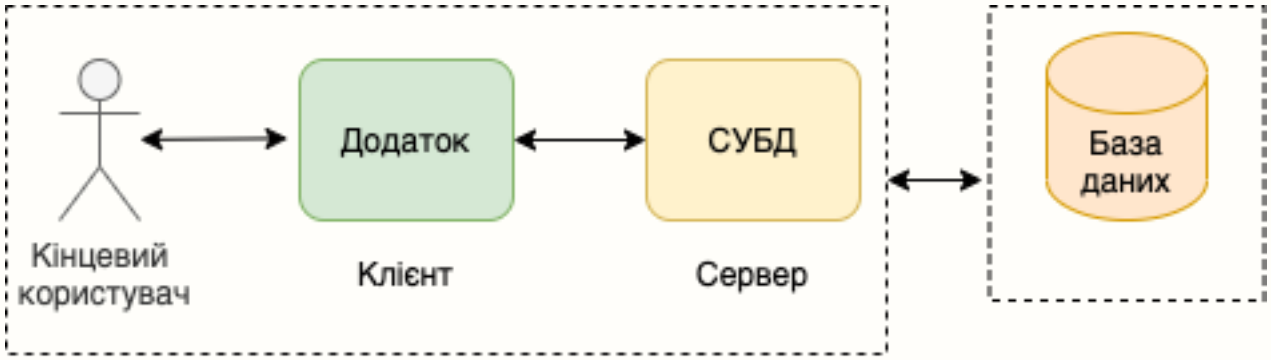

Рисунок 2.1 – Приклад клієнт-серверної архітектури

З рисунку 2.1 видно, що дана архітектура складається з таких частин:

Сервер - виступає як абстрактна ЕОМ в мережі, що здатна отримати HTTP-запит від клієнта, обробити його та повернути йому відповідь.

Клієнт - те, що може відправити та сформувати HTTP-запит та відправити його на сервер.

База даних - використовується як сховище для зберігання даних організованих відповідно до певної концепції, яка описує характеристику цих даних і взаємозв'язки між ними.

В цьому розділі розглянемо можливі технології для розробки веб-серверних додатків, проаналізуємо їх можливості та оберемо стек технологій для реалізації власного.

Мова буде йти про такі технології:

Для клієнтської частини – Angular, ReactJS, VueJS;

Для серверної частини - Java, NodeJS, Python;

Бази даних - PostgreSQL, HBase.

2.2 Java. Spring framework

Java наразі є найбільш поширеною та лідуючою в світі мовою програмування і платформою розробки.

Перша версія мови з'явилася ще в 1995 році та була розроблена компанією Sun Microsystems, згодом поглиненої компанією Oracle. Її назва спочатку була Oak лише потім її перейменували.

Java замислювалася як універсальна мова програмування, яку можна застосовувати для різного роду завдань. Спочатку цю мову використовували для програмування побутових приладів, наприклад, контролерів для переключення каналів кабельного телебачення. Такі прилади не володіють великими обчислювальними потужностями і об'ємом великої оперативної пам'яті, тому мова повинна була бути простою, здатною генерувати компактний код та бути трансльованою - переведеною в байт-код. Байт-код працює на віртуальній машині - JVM. JVM обробляє його і передає інструкції приладам, таким чином виступаючи інтерпретатором, перевагою є те , що байт-код обробляється на багато швидше ніж звичайний текст. Також було важливо не прив’язуватися до конкретної архітектури.

В основу розроблюваної мови була покладена мова C++. Тому Java запозичила деякий функціонал від C++. Мова мала реалізовувати такі риси: простоту, безпеку, об'єктно-орієнтовану модель розробки, надійність, інтерактивність, архітектурну незалежність, можливість інтерпретації, високу продуктивність і легкість у вивченні.[6]

Нижче наведені ключові концепції Java що роблять її відмінною від інших мов:

1) Java надає для широкого використання свої аплети - вони надійні, невеликі, динамічні, нативні мережеві додатки, що вбудовуються в веб-сторінки. Основна її перевага - швидкість налаштування і легке використання;
2) Підтримує об'єктно-орієнтовану модель розробки, що пов'язує простий і знайомий синтаксис з надійним і зручним у використанні середовищем розробки. Це дозволяє багатьом програмістам швидко створювати нове програмне забезпечення і аплети;
3) Надає розробникам великий різновид класів об'єктів, що допомагає легко абстрагувати багато системних функцій та створювати незалежні від платформи абстракції;

Серед основних переваг даної мови виділяють:

1) Об'єктно-орієнтованість - це дозволяє створювати складні програми, але прості в підтримці.
2) Інтерпретованість - інтерпретатор Java може виконувати байт-код безпосередньо на будь-якій машині, на яку перенесено інтерпретатор
3) Надійність - мова призначений для написання програм, які повинні надійно працювати в будь-яких умовах. Основна увага в цій мові приділяється ранньому виявленню можливих помилок, контролю в процесі виконання програми, а також усунення ситуацій, які можуть викликати помилки.
4) Незалежність від архітектури комп'ютера - скомпільована програма може виконуватися на будь-яких процесорах, а для її роботи потрібно лише виконуюча система Java.
5) Багатопотоковість - більш висока інтерактивна реакція і поведінка програм в реальному масштабі часу.

Наразі Java є лідером серед мов програмування, динамічно розвивається та надає широкий спектр інструментів, бібліотек та фреймворків для більш швидкої та ефективної розробки. Одним із популярних фреймворків є Spring.

Spring Framework - один з найпопулярніших фреймворків для створення веб-додатків на Java. Фреймворк став популярним серед Java-розробників перш

за все як альтернатива технології розробки Enterprise JavaBeans. Spring надає широкий спектр додаткових функцій яких бракує в Java EE, так, наприклад, Spring Batch, Spring MVC, Spring Boot та інші. Тим самим він надає можливість спростити розробку додатків, забезпечивши розгорнутою документацію і сучасними засоби для вирішення задач та проблем, що виникають при створенні додатків. Можливість застосувати Spring будь-якому Java-додатку з застосуванням безлічі розширень, так, наприклад, для веб-розробки, які він надає на Java Enterprise платформі робить його популярним серед розробників як ключовий фреймворк.[7]

Однією з основних особливостей Spring Framework є використання патерну Dependency Injection та управління об’єктами, які можуть бути введені. Введення залежності (DI) - є одним з основоположних принципів сучасних контейнерів програмного забезпечення. Якщо в традиційних системах компонент отримує пряме посилання на місцезнаходження необхідних для роботи об'єктів або сервісів, або звертається до сервіс-локатора і запитує посилання на реалізацію певного типу сервісу, то в разі DI контейнера оточення на основі конфігураційних даних, які можуть бути надані системи у вигляді xml-файлу або анотацій, саме знає про необхідні взаємозв'язки між компонентами, і надає потрібні об'єкти під час ініціалізації або виконання. Використання впровадження залежності вносить додаткову гнучкість в систему, оскільки полегшується створення альтернативних реалізацій сервісу, і дозволяє вказувати в конфігурації, яка саме реалізація повинна бути використана.[8][9]

Основні переваги використання контейнера, що реалізує принцип впровадження залежностей:

1) Централізоване і зовнішнє управління залежностями, в разі використання текстових конфігураційних файлів, які не підлягають компіляції, і може бути адаптоване індивідуально до різних стадій розробки і функціонування системи, не вимагаючи при цьому перекомпіляції.

2) Відсутність допоміжного сполучного коду. При використанні контейнера, що підтримує принцип DI, значно зменшується розмір коду, оскільки всі необхідні залежності і компоненти «склеюються» автоматично.
3) Поліпшення тестування окремих компонент системи. Використання dependency injection дозволяє легко замінювати залежні компоненти, що є особливо корисним для організації unit test системи.

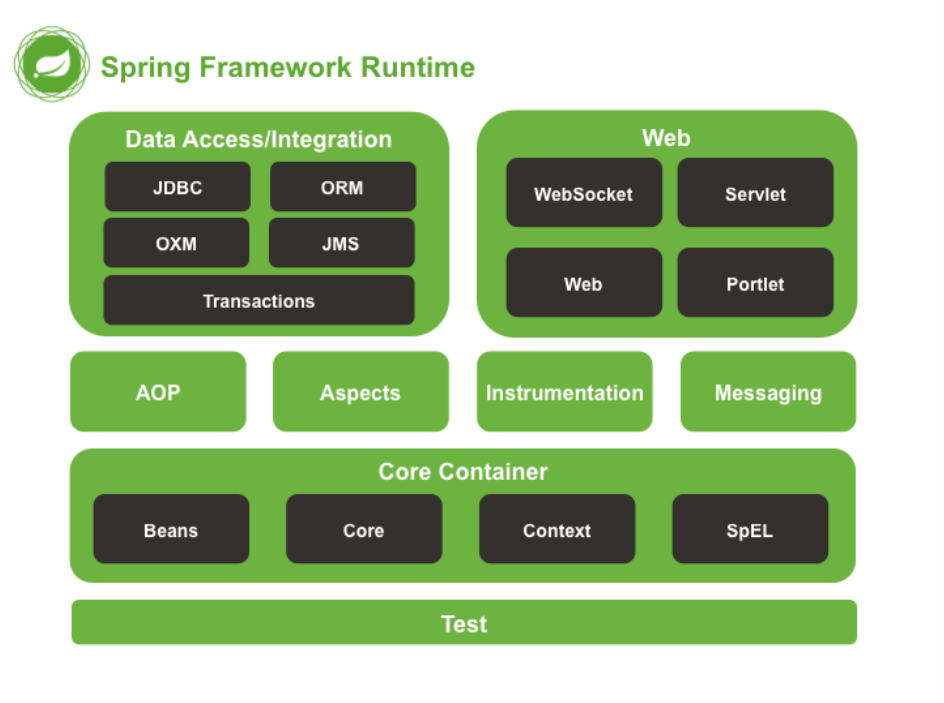

Рисунок 2.2 – Структура Spring фреймворка

На рис. 2.2 зображена структура Spring фреймворка, що складається з трьох основних рівнів:

1) Web - містить декілька фреймворків, які використовуються для полегшення завдання з розробки веб-додатків.
2) Data Access - містить інструменти для базою даних та даними.
3) Core - ядро основних концепцій фреймворку.

Таким чином, використання даного фреймворку надає потужний функціонал для більш ефективної розробки.

2.3 Python.Django

Django - це високорівневий відкритий фреймворк для розробки веб-систем на мові Python. Його структура відповідає особливостям мови. Творці реалізували в Django патерн MVC. Така архітектура дозволяє розробнику працювати з візуальним представленням і бізнес-логікою додатка окремо. При роботі з Django фахівці частіше використовують термін MVT - Model-View-Template або модель-уявлення-шаблон. Компоненти MVT можна використовувати незалежно один від одного. На рис представлена схема такої архітектури.

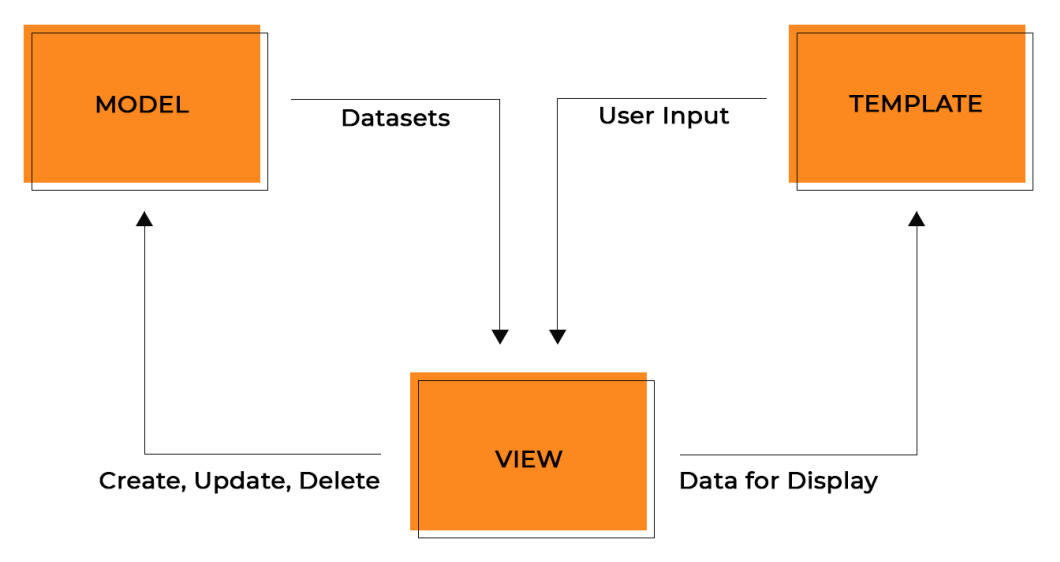

Рисунок 2.3 – архітектура патерну MVT

Модель являє собою клас, що містить інформацію про дані, їх поля та описує поведінку. Як правило модель співставляється з таблицею в базі даних. Фреймворк підтримує такі бази даних PostgreSQL, MySQL, SQLite та Oracle. Оскільки взаємодія між рівнями відбувається через API то модель нічого не знає про інші рівні.

В архітектурі модель відповідає за бізнес-логіку, методи, властивості і інші елементи, пов'язані з маніпуляцією даними. Також моделі дозволяють розробникам створювати, читати, оновлювати та видаляти об'єкти з бази даних.

View відповідає за маніпуляції з HTTP-запитами та реалізує бізнес-логіку за допомогою певну методів і властивостей. Тобто уявлення отримує дані від моделі і надає шаблонами (templates) доступу до цих даних або попередньо обробляє дані і потім надає до них доступ шаблонам.[10]

В Django реалізований багато шаблонів і власна мова розмітки. Шаблони являють собою файли з HTML-кодом, за допомогою якого відображаються дані. Вміст файлів може бути статичним або динамічним. Шаблони не містять бізнес-логіки. Тому вони тільки відображають дані.

Нижче описані основні переваги фреймворка, завдяки яким він став популярним:

1) Розвинена екосистема. Досвідчені розробники рекомендують сприймати Django як систему. Це означає, що фреймворк зазвичай використовується з великою кількістю сторонніх додатків. Їх можна вибирати в залежності від потреб конкретного проекту. Наприклад, блок авторизації або блок підписки на розсилку застосовується практично в кожному проекті;
2) Зрілість. Django був представлений в 2005 році. За 14 років існування він сильно змінився і удосконалився. У фреймворку постійно з'являються нові можливості, а старі удосконалюються;
3) Адміністративна панель. Автоматична генерація адміністративної панелі, що позбавляє розробника від необхідності створювати її вручну;
4) За допомогою сторонніх додатків консоль управління Django можна вдосконалити і адаптувати під потреби свого проекту. Крім того, фреймворк дозволяє налаштовувати інтерфейс дефолтной адміністративної панелі;
5) SEO-дружність. Написаний на Python код виходить читабельним і зрозумілим навіть непідготовленим людям. Це один з факторів, завдяки яким веб-додатки на Python вважаються SEO-дружніми. Django генерує семантичні URL. У додатках на Django легко реалізуються інші функції, необхідні для пошукової оптимізації;

6) Можливість розширення. Функціональність Django розширюється за допомогою плагінів. Це програмні модулі, які дозволяють швидко додати на сайт потрібну функцію. При необхідності можна відключати або замінювати плагіни, щоб пристосувати її до поточних потреб проекту;
7) Бібліотеки. У популярних мовах програмування є бібліотеки, за допомогою яких зручно вирішувати спеціальні завдання. В бібліотеках можна знайти готові рішення: функції, класи, конфігурації і так далі. Завдяки таким рішенням розширюються можливості мови, а також спрощується створення додатків;
8) Фреймворк підтримує використання бібліотек при розробці веб-додатків. Найпопулярнішу з них - Django REST Framework - спрощує роботу з API; Django CMS - зручний інструмент для управління контентом; Django-allauth - реалізує такі функції, як реєстрація, авторизація, управління обліковими записами;
9) ORM. Підтримка об'єктно-реляційного відображення (ORM), яке забезпечує взаємодію додатків з базами даних. ORM автоматично передає дані з БД, наприклад, PostgreSQL або MySQL, в об'єкти, які використовуються в коді програми.

Підводячи підсумки, можна сказати. що даний фреймворк має досить потужний функціонал для написання веб-застосунків, його використання неодмінно допоможе вирішити поставлену задачу зручно та ефективно.[11]

2.4 ReactJS

React - це JavaScript бібліотека для створення користувацьких інтерфейсів, на різних платформах, яка була розроблена програмістами з компанії Facebook і відкрита для всіх бажаючих. Це проста і продуктивна бібліотека JavaScript, побудована на основі компонентів. Замість того щоб надавати повний набір інструментів для створення додатків, React дозволяє вибирати, що і як реалізувати в моделях даних, серверних викликах і інших функціях додатків.

Модель React широко використовує функціональні, а також об'єктно-орієнтовані концепції програмування і фокусується на компонентах як єдиній основі для розробки. Компоненти часто відповідають елементам користувальницького інтерфейсу, наприклад календарям, заголовкам, панелям навігації і іншим, вони також можуть відповідати за маршрутизацію на стороні клієнта, форматування даних, стилізацію і інші функції клієнтської програми.

Наведемо основні концепції з яких складається React-додаток:

1) Компоненти. Це інкапсульовані блоки функціональності, які являються основою в React. Вони використовують дані (властивості і стан) для рендеринга призначених для користувача інтерфейсів;
2) Елементи – об’єкти JavaScript. Дані об’єкти представляють HTML-елементи та описують DOM-елементи, приклад DOM-дерева наведений на рис. 2.4;

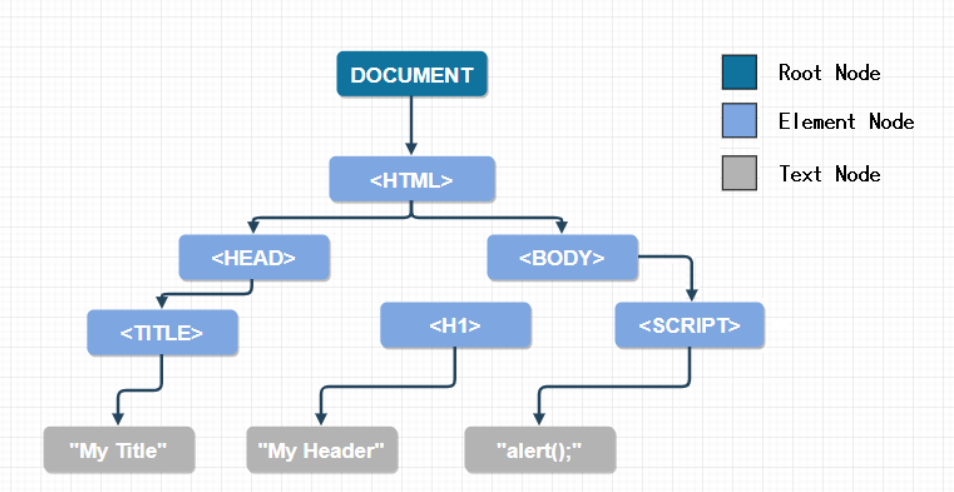

Рисунок 2.4 – Приклад DOM-дерева в ReactJS

3) Бібліотеки React. React містить набір основних бібліотек. Основна бібліотека React працює з інтерактивними бібліотеками react-dom і react-native і орієнтована на специфікацію і визначення компонентів. Вона дозволяє створювати дерево компонентів, які можна використовувати для засобів візуалізації (рендеринга) браузера або іншої платформи. React-DOM - одне з таких засобів, призначене для рендеринга в браузері і на стороні сервера. Бібліотеки React Native сфокусовані на нативних

платформах і дозволяють створювати React-додатки для iOS, Android і інших платформах;

4) Сторонні бібліотеки. React не поставляється з інструментами моделювання даних, HTTP-викликів, бібліотеками стилів або іншими поширеними елементами інтерфейсного додатка. Тому в своєму додатку необхідно задіяти додатковий код, модулі або інші інструменти. І хоча цими технологіями React не комплектується, її широка екосистема складається з неймовірно корисних бібліотек;
5) Технологія JSX - дозволяє писати код, схожий на HTML, але їм не є. Препроцесор JSX, такий як компілятор Babel, перетворює ваш JavaScript в код, сумісний зі старими браузерами. Він аналізує весь JSX-код і перетворює його в звичайний JavaScript- код. Це необхідно, тому що виконання не перетвореного JSX-коду в браузері неприпустимо - при аналізі JavaScript будуть виникати синтаксичні помилки.[12]

Основна концепція React - спростити задачі та виключити непотрібну складність при розробці. Серед основних переваг бібліотеки виділяють:

1) Високий рівень гнучкості і максимальна чуйність;
2) Віртуальна DOM (document object model), яка дозволяє впорядковувати документи форматів HTML, XHTML або XML в дерево, яке найкраще підходить веб-браузерів для аналізу різних елементів веб-додатків. На рис. 2.5 схематично зображена дані взаємодія;[13]

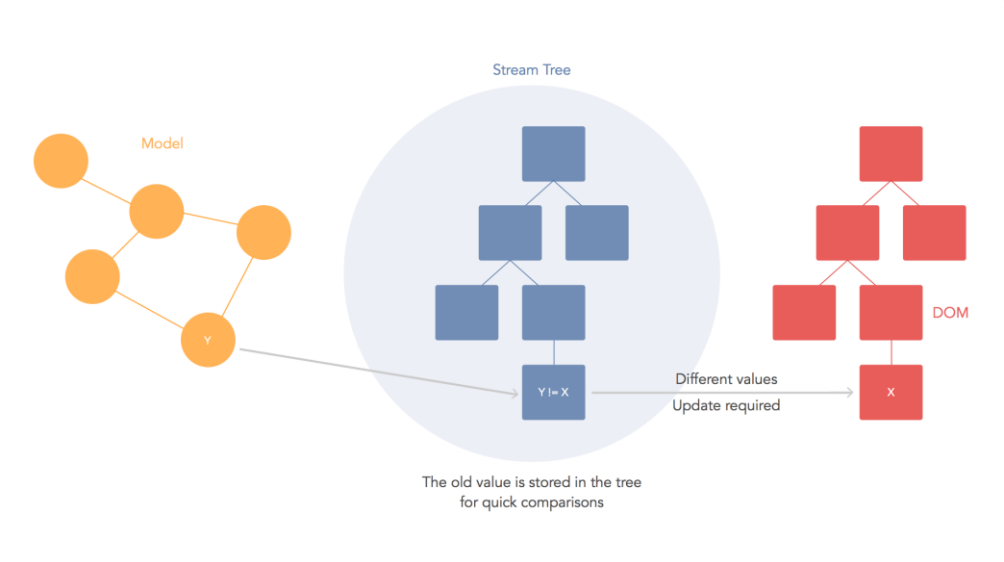

Рисунок 2.4 –Відображення DOM-дерева в ReactJS

3) Можливість зв'язувати дані від великих до менших. Таким чином потік даних реалізується так, що дочірні елементи не впливають на батьківські дані;
4) Бібліотека з відкритим вихідним кодом постійно отримує безліч щоденних оновлень і покращень відповідно до вимог розробників по всьому світу;
5) Мала вага бібліотеки, так як дані, які виконуються на стороні клієнта, можуть легко бути представлені на стороні сервера в той же самий час;
6) Проста міграція між версіями. Розробники «codemods» для автоматизації більшої частини цього процесу.

В цілому розробникам нескладно вивчити React, щоб в подальшому спростити додаток і зменшити технічні витрати за рахунок багаторазового використання коду. Бібліотека надає потужний та гнучкий інструментарій для розробки веб-додатків, також, оскільки вона розробляється і підтримується висококваліфікованими розробниками то набуває все більшої популярності з кожним роком.

2.5 VueJS

Vue.js - це прогресивний фреймворк для JavaScript, що використовується для побудови веб-інтерфейсів, розробки настільних, односторінкових та мобільних додатків. Розширення HTML і база JS швидко зробили Vue улюбленим інтерфейсним інструментом, про що свідчить його використання такими великими компаніями як Adobe, Behance, Alibaba, Gitlab та Xiaomi.

Назва фреймворку - Vue - в англійській мові таке саме фонетичне, як і view, і воно відповідає традиційній архітектурі MVC (рис. 2.5). Простіше кажучи, view - це інтерфейс користувача програми чи веб-сайту, а основна бібліотека Vue.js за замовчуванням фокусує шар представлення. Але, MVC не означає, що Vue.js не можна використовувати з іншим архітектурним підходом, таким як архітектура на основі компонентів, що використовується в React.[14]

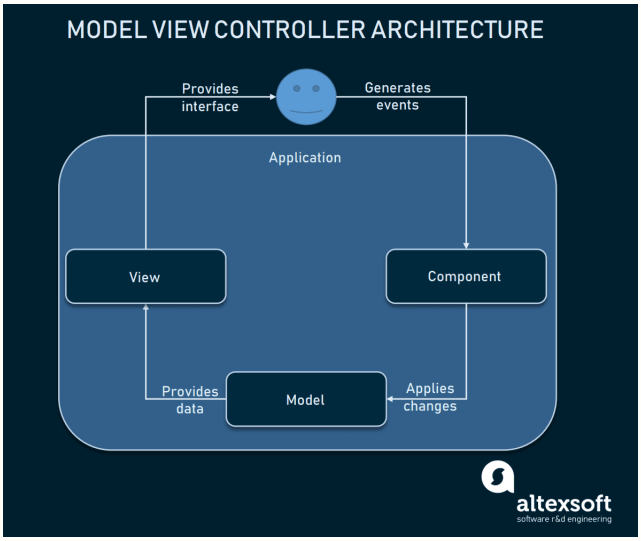

Рисунок 2.5 – MVC архітектура в VueJS

VueJS став другим найулюбленішим фреймворком у 2020 році. Подивимось, які на це причини:

1) Маленький розмір. Завантажений zip-файл із фреймворком важить 18 КБ. Фреймворк не лише швидко завантажує та встановлює бібліотеку, але також позитивно впливає на SEO та UX розробляємого продукту.

2) Візуалізація та продуктивність віртуального DOM. Об’єкт документа Модель (DOM). DOM - це представлення HTML-сторінок з їх стилями, елементами та вмістом сторінки як об’єктами. Об'єкти, що зберігаються як деревоподібна структура, створюються браузером під час завантаження сторінки. Коли користувач взаємодіє зі сторінкою, об’єкти змінюють свій стан, так що браузер повинен оновити інформацію та відобразити її на екрані. Але, оновлення всього DOM є громіздким. Для швидкості Vue.js використовує віртуальний DOM: це як копія оригінального DOM, який з’ясовує, які елементи оновлювати, замість рендеринга цілого DOM. Цей підхід робить візуалізацію сторінок досить швидким та покращує продуктивність програми.
3) Реактивне двостороннє прив’язку даних. Ще однією перевагою маніпуляцій DOM є двостороннє прив’язка даних, успадкована Vue від Angular. Двостороння прив'язка даних - це зв'язок між оновленням даних моделі та поданням (UI). Зв’язані компоненти містять дані, які можна час від часу оновлювати. За допомогою двостороннього прив’язки даних простіше оновлювати пов’язані компоненти та відстежувати оновлення даних.

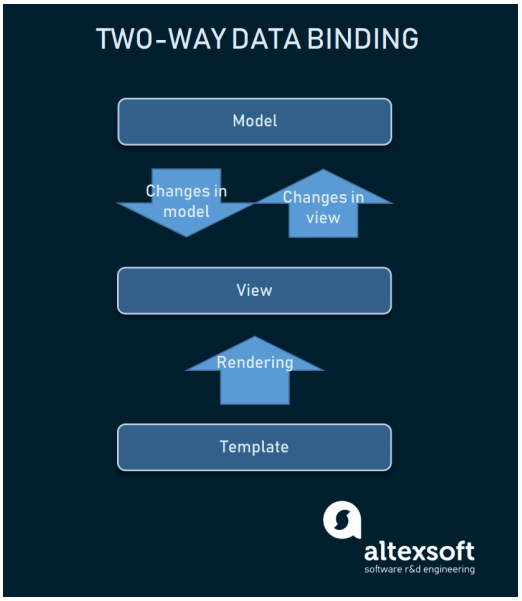

Рисунок 2.6 – Реактивна двостороння прив’язка даних в VueJS

У Vue пов'язані дані оновлюються реактивно, як і об'єкти DOM, що чудово підходить для будь-якої програми, яка вимагає оновлення в режимі реального часу. З точки зору розробників, реактивність Vue зробить оновлення даних зрозумілішим та простішим для заповнення.

Однофайлові компоненти та читабельність Кожна частина програми чи веб-сторінки у Vue є компонентом. Компоненти представляють інкапсульовані елементи інтерфейсу. У Vue.js компоненти можна писати у форматі HTML, CSS та JavaScript, не розділяючи їх на окремі файли. [15]

Розбиття коду програми насправді є архітектурним підходом, який називається Component Based Architecture, і він також використовується в Angular та React. Такий архітектурний підхід має багато переваг:

1) Багаторазове використання компонентів. Інкапсульовані компоненти - це, в основному, фрагменти коду, які можна використовувати повторно як шаблони для подібних системних елементів.
2) Зчитуваність коду. Оскільки всі компоненти зберігаються в окремих файлах (а кожен компонент - це лише один файл), код легше читати та розуміти, що полегшує обслуговування та виправлення.
3) Добре для юніт-тестування. Модульне тестування - це перевірка якості, спрямована на перевірку того, як найменші частини програми працюють самі по собі. Наявність компонентів значно спрощує це завдання.
4) Інтеграційні можливості та гнучкість. Важливим аспектом будь-якої нової технології є можливість інтеграції з іншими додатками. З Vue.js це легко, оскільки він не вимагає жодних інструментів для роботи крім JavaScript .
5) Vue також дозволяє писати шаблони за вашим бажанням: HTML, JS або JSX. Поміж своїми компонентами та полегшеною природою Vue можна використовувати практично в будь-якому проекті.
6) Екосистема надійних інструментів. За роки свого існування Vue.js отримав потужний набір інструментів для роботи, так наприклад, інструменти налагодження браузера, візуалізатор сервера та менеджер стану.

7) Легкий у вивченні. Інструмент може широко застосовуватися лише тоді, коли це легко зрозуміти, що може бути у випадку з вивченням Vue.js. Щоб розпочати кодування, Vue не вимагає глибоких знань бібліотек, JSX та TypeScript, як це часто буває з іншими інтерфейсними технологіями. Для початку вам знадобляться лише базові знання HTML, CSS та JavaScript, звичайно.
8) Найпопулярніші редактори коду, такі як Sublime text, Visual Studio та Atom, підтримують Vue, що робить спробу це легше.
9) Коротка документація. Вона добре структурований і охоплює всі можливі теми, точно описуючи все, від встановлення до більш поглиблених речей, таких як реактивність та масштабування програми.
10) Підтримка спільноти. Vue.js існує завдяки своїй спільноті. Через це учасники спільноти досить активні як у чаті Discord, так і на форумі.

2.5 Angular

Angular - JavaScript-фреймворк з відкритим програмним кодом, найбільш підходящий для вашої розробки веб-додатків а саме написання клієнтської частини. Він повністю розширюваний і добре працює з іншими бібліотеками. Кожну функцію можна змінити або замінити відповідно до вашого унікального робочого циклу розробки та потреб..

Використовує такі механізми:

Прив’язка даних. Прив’язка даних - це автоматичний спосіб оновлення подання при зміні моделі, а також оновлення моделі при зміні подання. Це чудово, оскільки виключає маніпуляції з DOM зі списку речей, про які слід турбуватися.

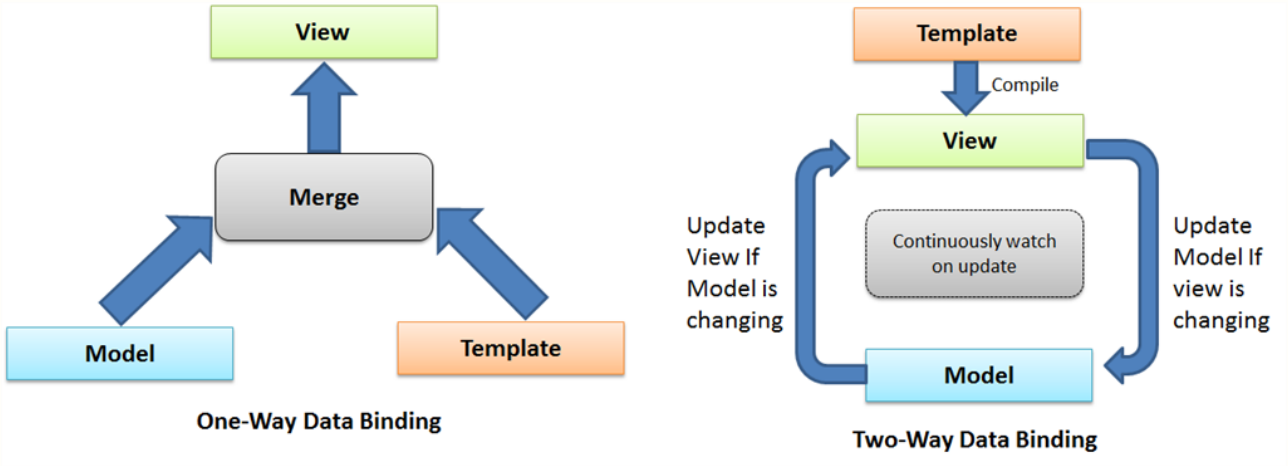

Рисунок 2.7 – принцип роботи прив'язки даних в Angular

Контролер. Контролери - це поведінка елементів DOM. Angular дозволяє висловити поведінку в читабельній формі без звичайного зразка оновлення DOM, реєстрації зворотних викликів або перегляду змін моделі.

Директиви. Директиви - це унікальна та потужна функція, доступна в Angular. Директиви дозволяють вигадувати новий синтаксис HTML, специфічний для розробляємої програми.

Багаторазові компоненти. Вони використовуються для створення компонентів, що використовуються багаторазово. Компонент дозволяє приховати складну структуру DOM, CSS та поведінку. Це дозволяє зосередитись на тому, що робить програма, або на тому, як вона виглядає окремо.

Локалізація. Важливою частиною серйозних програм є локалізація. Фільтри та встановлені директиви Angular дають вам будівельні блоки, щоб зробити вашу програму доступною у всіх мовах.

Dependency injection. Ін’єкція залежностей у Angular дозволяє декларативно описати підключення вашої програми. Це означає, що програмі не потрібен метод main(). Введення залежності також є основним елементом Angular. Це означає, що будь-який компонент, який не відповідає вашим потребам, може бути легко замінений.

Глибоке зв’язування. Пряме посилання відображає місце перебування користувача в додатку. Це корисно, щоб користувачі могли створювати закладки та посилати електронною поштою посилання на місця в додатку. Angular поєднує в собі переваги глибоких зв'язків із поведінкою, схожою на настільний додаток.

Перевірка форми. Перевірка форми на стороні клієнта є важливою частиною чудового користувацького досвіду. Angular дозволяє оголосити правила перевірки форми без необхідності писати код JavaScript. [16]

2.6 PostgreSQL

PostgreSQL - це гнучка об'єктно-реляційна система баз даних з відкритим кодом, яка використовує та розширює мову SQL. Вона поєднує у собі багато функцій, за допомогою яких дані безпечно зберігаються та масштабують найскладніші робочі навантаження.

Походження PostgreSQL бере свій початок у 1986 році в рамках проекту POSTGRES в Університеті Каліфорнії і має більше 35 років активного розвитку на базовій платформі.

PostgreSQL отримала свою репутацію завдяки перевіреній архітектурі, надійності, цілісності даних, широкому набору функцій, розширюваності відданості спільноти з відкритим кодом, що стоїть за програмним забезпеченням, для постійного забезпечення продуктивних та інноваційних рішень. PostgreSQL працює на всіх основних операційних системах, сумісний з ACID з 2001 року та має потужні доповнення, такі як популярний розширювач геопросторових баз даних PostGIS. Не дивно, що PostgreSQL став реляційною базою даних із відкритим кодом для багатьох людей та організацій.

PostgreSQL має безліч функцій, спрямованих на те, щоб допомогти розробникам створювати додатки, адміністраторам захищати цілісність даних та створювати відмовостійкі середовища, а також допомагати керувати даними незалежно від того, наскільки великим чи малим є набір даних. На додаток до

того, що PostgreSQL є безкоштовним та відкритим кодом, він дуже розширюваний. Наприклад, можна визначати власні типи даних, створювати власні функції, писати код з різних мов програмування, не перекомпільовуючи базу даних.

PostgreSQL намагається відповідати стандарту SQL там, де така відповідність не суперечать традиційним особливостям або можуть призвести до поганих архітектурних рішень. Багато функцій, які вимагає стандарт SQL, підтримуються, хоча іноді з дещо різними синтаксисом або функціями. Станом на випуск версії 13 травня 2021 р. PostgreSQL відповідає принаймні 173 із 179 обов'язкових функцій для SQL: 2016 Core conformance. На момент 2020 року жодна реляційна база даних не відповідає повній відповідності з цим стандартом. У документації PostgreSQL можна знайти багато інших функцій. Крім того, PostgreSQL є дуже розширюваним

Нижче наведено перелік різноманітних особливостей PostgreSQL:

- Функції записуються у вигляді блоків коду, які виконуються на серверній частині. Вони можуть бути написаними при використанні різних мов програмування;
- Тригер - визначають операцію, яка повинна виконуватися при настанні деякої події в базі даних, ініційовані Insert, Update і Delete операціям формулюються як функції. Наприклад, операція INSERT може запустити тригер, який повторює доданий запис згідно з конкретною умовою;
- Механізмом правил є механізм створення користувацьких обробок як Insert, Update і Delete операцій так і операцій вибірки. Правила починають працювати на етапі розбору запитів - це є їх основною відмінністю від тригерів. Також вони дозволяють змінювати поведінку системи при виконанні операцій до таблиці;
- Багатоверсійність надає декільком користувачам можливість одночасно модифікувати БД;

- Розширення PostgreSQL для власних потреб можливо багатьох відношеннях. Так наприклад можна додавати власні перетворення типів, типи даних, функції, домени, індекси, оператори і процедурні мови.;
- Спадкування в PostgreSQL реалізовано на рівні таблиць. Таблиці можуть успадковувати характеристики і набори полів від батьківських таблиць.[17]

2.7 HBase

HBase - це нереляційна, розподілена, з відкритим вихідним кодом, написана на мові Java база даних. Спочатку HBase класу NoSQL створювалася компанією Powerset в 2007 році для обробки великих обсягів даних для пошукової системи на природній мові. Вона відноситься до категорії «сімейство стовпців» і являє собою стовпчико-орієнтоване, мультіверсійне сховище типу «ключ-значення» (key-value). Вона працює поверх розподіленої файлової системи HDFS.

Модель даних HBase відрізняється від класичних реляційних СУБД, реалізуючись за типом ключ-значення - рис. 2.8.

Модель бази даних характеризується такими принципами:

- Дані організовані в таблиці, проіндексовані первинним ключем;
- Для кожного первинного ключа може зберігатися необмежений набір атрибутів - колонок;
- Колонки організовані в групи колонок. Список і назви груп колонок фіксований і має чітку схему;
- Для кожного атрибута може зберігатися кілька різних версій. Різні версії мають різний штамп часу;
- Записи фізично зберігаються в порядку, відсортованому по первинному ключу. При цьому інформація з різних колонок зберігається окремо, завдяки чому можна зчитувати дані тільки з потрібного сімейства колонок, таким чином, прискорюючи операцію читання;

- Атрибути, що належать одній групі колонок і відповідні одному ключу, фізично зберігаються як відсортований список. Будь-атрибут може бути відсутнім або бути присутнім для кожного ключа. Відсутність атрибута не тягне за собою ніяких накладних витрат на зберігання порожніх значень;
- Якщо різниця між штампом часу для певної версії і поточним часом більше TTL, така запис позначається до видалення. Аналогічно, якщо кількість версій для певного атрибута перевищено максимальну кількість версій;

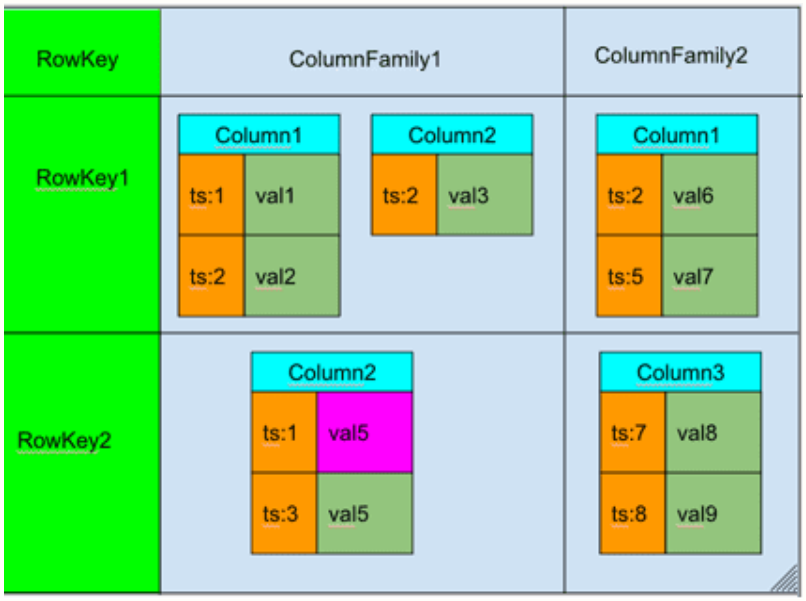

Рисунок 2.7 – Модель даних Apache HBase

Наведемо основні переваги використання HBase:

- Специфічна модель даних, яка не обмежує число стовпців, які можна згрупувати в групи або сімейства. HBase добре підходить для розріджених наборів даних;
- Простежується аналогія з реляційними СУБД в плані індексу первинного ключа - в HBase дані в таблицях впорядковані за строковим ключам при динамічному секціонуванні діапазону рядків;
- Вбудований механізм тимчасових міток, які додаються автоматично, але можуть бути змінені вручну;
- Наявність інструментів розширюваності (REST і інші API-інтерфейси Java і шлюзів) і зовнішніх SQL-рішень (Apache Phoenix, Drill, Hive і Cloudera

Impala), що дозволяють працювати з даними, що зберігаються в HBase , як з реляційними таблицями.

- Висока продуктивність і швидкість роботи
- Висока доступність і відмовостійкість, забезпечуються за допомогою реплікації через центр обробки даних, неподільні і узгоджені операції на рівні рядків, а також автоматичний розподіл навантаження і балансування таблиць за рахунок спеціальних механізмів.
- Здатність до масштабування - Apache HBase розрахована на підтримання високої продуктивності навіть при збільшенні кластера до сотень вузлів для роботи з мільярдами рядків і мільйонами стовпців.[18]

2.8 Вибір технологій для розробки власного додатку та їх переваги

Були розглянуті та проаналізовані популярні на 2021 рік технології для розробки веб-додатків. Усі вони займають почесне місце в рейтингу найбільш популярних технологій серед розробників з усього світу. Всі з розглянутих технологій могли б вирішити поставлену задачу для розробки власного додатку як для серверної так і для клієнтської частини. Були розглянуті та проаналізований функціонал та особливості таких технологій як: Java Spring Framework Python Django, NodeJs - для розробки серверної частини, ReactJS, VueJS, Angular - для клієнтської частини та PostgreSQL, HBase - в якості баз даних.

Говорячи про технології для написання серверної частини всі вони мають потужний набір можливостей, хоч і мають свої певні мінуси порівнюючи їх один з одним. Підсумовуючи, для розробки додатку я обрала фреймворк Java Spring, тому, що він має такі переваги:

- Велика і зрозуміла документація;
- Наявність анотації ;
- Надає широкий спектр готових рішень: Spring MVC, Spring Data та інші;
- Зручний API-інтерфейс;

- Наявність інверсії управління;
- Присутність невеликого досвіду використання Spring;

Щодо клієнтської частини було обрано Angular. Хоча на ринку існує багато альтернатив, декілька з яких було розглянуто у другому розділі. Вибір впав саме на цю технологію через такі переваги даного фреймворку:

- Двостороння зв’язаність даних, що дозволяє моментально відобразити на об’єктах додатку зміни в користувацькому інтерфейсі і навпаки;
- Надає інструменти і шаблони дизайну для побудови проекту;
- Підтримується і має велику спільноту;
- Широкий функціонал;

Вибір бази даних зупинився на класичному об'єктно-реляційному підході PostgreSQL замість нереляційної HBase. Перевагами для мене стали такі особливості даної бази даних:

- Об'єктно-орієнтована СУБД;
- Підтримка багаточисельних типів даних;
- Велике співтовариство - існує досить велика спільнота в якому легко знайти відповіді на свої питання;
- Розмір даних;

Проаналізувавши існуючі технології я дійшла висновку використовувати такі технології як: Java, Spring Framework, Angular, PostgreSQL.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ДОДАТКУ ДЛЯ ПЛАНУВАННЯ ЗАДАЧ ТА УПРАВЛІННЯ ПРОЕКТАМИ

3.1 Формування функціональних вимог

Функціональні вимоги - це перелік вимог які має виконувати система. Перш за все вони залежать від типу розробляємої системи та від потреб користувачів.

Наведемо функціональні вимоги до розробляємого додатка:

- Аутентифікація та авторизація користувача;
- Панель адміністрування проектів;
- Налаштування профілю користувача;
- Панель адміністратора;
- Можливість зміни ролі для користувача;

Роль адміністратора, що включає такі додаткові можливості:

- Панель адміністратора користувачів, що надає змогу додавати нових користувачів та назначати їм ролі;
- Панель адміністрування проектів.

Панель приладів що включає:

- Календар з відображенням власних задач, можливістю внести кількість затраченого часу на задачу за день та відслідковувати дедлайни;
- Список власних задач з основною інформацією про них;
- Пошук задачі по назві;

Панель адміністрування проектів, що включає:

- Відображення задач проекта у вигляді канбан-дощок;
- Створення нового проекту;
- Можливість коментувати проект;
- Можливість видаляти коментарі;
- Створення нової задачі для проекта;
- Редагування інформації про задачу;
- Видалення задачі;
- Можливість назначати відповідальних користувачів за задачу в проекті;
- Видалення проекта;
- Пошук по назві задачі;
- Таблиця для відслідковування витраченого часу на задачу;
- Історія роботи над задачею;
- Список всіх задач.

Налаштування проекту:

- Додавати нові спеціальні поля ;
- Створювати нові категорії;
- Створювати нові канбан-дошки;
- Створювати нові спринти.

Додавати учасників до проекту та надавати їм такі види ролей:

- Manger - адміністратор проекту;
- Contributor - може змінювати стан і спринт до якого відноситься задача;
- Client - може крім прав contributor додавати нові задачі в проект та модифікувати їх.

3.2 Опис архітектури

Для розробки додатку для планування задач та управління проектами було обрано використовувати такий набір технологій: Java, Spring Framework, Angular, PostgreSQL. Метою розробки було

Було обрано клієнт-серверну архітектурний як шаблон для розробки власного додатку. Як було описано у першому розділі, даний підхід є основною ідеєю для побудови веб-додатків. На рис. зображено графічне представлення даного концепту.

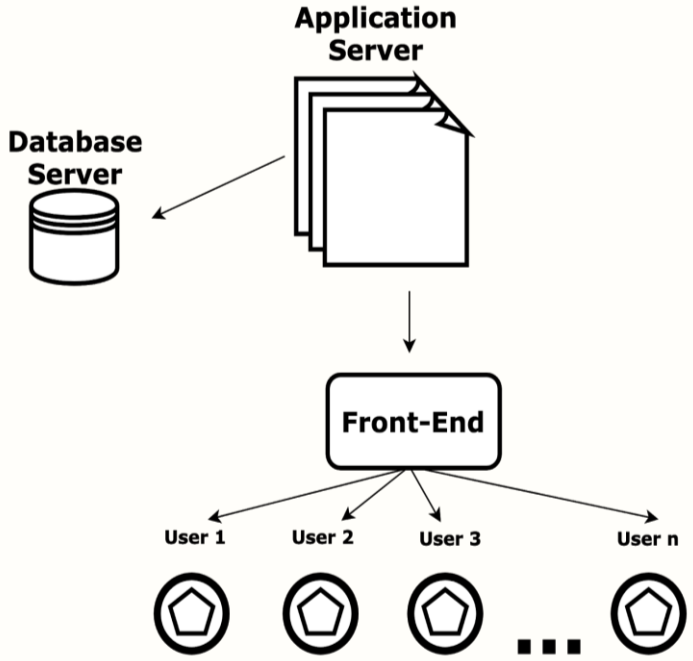

Рисунок 3.1 – Стандартна архітектура клієнт-сервер

Як можна бачити на схемі вище, стандартна архітектура клієнт-сервер складається з трьох частин:

1) Користувацький інтерфейс або рівень клієнта: це частина програмного забезпечення, яка взаємодіє з користувачами. Цей рівень містить екрани входу, меню, екрани даних та звіти, які передають та приймають інформацію до та від користувачів, тобто відповідає за відображення інформації.
2) Сервер додатків: Це сервер, на якому встановлені програмні модулі програми. Він підключається до бази даних і взаємодіє з користувачами. Являється логікою програми, що є рівнем обробки, діє як міст для з'єднання логіки програми з даними на серверах баз даних тощо.

3) Сервер бази даних: Цей сервер містить таблиці, індекси та дані, керовані додатком. Тут виконуються операції пошуку, вставки, видалення, оновлення та інші операції з даними.

На рис. 3.2 зображена взаємодія між частинами клієнт-серверної архітектури.

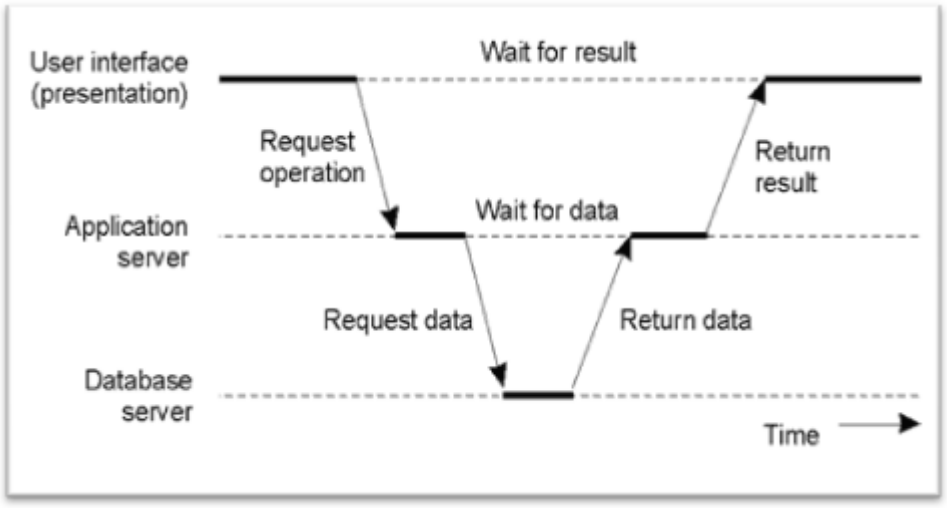

Рисунок 3.2 – Трирівнева клієнт-серверна архітектура

Наведена діаграма зображує трирівневий клієнт-сервер із трьома шарами. розміщені у правильному порядку. Перший рівень, який є інтерфейсом користувача, взаємодіє з користувачем, і тоді він може запитувати певні запити. Потім цей запит буде відправлений на сервер додатків, який знаходиться на другому рівні. На доставлення запиту з інтерфейсу користувача на сервер додатків знадобиться деякий час. Як тільки запит потрапляє на сервер додатків, він обробляє запит, щоб визначити, які дані йому потрібні з сервера бази даних. Після ідентифікації він надішле запит на сервер бази даних із запитом на певний фрагмент даних. Потім сервер бази даних отримував би цю частину даних з бази даних і повертав дані назад на сервер додатків, цей процес зайняв би певний час, як показано стрілками на схемі. Отримавши дані сервер додатків, перетворить дані у версію, сумісну з користувальницьким інтерфейсом. Це робиться тому, що більшість баз даних зберігають дані як вихідні дані. Цей процес перетворення також займе дещо, як показано на схемі темною жирною лінією. Після

завершення перетворення результати повертаються назад до інтерфейсу користувача.

3.3 Структура та аналіз MVC

При проектуванні власного додатку було вирішено використовувати архітектурний шаблон MVC. Розділення даних та бізнес-логіки від візуалізації є головною перевагою даного шаблону.

Model-View-Controller або MVC - це архітектурний шаблон, який розділяє додаток на три основні логічні компоненти Model, View та Controller. Звідси і абревіатура MVC.

Кожен компонент архітектури створений для обробки конкретного аспекту розробки програми. MVC відокремлює бізнес-логіку та рівень презентації один від одного, як це зображено на рис.3.2. Він традиційно використовувався для графічних користувальницьких інтерфейсів. На сьогоднішній день архітектура MVC стала популярною для розробки веб-додатків.

Три важливі компоненти MVC:

1) Модель - Являється центральним компонентом шаблону і відображає поведінку додатку, незалежно від інтерфейсу користувача. Модель відповідає за керування всіма даними та пов'язаними з ними логіку;
2) Вид: Відповідає за відображення даних користувачеві. Все, що бачить користувач, генерується видом;
3) Контролер: Обробник подій, ініційованих користувачем (натискання на кнопку, перехід за посиланням, відправка форми тощо).

Розглянемо детальніше ці компоненти:

View - це візуальне зображення моделі MVC. Цей рівень створює інтерфейс для відображення фактичного результату для користувача. Однак подання нічого не відображатиме саме по собі. Цим займається контролер або

модель, які надають команди поданню про те, що відображати користувачеві. Він також обробляє запити від користувача та інформує контролер. Подання пов'язане з його моделлю і отримує дані, необхідні для презентації. Іноді він також оновлює модель, надсилаючи відповідні повідомлення.

Контролер - це та частина програми, яка обробляє взаємодію користувача. Контролер інтерпретує вхідні дані миші та клавіатури від користувача, перетворюючи їх на команди для model та view, які необхідно змінити. Контролер відправляє команди до model для оновлення її стану та надсилає команди до пов'язаного view для зміни презентації представлення.

Компонент моделі зберігає дані та пов'язану з ними логіку. Він представляє дані, які передаються між компонентами контролера або будь-якою іншою пов'язаною бізнес-логікою. Наприклад, об’єкт Controller отримає інформацію про клієнта з бази даних. Він обробляє дані та надсилає їх назад до бази даних або використовує їх для рендерінгу тих самих даних. [19]

Він відповідає на запит від view, а також відповідає інструкціям контролера щодо оновлення.

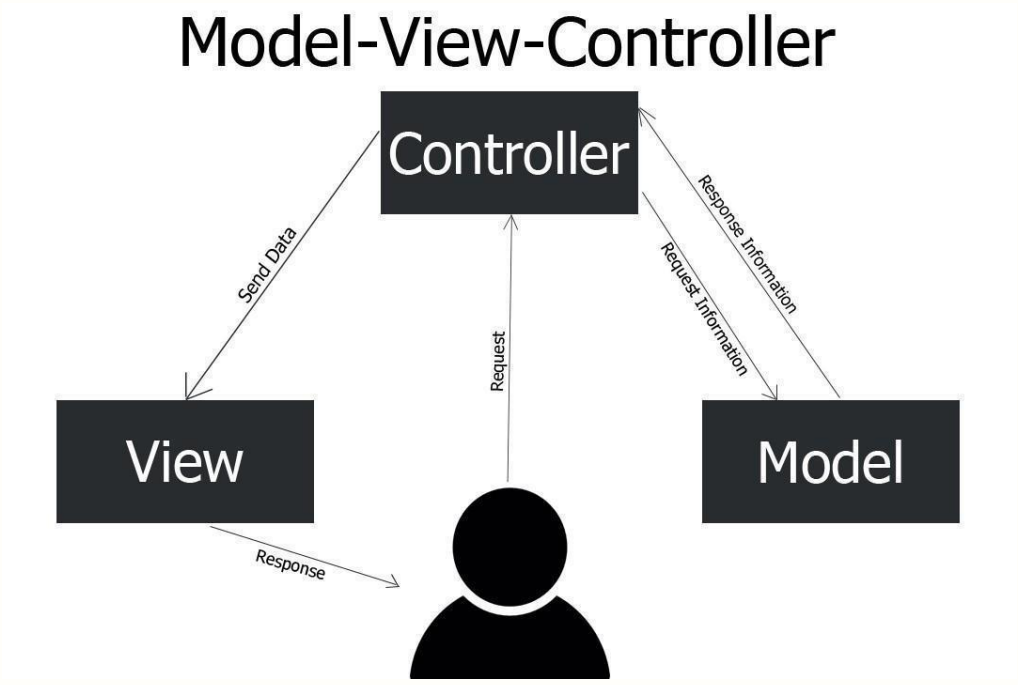

Рисунок 3.2 – Схема взаємодії компонентів в MVC

Розглянемо основні переваги використання MVC:

1. Швидкий процес розробки. MVC підтримує швидкий і паралельний розвиток. Якщо модель MVC використовується для розробки будь-якого конкретного веб-додатку, можливо, наприклад, одночасно працювати над поданням та над контролером для створення бізнес-логіки веб-додатку. Отже, таким чином, програма, розроблена за допомогою моделі MVC, може бути виконана швидше, ніж програми, розроблені з використанням інших шаблонів розробки.

2. Можливість надати декілька подань. У моделі MVC можна створити кілька подань для моделі. Сьогодні зростає попит на нові способи доступу до вашої програми, і для цього розробка MVC, безумовно, є чудовим рішенням. Більше того, у цьому методі дублювання коду дуже обмежене, оскільки воно відокремлює дані та бізнес-логіку від відображення.

3. Модифікація не впливає на всю модель. Для будь-якого веб-додатка користувальницький інтерфейс, як правило, змінюється часто. Очевидно, що розробники часто вносять зміни у свій веб-додаток, наприклад, змінюючи кольори, шрифти тощо. Більше того, додавати новий типів подання дуже просто в шаблоні MVC, оскільки частина модель не залежить від частини видів. Тому будь-які зміни в Моделі не вплинуть на всю архітектуру.

4.Оскільки компоненти мають низьку залежність один від одного, їх легко підтримувати

5.Застосування MVC робить додаток виразнішим та зрозумілішим Таким чином, шаблон дизайну MVC, безумовно, є чудовим підходом до створення програмних додатків. Проекти, які розробляються за допомогою моделі MVC, можуть бути легко розроблені з меншими витратами і за менший час. [20]

3.4 Розробка додатку

Після здійсненого аналізу існуючих технологій для розробки в 2 розділі, було вирішено реалізувати власну систему як веб-додаток. Вона складатиметься з таких частин: клієнт, сервер та база даних. В свою чергу серверна частина

складатиметься з таких компонентів: Controller, Service, Repository, Model та бази даних. Взаємодія компонентів серверної частини зображена на рис. 3.3.

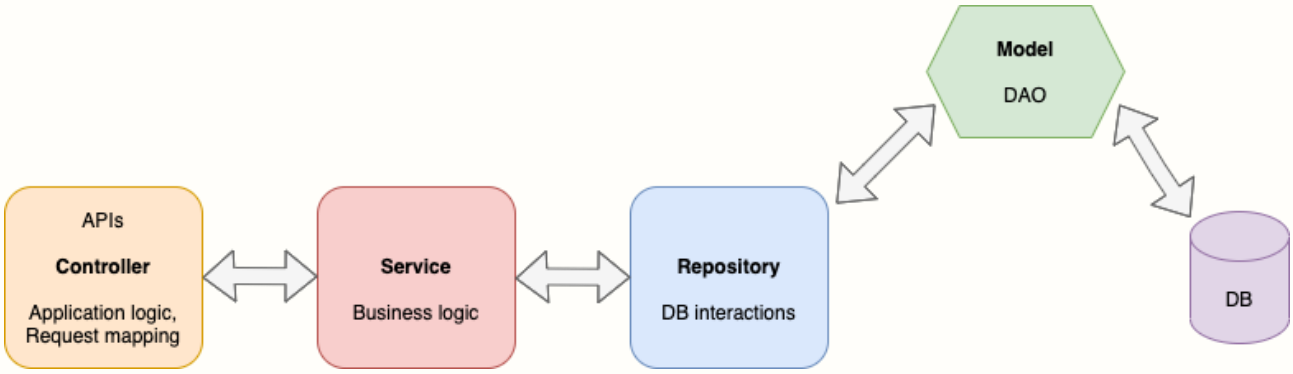

Рисунок 3.2 – Взаємодія компонентів серверної частини

Спочатку розробляється база даних. При її проектуванні були враховані всі функціональні вимоги до розробляємої системи. Таблиці та поля бази даних мають таку ж структуру як і компоненти Model, оскільки Model являється представленням об'єктів з бази даних у вигляді програмного коду. Нижче, на рис. 3.3 наведена структура бази даних.

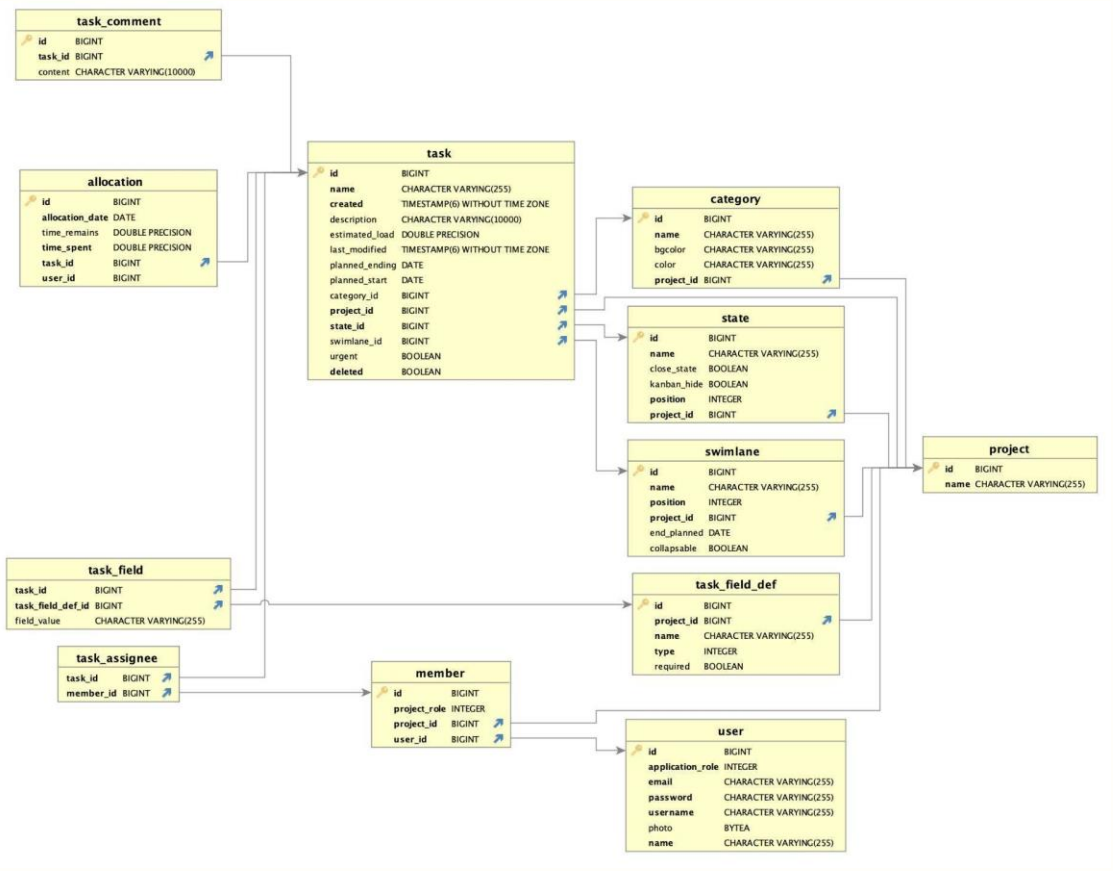

Рисунок 3.3 – Структура бази даних

Перша таблиця має назву task, вона зберігає інформацію про задачу і складається з таких полів:

- id - ідентифікатор задачі, визначає її унікальність
- name - назва задачі
- created - дата створення задачі
- description - опис задачі
- estimated_load - розрахункове навантаження, що описує необхідну кількість годин для виконання задачі
- last_modified - дата останньої коректної зміни в проекті
- planned_ending - орієнтовна дата завершення роботи над задачею
- planned_start - орієнтовна дата початку роботи над задачею
- category_id - ідентифікатор категорії до якої відноситься задача
- project_id - ідентифікатор проекта до якого відноситься задача
- state_id - ідентифікатор стану до якого відноситься задача
- swimlane_id - ідентифікатор спринта до якого відноситься задача

- urgent - ідентифікатор що вказує чи відноситься задача до термінової
  deleted - ідентифікатор що вказує чи видалена задача

Таблиця category, що описує інформацію про категорію до якої відноситься задача, складається з таких полів:

- id - ідентифікатор категорії, визначає її унікальність
- name - назва категорії
- bgcolor - колір яким позначається категорія на канбан-дошці
- project_id -ідентифікатор задачі до якого відноситься категорія

Таблиця state що характеризує стан в якому знаходиться задача, складається з таких полів:

- id - ідентифікатор стану, визначає його унікальність
- name - назва стану
- close_state - ідентифікатор що вказує чи відкрита робота над задачею
- kanban_hide - ідентифікатор що вказує чи відображається задача на канбан-дошці
- position - порядковий номер стану
- project_id - ідентифікатор проекта до якого відноситься стан

Таблиця swimlane описує інформацію про спринт до якого відносяться задачі в проекті. Має такі поля:

- id - ідентифікатор спринта, визначає його унікальність
- name - назва спринта
- position - порядковий номер спринта
- project_id - ідентифікатор проекта до якого відноситься спринт
- end_planned - орієнтовна дата завершення роботи над спринтом
- collapsable - ідентифікатор що визначає чи можна згорнути задачі які відносяться до відповідного спинта

Таблиця task_field_def містить інформацію про додаткове поле в описі задачі, що створене користувачем. Складається з таких полів:

- id - ідентифікатор поля, визначає його унікальність
- project_id - ідентифікатор проекта до якого відноситься поле

- name - назва поля
- type - тип поля
- required - ідентифікатор, що визначає чи є поле обов'язковим

Таблиця user яка містить інформація про користувача та складається з таких полів:

- id -ідентифікатор користувача, визначає його унікальність
- application_role - роль користувача
- email - електронна пошта користувача
- password - особистий пароль для входу в систему
- username - ім'я користувача в системі
- photo - фото
- name - нік-нейм користувача в системі

Таблиця project яка містить інформацію про проект та та складається з таких полів:

- id - ідентифікатор проекта, визначає його унікальність
- name - назва проекта

Таблиця task_comment що містить інформацію про коментар до задачі та складається з таких полів:

- id - ідентифікатор коментаря, визначає його унікальність
- task_id - ідентифікатор задачі до якої відноситься коментар
- content - строка що містить інформацію про автора коментаря та дату його написання

Таблиця allocation відповідає за відслідковування прогресу роботи над задачею. Містить такі поля:

- id - ідентифікатор таблиці, визначає його унікальність
- allocation_date - дата внесення змін
- time_remains - кількість годин що залишається витратити на виконання задачі
- time_spent - кількість годин що було витрачено на виконання задачі
- task_id - ідентифікатор задачі до якої відноситься інформація про прогрес

- user_id - ідентифікатор користувача до якого відноситься інформація про прогрес над виконуваною задачею

Таблиця task_field яка відповідає за співвідношення додаткового поля в інформації про задачу з конкретною задачею. Містить такі поля:

- task_id - ідентифікатор задачі
- task_field_def_id - ідентифікатор додаткового поля
- field_value - інформація, що записана в полі

Таблиця member містить інформацію про учасника проекта відповідального за певну задачу. Складається з таких полів:

- id - ідентифікатор учасника, визначає його унікальність
- project_role - роль учасника в проекті
- project_id - ідентифікатор проекта, визначає його унікальність
- user_id - ідентифікатор користувача, визначає його унікальність

Таблиця task_assignee що відповідає співвідношенню задачі до користувача відповідального за неї. Містить такі поля:

- task_id - ідентифікатор задачі
- member_id - ідентифікатор відповідального за задачу користувача

Далі наведена таблиця 1.1 відношень між таблицями у базі даних.

Таблиця 1.1 - Відношень між таблицями у базі даних

Зв’язок	Таблиця 1	Таблиця 2	Опис
Many-to-One	task	category	Багато задач може відноситися до однієї категорії.
Many-to-One	task	state	Багато задач може відноситися до одного стану.
Many-to-One	task	swimlane	Багато задач може відноситися до одного спринта.

Many-to-One	task	project	Один проект може містити багато задач.
Many-to-One	task_comment	task	Одна задача може мати багато коментарів.
Many-to-One	allocation	task	Одна задача може мати багато записів про прогрес над нею.
Many-to-One	task_field	task	Одна задача може мати багато додаткових полів.
Many-to-One	task_field	task_field_def	Одне додаткове поле може бути відображене у багатьох задачах.
Many-to-Many	task_assignee	task	Багато користувачів можуть бути відповідальними за одну задачу.
Many-to-Many	task_assignee	member	Багато користувачів можуть бути призначеними відповідальним за багато задач.
Many-to-One	member	user	Один користувач може виконувати роль відповідального за багато задач.
Many-to-One	member	project	У одному проекті можуть бути багато відповідальних за задачі.
Many-to-One	category	project	У одному проекті задачі можуть бути віднесені до багатьох категорій.
Many-to-One	state	project	У одному проекті задачі можуть бути віднесені до багатьох станів.
Many-to-One	swimlane	project	У одному проекті можуть бути багато

			спринтів.
Many-to-One	task_field_def	project	У одному проекті може бути визначено багато додаткових полів.

Після розробки бази даних були реалізовані необхідні компоненти для функціонування серверної частини, а саме - Controllers, Services, Repositories.

Для доступу до даних та спрощення роботи з ними була використана технологія Spring Data JPA. Головними компонентами для взаємодії з даними в Spring Data є репозиторії. У своєму проекті були створені такі репозиторії:

CategoryRepository - надає базовий набір методів для доступу до даних що логічно пов'язані з категоріями.

CommentRepository - надає базовий набір методів для доступу до даних що логічно пов'язані з коментарями та можливість виконувати операції над ними.

ProjMemberRepository - надає можливість роботи з даними які пов'язані з учасниками проекту та можливість виконувати операції над ними.

ProjectRepository - даний репозиторій надає доступ до даних, що пов'язані з проектом та можливість виконувати операції над ними.

StateRepository - забезпечує доступ до даних які пов'язані зі станом та надає можливість виконувати операції над даними такі як сортування.

SwimlRepository - забезпечує доступ до даних які пов'язані зі спринтом та надає можливість виконувати операції над даними такі як сортування.

TaskRepository- репозиторій надає доступ до даних пов'язаних з інфомацією про задачі і проектах та можливість виконувати операції над ними.

TaskfieldRepository - надає доступ до даних які логічно пов'язані з додатковими полями проекта та можливість виконувати операції над даними такі як сортування.

HistoryRepository - створений для можливості роботи з даними, що тісно пов'язані з історією змін в задачах.

UserRepository - репозиторій, що надає доступ до даних про користувача додатку та можливість виконувати операції над ними .

Дані репозиторії були наслідуваними від таких репозиторіїв як: crudRepository, який надає базовий набір методів для доступу до даних та PagingAndSortingRepository - універсальний інтерфейс, який розширює CrudRepository і додає підтримку сортування.

Для обробки http запитів зі сторони клієнта і взаємодії View з Model були створено контроллери.

Кожен зі створених сервісів відповідає за роботу бізнес-логіки додатку. Наведемо деякі з них:

UserDetailService - даний сервіс реалізує виконання бізнес-логіки пов’язаної з користувачем: авторизація, автентифікації редагування та перегляд інформації про користувача тощо.

TaskService - надає можливість користувачу створювати проекти та задачі для них, вести роботу над ними та відслідковувати прогрес.

ProjService - реалізовує бізнес-логіку для роботи з проектами.

3.5 Результат розробки

В результаті розроблений додаток відповідає всім функціональним вимогам та надає користувачам звучний та приємний інтерфейс. Далі ми детальніше розглянемо розроблену систему продемонструючи її інтерфейс та інструменти які вона надає.

Перше вікно яке бачить користувач - вікно авторизації, що зображене на рис.3.4

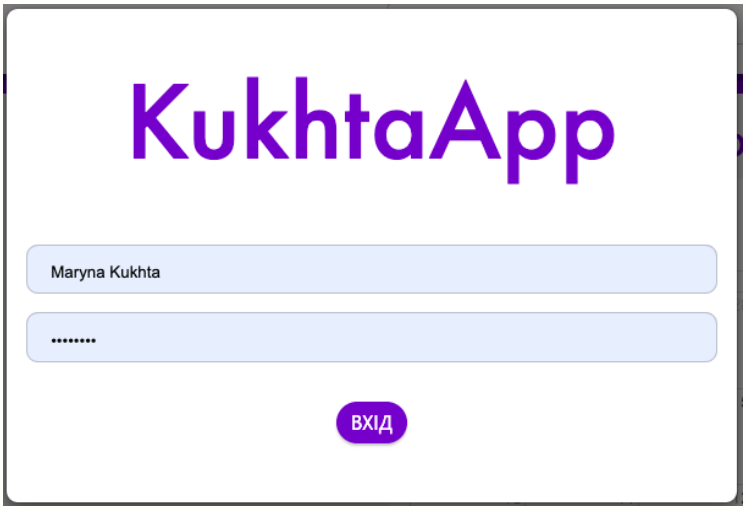

Рисунок 3.4 – Вікно авторизації користувача

Після авторизації користувач потрапляє на головну сторінку додатка. На цій сторінці користувач може побачити список поточних задач які йому необхідно виконати. У списку відображена назва задачі, її статус, очікувана дата початку, очікувана дата завершення роботи над задачею, спринт до якого вона відноситься, категорія, розрахункове навантаження та скільки годин вже витрачено на виконання задачі.

Праворуч від приску задач відображений календар поточного місяця з відміченими на ньому задачами, які присвоєні користувачеві. Головна сторінка зображена на рис. 3.5.

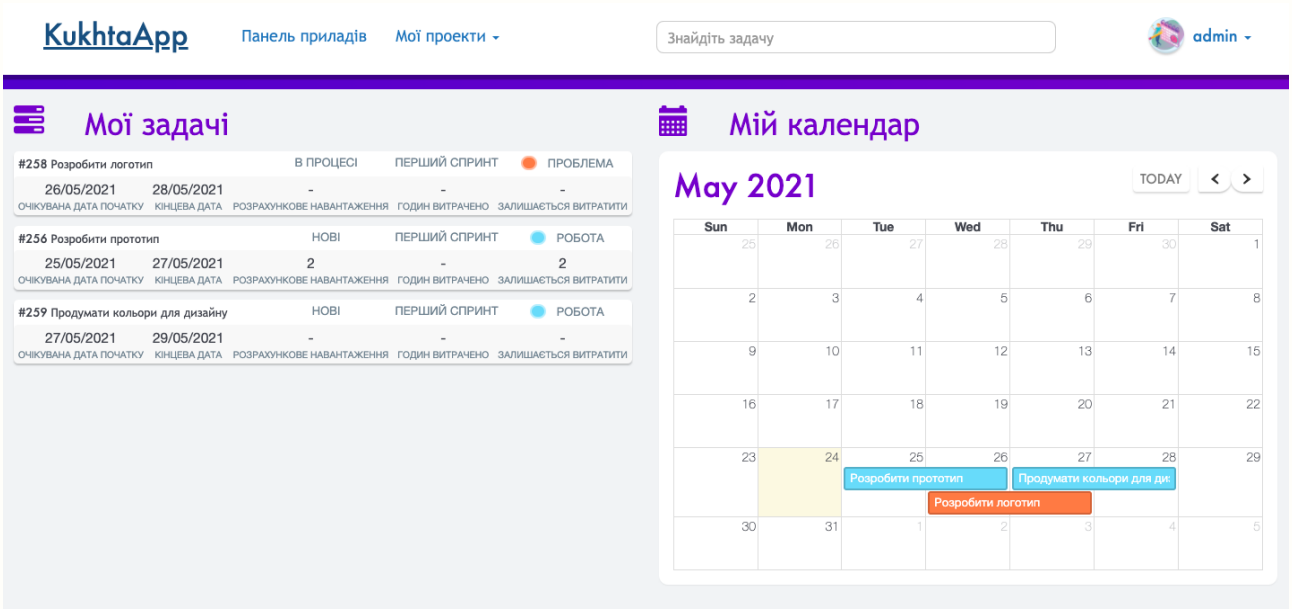

Рисунок 3.5 – Головна сторінка додатку

Натиснувши на день тижня у календарі користувач може внести показники своєї продуктивності над задачею - а саме записати, скільки часу він витратив в поточний день на роботу над нею.

Після того, як користувач ввів інформацію щодо своєї робити на головному екрані відобразиться оновлений список “Мої задачі” та користувач побачить оновлену статистику по певній задачі. В даному випадку оновилися поля “годин витрачено” та “залишається витратити” для задачі “Розробити прототип.”. Це можна побачити на рис. 3.6

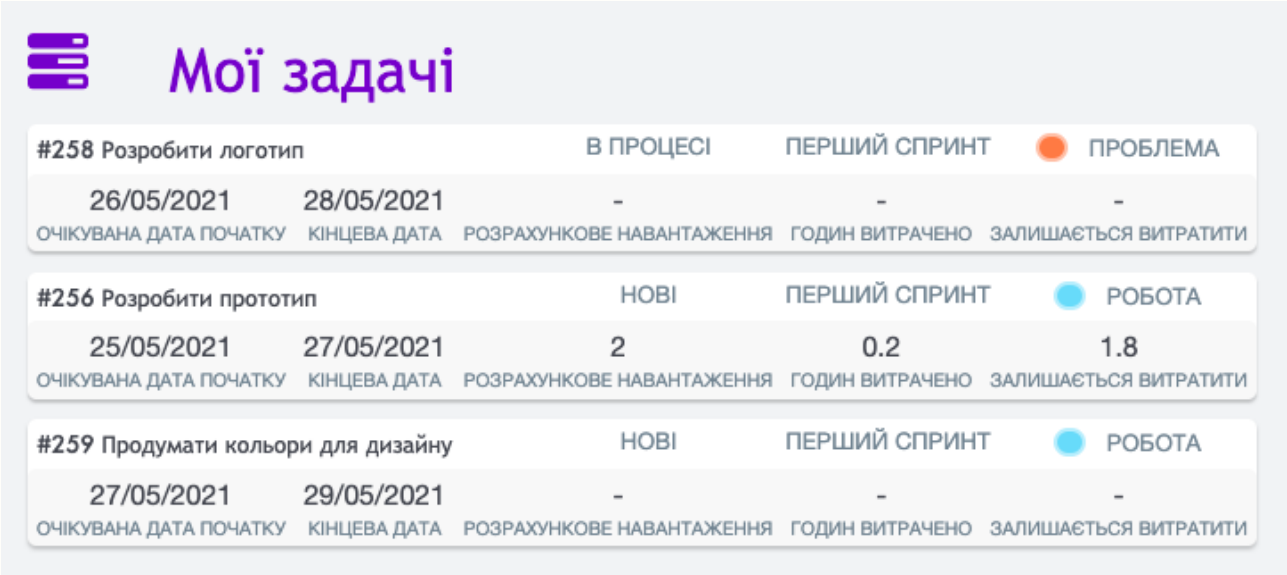

Рисунок 3.6 – Фрагмент списку “Мої задачі”

Натиснувши на будь-яку задачу із списку можна перейти в режим редагування задачі. Дана сторінка зображена на рис. В цьому режимі користувач може редагувати задачу, а саме назву, поточний стан виконання, спринт до якого відноситься задача, відмітити її як термінову, змінити орієнтовані дати початку і кінця, розрахункове навантаження, опис. Є можливість додати команду із одного чи декількох користувачів додавши їх до списку призначених до роботи над задачею.

Після редагування задачі користувачі можуть переглядати історію змін. Так наприклад, зміна стану задачі відобразиться в історії, де буде вказана сама зміна, категорія зміни та вказано хто саме і коли її зробив. Даний функціонал зображений на рис. 3.7.

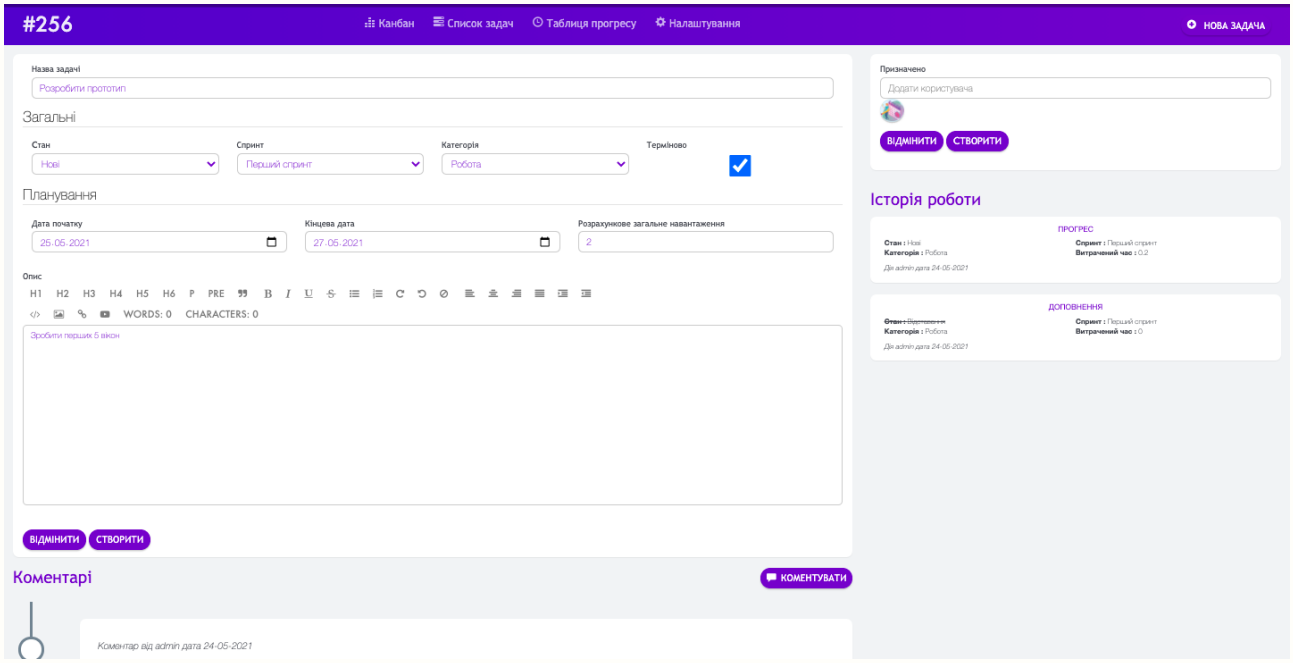

Рисунок 3.7 – Сторінка редагування задачі

Також, до задачі можна додати коментар, натиснувши на кнопку “Коментувати” - рис. 3.8.

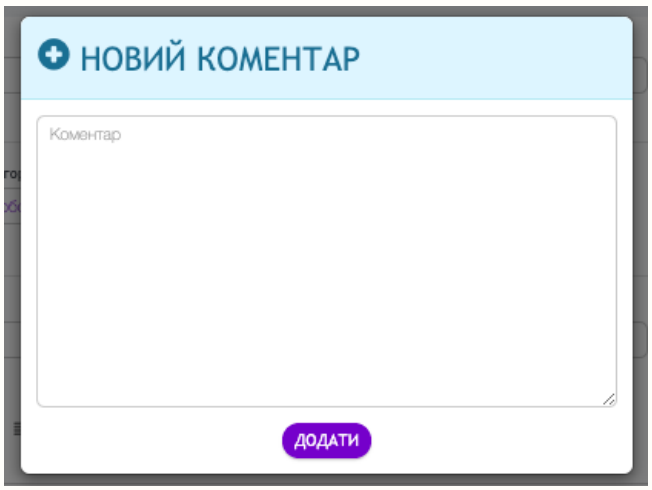

Рисунок 3.8 – Вікно для введення нового коментаря

Результат роботи даної функції можна побачити на рис. 3.9 При відображені коментаря видно хто його автор та дату написання, при бажанні автор коментаря може його видалити.

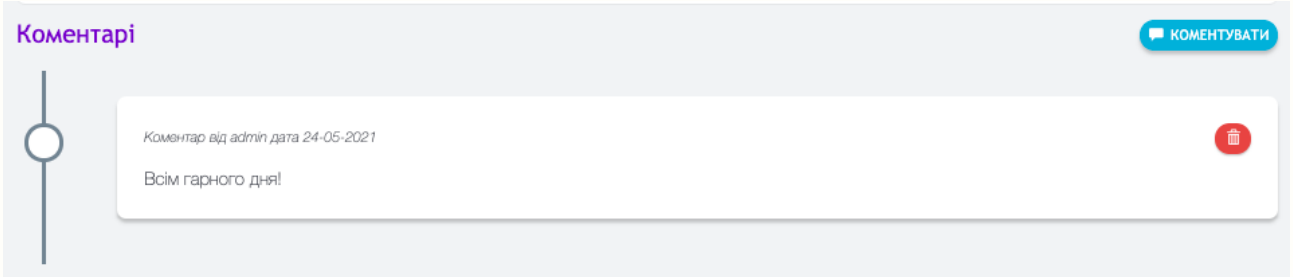

Рисунок 3.9 – Результат роботи введення нового коментаря

Через головний екран, натиснувши на кнопку “Мої проекти” та перейшовши до будь-якого з них користувач потрапляє до сторінки проекту. На цій сторінці відображені канбан-дошки з задачами до проекту - рис.3.10. Кожна дошка має свою назву.

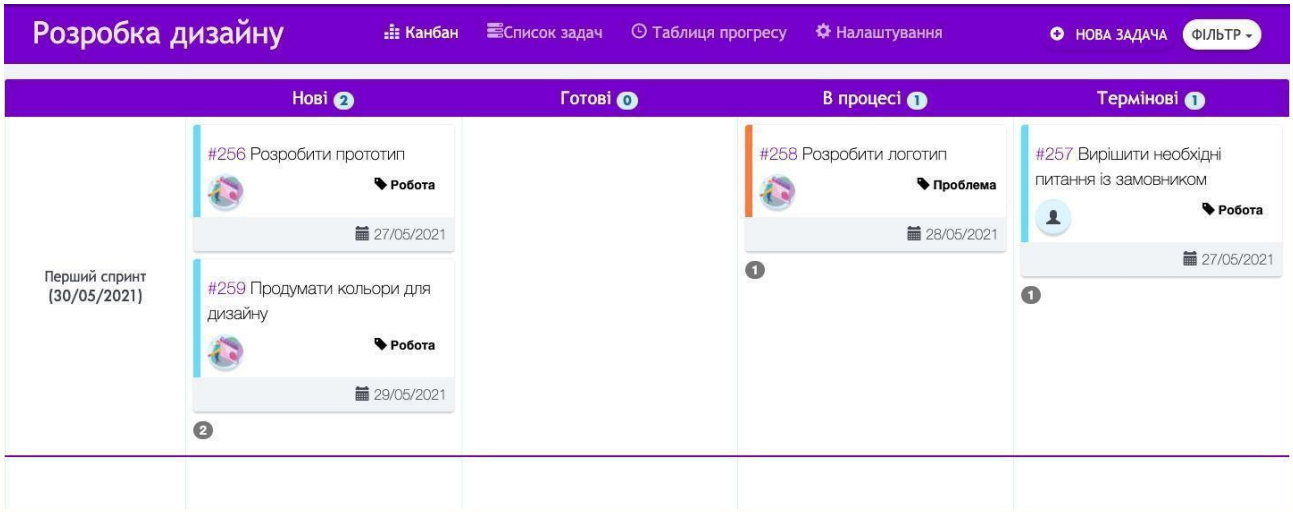

Рисунок 3.10 – Канбан-дошки проекту

На кожній задачі відображена кінцева дата її виконання, назва, категорія до якої вона відноситься та зображення користувачів які відповідальні за неї. Натиснувши на задачу користувач потрапить на сторінку її редагування, функціонал якої був описаний раніше.

Натиснувши на кнопку “Фільтр” задачі можна фільтрувати по таким показникам як: призначено мені та термінові.

Натиснувши на кнопку “Список задач” користувач потрапляє на сторінку де у вигляді карток відображені всі задачі поточного проекту. Дана сторінка зображена на рис. 3.11.

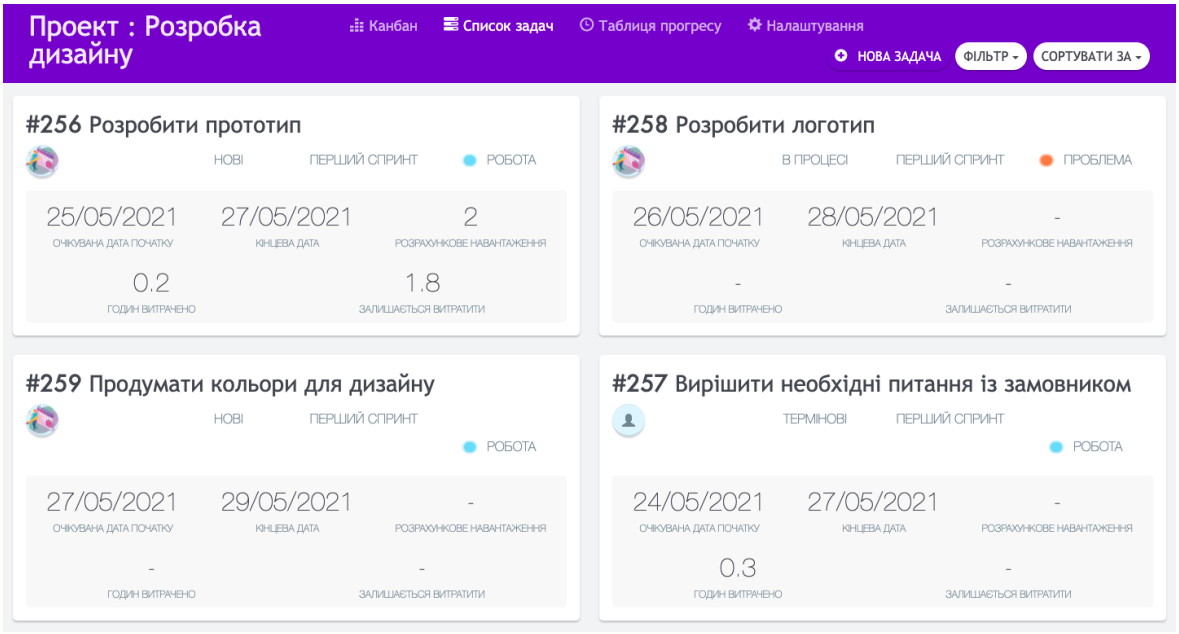

Рисунок 3.11 – Сторінка зі списком задач поточного проекту

На кожній картці відображається інформація про задачу - очікувана дата початку, кінцева дата, розрахункове навантаження, скільки роботи в годинах залишається зробити, скільки годин витрачено на задачу, до якої канбан-дошки та спринта вона відноситься, категорія та відповідальні користувачі за неї. Також на цій сторінці можна видалити або редагувати задачу.

Натиснувши на кнопку “Фільтр” задачі можна відфільтрувати за будь-якою наведеною на картці інформацією.

Натиснувши на кнопку ”Сортувати за” задачі можна відсортувати за будь-яким із наведеною на картці інформацією. Даний функціонал зображений на рис.3.12.

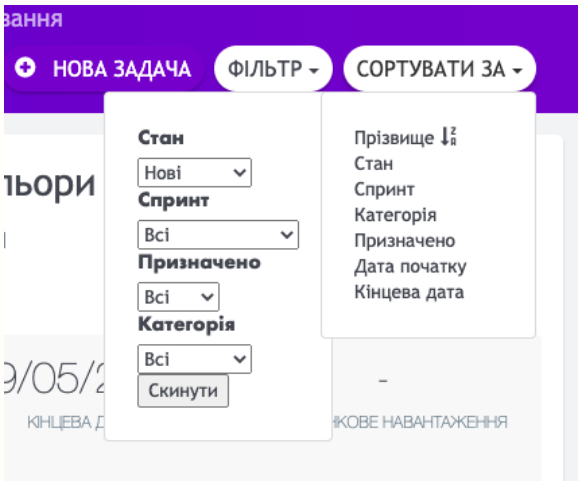

Рисунок 3.12 – Демонстрація можливості фільтрації та сортування

Якщо натиснути на кнопку “Таблиця прогресу” користувач переходить на сторінку що містить інформацію про прогрес кожного із учасників проекту. У таблиці відображається ім’я користувача та скільки часу він витратив на певну задачу у конкретний день. Дана сторінка зображена на рис.3.13.

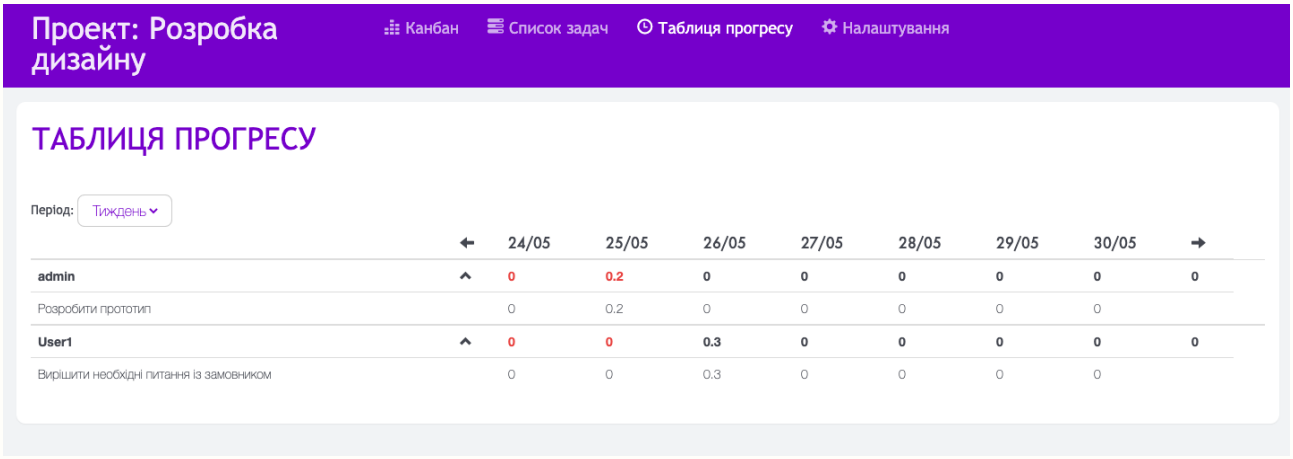

Рисунок 3.13 – Таблиця прогресу до проекту

Перейшовши на сторінку “Налаштування” можна додати спеціально поля до проекту як зображено на рис.3.14 . Тоді при створенні нового проекту або редагуванні існуючого з’явиться новостворене поле.

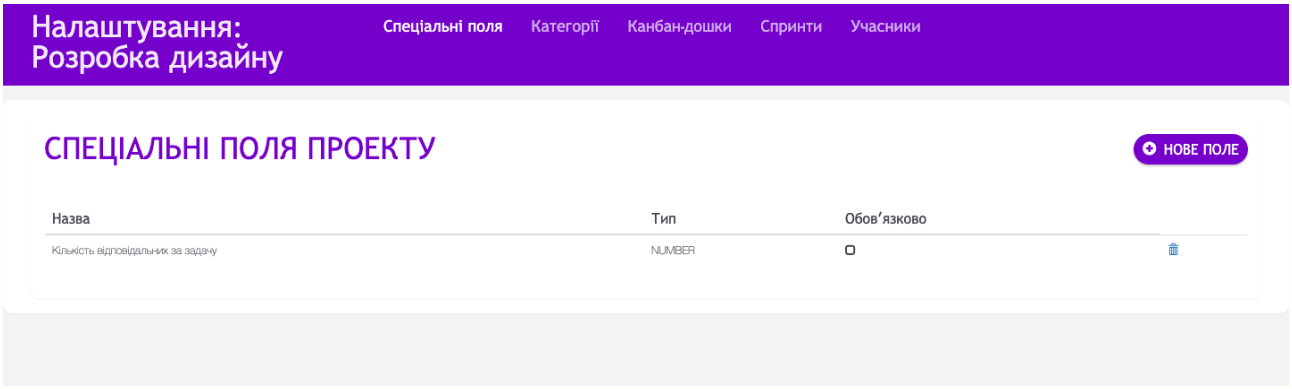

Рисунок 3.14 – Демонстрація можливості додати нові поля до проекту

Щоб його створити потрібно натиснути на кнопку ”Нове поле”, після цього з'явиться діалогове вікно яке пропонує ввести назву поля, його тип - число, дата або текст, та чекбокс щоб відмітити його обов'язковим для заповнення, така можливість зображена на рис. 3.15.

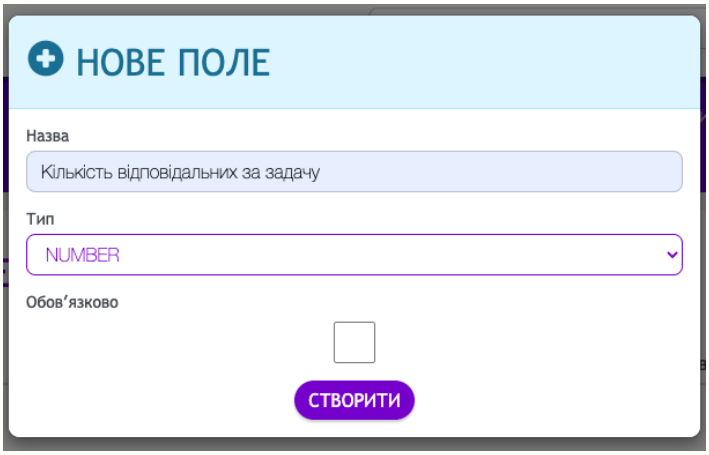

Рисунок 3.15 – Демонстрація створення нового поля

Перейшовши на сторінку “Категорії” користувач може створити додаткові категорії для завдань натиснувши на кнопку “Нова категорія”. Для її створення у діалоговому вікні потрібно ввести її назву та задати колір як це можна побачити на рис. 3.16.

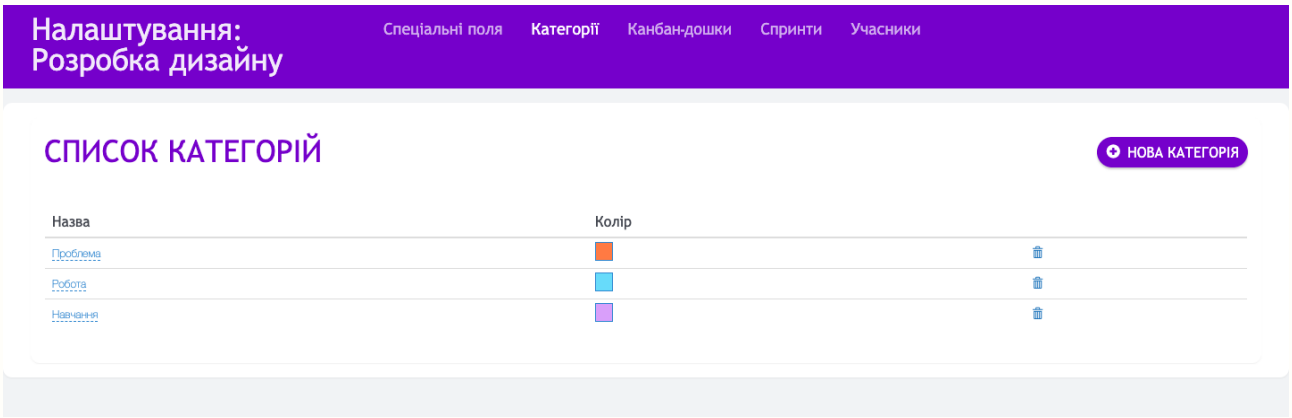

Рисунок 3.16 – Демонстрація можливості створення нової категорії для задачі

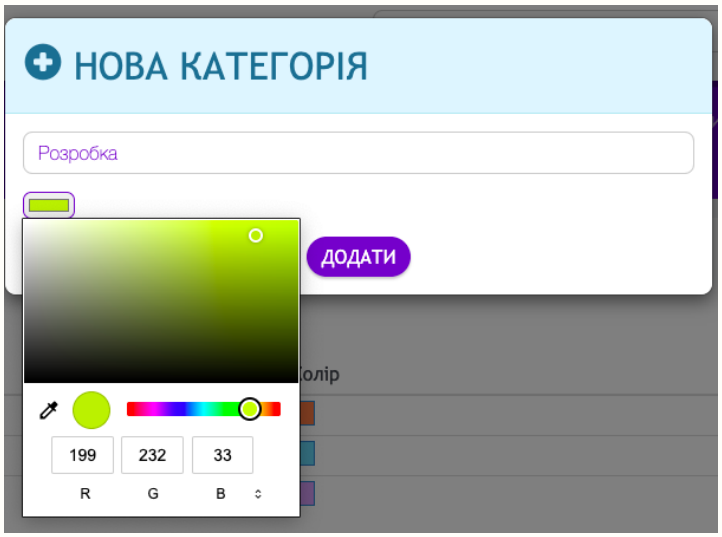

Рисунок 3.17 – Демонстрація створення нової категорії

На сторінці “Канбан-дошки” відображається список можливих назв для канбан-дощок та їх кількість у поточному проекті рис. 3.18.

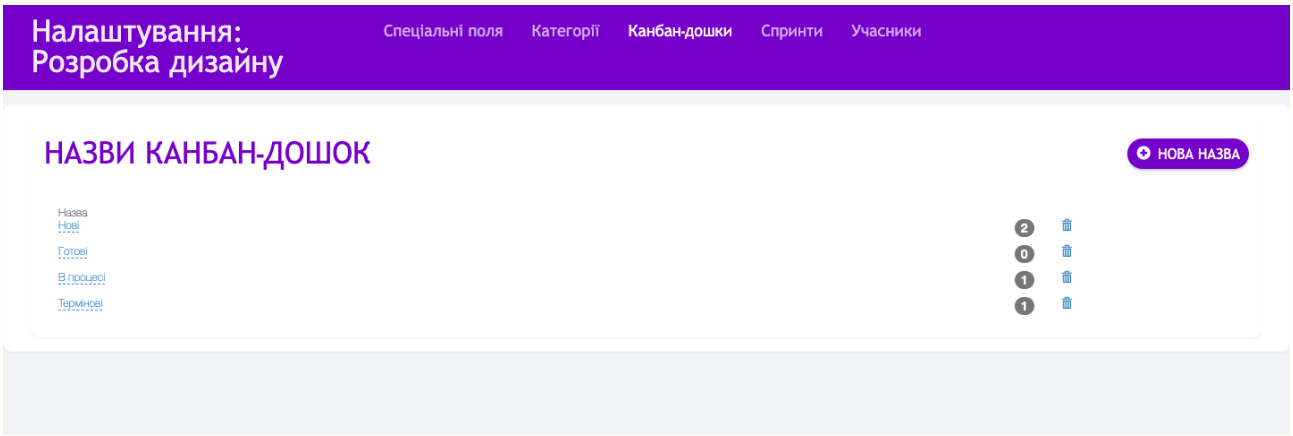

Рисунок 3.18 – Демонстрація можливості створення нової канбан-дошки

Можна додати нову натиснувши на кнопку “Нова назва” та ввівши необхідні дані у діалоговому вікні як показано на рис. 3.19.

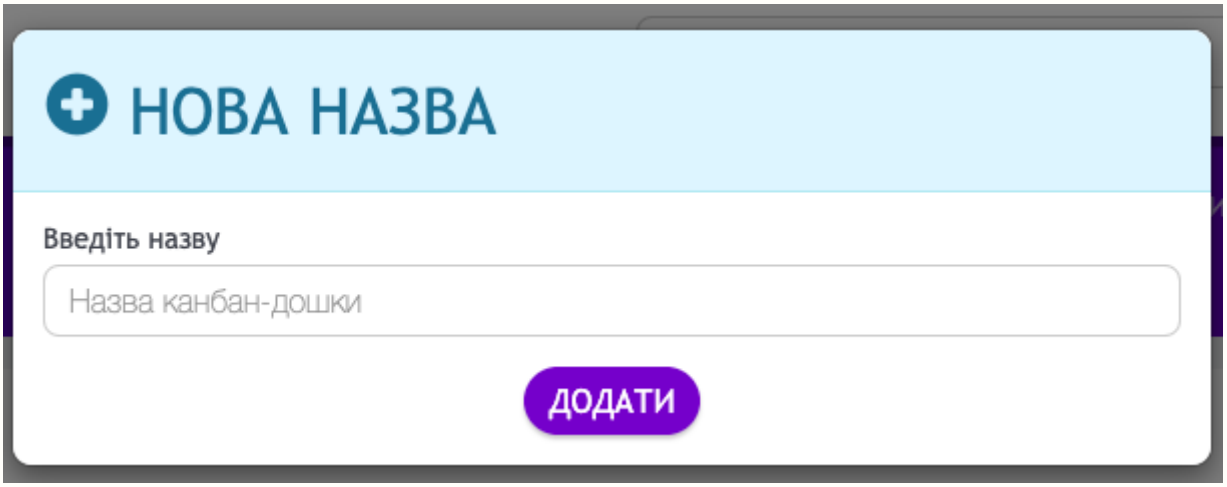

Рисунок 3.19 – Демонстрація створення нової канбан-дошки

На сторінці “Спринти” відображається список спринтів, які доступні у поточному проекті, їх дати завершення та кількість, рис.3.20.

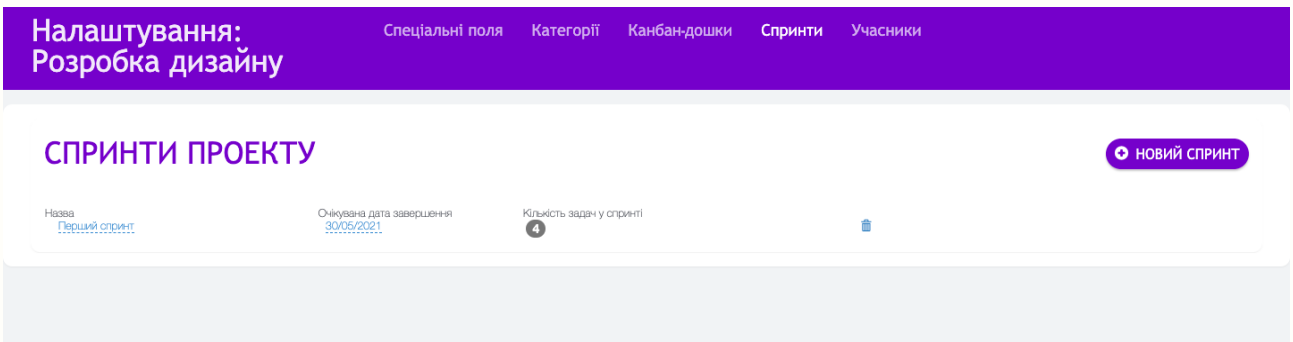

Рисунок 3.20 – Демонстрація можливості створення нового спринта

Натиснувши на кнопку “Новий спринт” можна додати новий спринт ввівши необхідну інформацію у діалоговому вікні як показано на рис.3.21. Надалі він відображатиметься на сторінці з канбан-дошками до проекту.

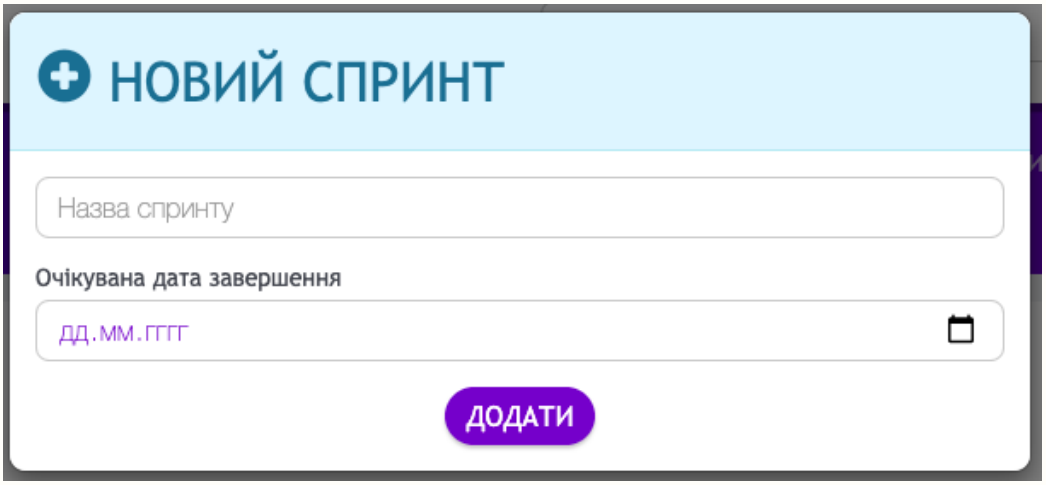

Рисунок 3.21 – Демонстрація створення нового спринта

До кожного з проектів можна додати учасників та роль у проекті, щоб надати їм доступ до перегляду задач проекту та в подальшому мати змогу назначати їх відповідальними за задачі. Даний функціонал зображений на рис.3.22.

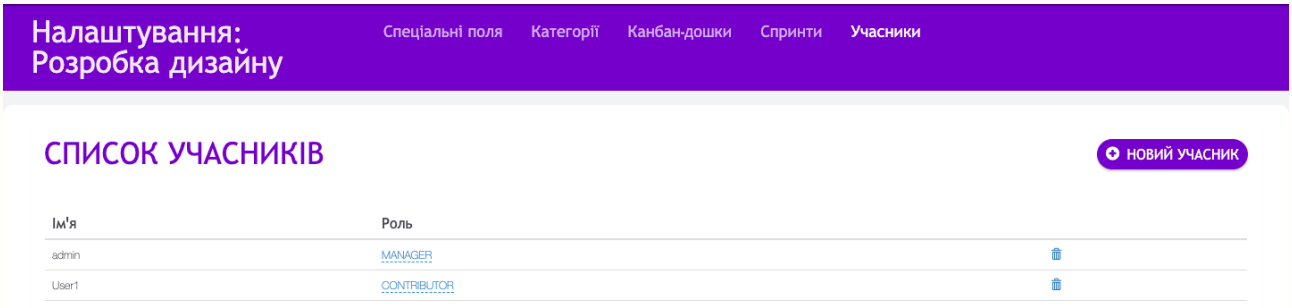

Рисунок 3.22 – Демонстрація можливості додати учасників до проекту

Натиснувши аватар авторизованого користувача і вибравши у меню сторінку “Профіль” є можливість змінити налаштування профілю. А саме пароль, електронну адресу та зображення профілю як зображено на рис.3.23 .

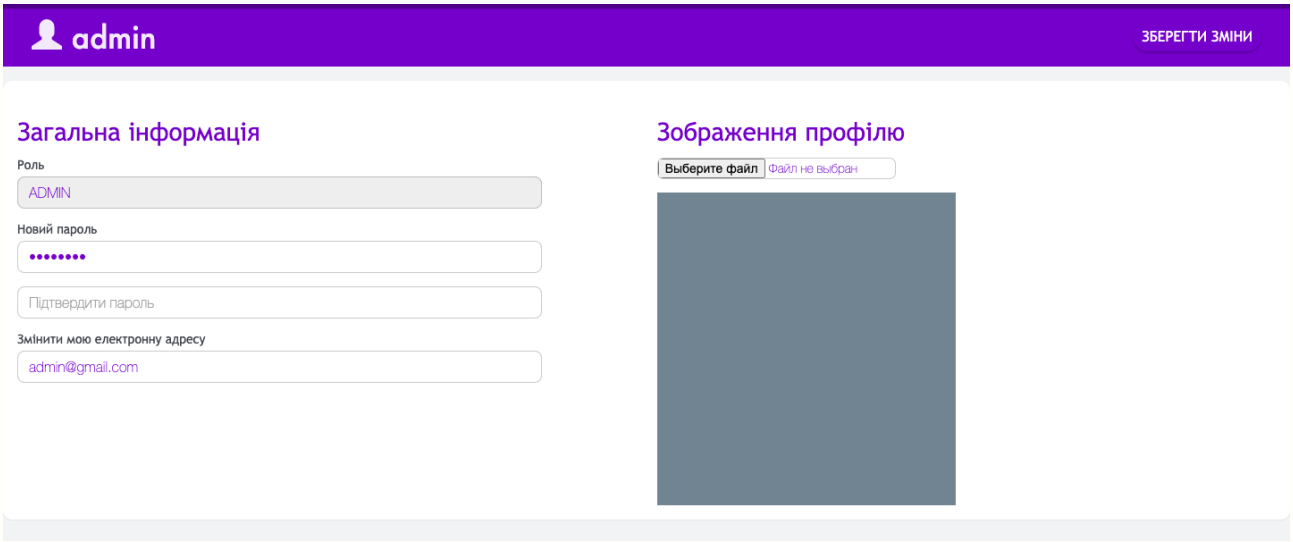

Рисунок 3.23 – Налаштування профілю користувача

Перейшовши через меню на сторінку “Адміністратор користувачів”, адміністратор може додати нового користувача призначивши йому роль користувач чи адміністратор. Я це можна побачити на рис.3.24.

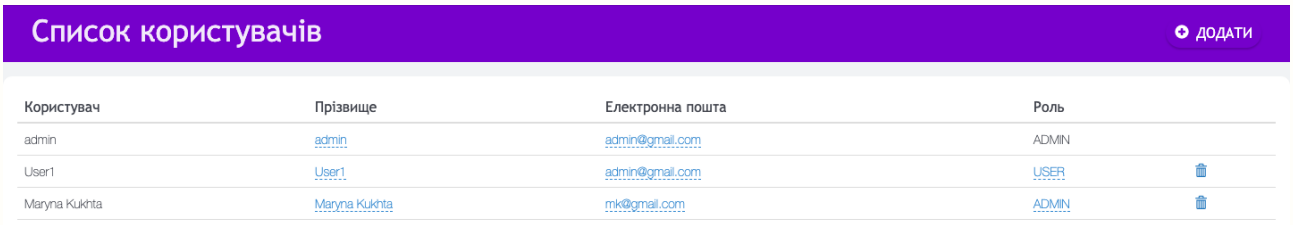

Рисунок 3.24 – Адміністратор користувачів

Натиснувши на кнопку “Додати” у діалогове вікно необхідно ввести інформацію про нового користувача. Дане вікно зображене на рис. 3.25

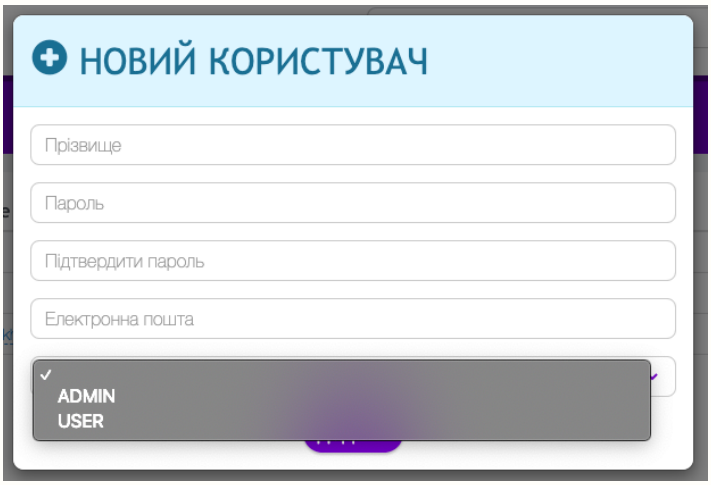

Рисунок 3.25 – Демонстрація реєстрації нового користувача

Перейшовши на сторінку “Адміністрування проектів” користувач попляє на сторінку де у вигляді карток відображені всі його проекти. На цій сторінці можна перейти в режим редагування проекта, видалити його або створити новій натиснувши на кнопку “Додати”, як зображено на рис. 3.26.

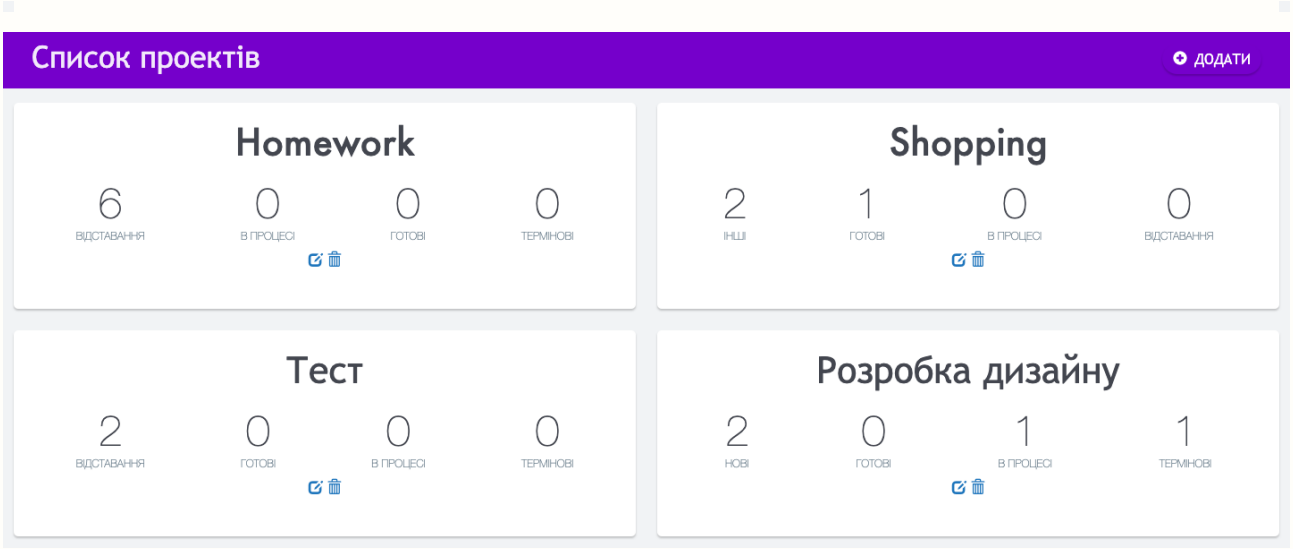

Рисунок 3.26 – Список всіх проектів користувача

3.6 Основні переваги в порівнянні з існуючими аналогами

Перед проектування власного додатку було проаналізовано сучасні на ринку аналоги, що описані в розділі 1. Усі розглянуті додатки мають певні переваги та недоліки у порівнянні з власною розробленою системою. Нижче наведена таблиця з порівнянням.

Таблиця 3.3 - порівнянні власного додатку з існуючими аналогами

Основна функціональність	Аналоги присутні на ринку		Власний додаток
	Jira	Trello	KukhtaApp
Створення проектів та задач	+	+	+
Push-up сповіщення	+	+	-

Відстеження статусів проектів	+	+	+
Можливість швидкого освоєння системи	-	+	+
Типізація системи організації задач	+	-	+
Можливість використання для роботи над великими проектами	+	+	+
Таблиця для відслідковування прогресу	+	-	+
Додавання коментарів учасниками команди до задач проекту	+	+	+
Зручний інтуїтивний інтерфейс	-	+-	+
Можливість працювати в додатку як над великими так і малими проектами безкоштовно	-	-	+
Має вбудований безкоштовний інтерактивний календар	-	-	+
Необмежена безкоштовна кількість користувачів у проекті	-	-	+
Простота налаштування проекту	-	+	+

На відміну від більшості аналогів, враховуючи Jira, власний додаток має більш просте налаштування і не потребує особливих навичок для використання. Так, наприклад у одного з найвідоміших аналогів Jira досить складні налаштування. Щоб усе працювало як треба, необхідно витратити чимало часу та сил. На офіційному сайті є спеціальний курс відеолекцій з тестами для підготовки майбутніх користувачів. Власний додаток не перевантажений функціоналом, що забезпечує його користувачам легке використання - нічого зайвого. На відміну від Trello у створеному додатку є можливості видаляти картки, а не лише архівувати. Також він має достатньо функціоналу для повноцінної роботи як над малими так і над великими проектами. Додатковою перевагою власного додатку є вбудований інтерактивний календар, якого немає в аналогах. Створений додаток надає всі свої інструменти безкоштовно, що також є великою перевагою на відміну від існуючих систем. А легкий та інтуїтивно зрозумілий інтерфейс буде до вподоби всім користувачам.

ВИСНОВКИ

За останні роки помітно зростає попит на інструменти для менеджменту задач та проектів. Більшість користувачів та компаній переходить від звичайного планування на папері до сучасних додатків, які спрощують проблему планування, покращують та прискорюють процес роботи.

В результаті виконання дипломної роботи були отримані такі результати:

1) Проаналізована проблема планування задач за управління проектами. В результаті аналізу дійшли висновку, що більшість сфер, що впровадити в своє життя технології для вирішення цих проблем залишилися задоволеними від результатів використання. Тому розробка даної системи є актуальною;
2) Були проаналізовані існуючі на сьогоднішній день аналоги та в результаті аналізу сформовані функціональні вимоги до власного додатку;
3) Розглянуто архітектуру та інструменти розробки веб-додатків, вивчено та використано необхідний стек технологій для реалізації. Для розробки були використані інструменти що відповідають сучасному рівню технічних знань і технологій;
4) Проведена розробка всіх необхідних компонентів системи для клієнтської та серверної частини. Результатом розробки являється веб-додаток для планування задач та управління проектами, що відповідає всім передбаченим до нього вимогам, має широкий набір інструментів та відповідає вимогам до сучасних веб-застосунків;
5) Були описані переваги створеного додатку перед існуючими на ринку аналогами.

Без сумніву створений у додатку здатний вирішувати проблему планування задач по управління проектами та може використовуватися у всіх сферах де ця проблема є актуальною.

Є доцільним в майбутньому розширити набір інструментів додатку, розробивши додатковий функціонал для охоплення більшої кількості зацікавлених у використанні користувачів

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1.Atlassian Jira | Программное обеспечение для отслеживания задач и проектов Планируйте и отслеживайте agile-проекты и разработку [Електронний ресурс] – Режим доступу: https://www.atlassian.com/ru/software/jira.

2. JIRA: ПОДРОБНОЕ РУКОВОДСТВО ДЛЯ НОВИЧКОВ — BYTEX BLOG [Електронний ресурс] – Режим доступу: https://bytextest.ru/2018/08/31/jira-dlya-novichkov/.

3. Что такое Trello и как им пользоваться | Медиа Нетологии: образовательная платформа [Електронний ресурс] – Режим доступу: https://netology.ru/blog/trello.

4. Any.DO — обзор сервиса [Електронний ресурс] – Режим доступу: https://startpack.ru/application/any-do-list.

5. Any.do - 26 секретных фактов, обзор, характеристики, отзывы. [Електронний ресурс] – Режим доступу: https://rankquality.com/any-do/.

6. Хорстманн, Кей С. Java. Библиотека профессионала, том 1. Основы. 11-е изд.: Пер. с англ. СПб.: ООО "Диалектика", 2019. - 864 с.

7. Spring Framework Project [Електронний ресурс] – Режим доступу: https://spring.io/projects/spring-framework.

8. Core Technologies [Електронний ресурс] – Режим доступу: https://docs.spring.io/spring-framework/docs/current/reference/html/core.html.

9. Уоллс К. Spring в действии. – М.: ДМК Пресс, 2013. – 752 с.

10.Django overview [Електронний ресурс] – Режим доступу: https://www.djangoproject.com/start/overview/.

11.Почему Django — лучший фреймворк для разработки сайтов [Електронний ресурс] – Режим доступу: https://ru.hexlet.io/blog/posts/pochemu-django-luchshiy-freymvork-dlya-razrabotki-saytov.

12. React в действии. — СПб.: Питер, 2019. — 368 с.: ил. — (Серия «Для профессио- налов».

13. ReactDOM – React [Електронний ресурс] – Режим доступу: https://uk.reactjs.org/docs/react-dom.html.

14. The Pros and Cons of Vue.js [Електронний ресурс] – Режим доступу: https://www.altexsoft.com/blog/engineering/pros-and-cons-of-vue-js/.

15. Vue.js в действии. - Хэнчеп Эрик, Листуон Бенджамин СПб.: Питер, 2019.

16. Introduction to the Angular Docs [Електронний ресурс] – Режим доступу: https://angular.io/docs.

17. PostgreSQL 13.3, 12.7, 11.12, 10.17, and 9.6.22 Released! [Електронний ресурс] – Режим доступу: https://www.postgresql.org/about/news/postgresql-133-127-1112-1017-and-9622-released-2210/.

18. Apache HBase — Национальная библиотека им. Н. Э. Баумана [Електронний ресурс] – Режим доступу: https://ru.bmstu.wiki/Apache_HBase

19. Beginning Java MVC 1.0: Model View Controller Development to Build Web, Cloud, and Microservices Applications 1st ed. Edition.

20. Top 6 Most Important Benefits of MVC Architecture for Web Application Development Process [Електронний ресурс] – Режим доступу:https://siyainfo.com/blog/top-6-important-benefits-mvc-architecture-web-application-development-process/.