Міністерство освіти і науки України
Кам’янець-Подільський національний університет імені Івана Огієнка
Фізико-математичний факультет
Кафедра комп’ютерних наук

Дипломна робота

бакалавра

з теми: “ ДОСЛІДЖЕННЯ ФРЕЙМВОРКІВ FARM СТЕКУ ДЛЯ РОЗРОБКИ ПРОЄКТІВ ЩОДО ОНЛАЙН ПЛАНУВАННЯ ЗАВДАНЬ.”

Виконав: студент 3 курсу, KNms1-B21 групи
спеціальності 122 “Комп’ютерні науки”
Гуменюк Олександр Олександрович
Керівник:
Моцик Ростислав Васильович
доцент кафедри комп’ютерних наук,
кандидат педагогічних наук, доцент
Рецензент:
Сморжевський Ю.Л.
доцент кафедри математики,
кандидат педагогічних наук, доцент

Кам’янець-Подільський, 2024 р.

АНОТАЦІЯ

Дипломна робота присвячена дослідженню та аналізу фреймворків FARM стеку для розробки проєктів з онлайн планування завдань. У вступі роботи обґрунтовується актуальність теми, враховуючи зростаючу популярність і важливість онлайн платформ для планування роботи та керування проєктами в сучасному світі, де більшість процесів перенесено в онлайн-режим. Робота охоплює дослідження сучасних веб-технологій, аналіз потреб і вимог користувачів, а також переваг використання FARM стеку (FastAPI, React, MongoDB) для розробки онлайн інструментів для планування завдань.

Метою роботи є оцінка можливостей і переваг використання цих фреймворків та їх комбінації для створення ефективних і надійних онлайн-інструментів. У процесі виконання роботи автор дослідив теоретичні аспекти, провів аналіз існуючих програмних розробок, обґрунтував вибір інструментальних засобів розробки та реалізував програмний засіб. Особлива увага приділялася безпеці, продуктивності та зручності використання розробленого додатка.

У першому розділі роботи детально розглянуто сучасні веб-технології та їх можливості для створення онлайн платформи для планування завдань, а також описано потреби та вимоги користувачів. Автор досліджує можливості та переваги використання FARM стеку, який включає фреймворки FastAPI, React і MongoDB, для розробки онлайн інструментів для планування завдань. Проведено огляд та аналіз аналогічних програмних розробок, що дозволило визначити сильні та слабкі сторони існуючих рішень.

Другий розділ присвячено безпосередньо дослідженню фреймворків FastAPI, React і MongoDB для розробки проєктів щодо онлайн планування завдань. Описано постановку задачі, призначення та вимоги до програмного засобу, методологію розробки, загальний опис проєкту, обґрунтування вибору інструментальних засобів розробки. Особлива увага приділяється програмній реалізації та основним режимам функціонування програмного засобу. Окремо

розглянуто питання організації тестування та налагодження програмного засобу, а також надано рекомендації щодо використання та впровадження розробленого продукту.

Робота містить комплексний підхід до теми, враховує сучасні тенденції та потреби користувачів, виконана на належному рівні і заслуговує високої оцінки.

Ключові слова: FARM стек, FastAPI, React, MongoDB, онлайн планування завдань, веб-технології, розробка програмного забезпечення, веб-застосунок.

ABSTRACT

The thesis is dedicated to the study and analysis of FARM stack frameworks for the development of online task planning projects. The introduction of the thesis substantiates the relevance of the topic, considering the growing popularity and importance of online platforms for work planning and project management in the modern world, where most processes have transitioned to an online mode. The thesis covers the study of modern web technologies, the analysis of user needs and requirements, as well as the advantages of using the FARM stack (FastAPI, React, MongoDB) for developing online task planning tools.

The goal of the thesis is to evaluate the possibilities and advantages of using these frameworks and their combination to create effective and reliable online tools. During the work, the author studied theoretical aspects, analyzed existing software developments, justified the choice of development tools, and implemented a software tool. Special attention was paid to the security, performance, and ease of use of the developed application.

In the first chapter of the thesis, modern web technologies and their capabilities for creating an online task planning platform are discussed in detail, along with the description of user needs and requirements. The author explores the possibilities and advantages of using the FARM stack, which includes the FastAPI, React, and MongoDB frameworks, for developing online task planning tools. A review and analysis of similar software developments were conducted, allowing the identification of the strengths and weaknesses of existing solutions.

The second chapter is directly devoted to the study of the FastAPI, React, and MongoDB frameworks for the development of online task planning projects. The problem statement, purpose, and requirements for the software tool, development methodology, general project description, and justification of the choice of development tools are described. Particular attention is paid to the software implementation and the main operating modes of the software tool. Issues related to the organization of testing and debugging of the software tool are also considered

separately, and recommendations for the use and implementation of the developed product are provided.

The thesis contains a comprehensive approach to the topic, takes into account modern trends and user needs, is executed at an appropriate level, and deserves a high rating.

Keywords: FARM stack, FastAPI, React, MongoDB, online task planning, web technologies, software development, web application.

ВСТУП

Сьогодні, з ростом інформаційних технологій, онлайн планування завдань стає все більш популярним і важливим. Багато компаній та організацій використовують онлайн платформи для планування роботи та керування проєктами. Це спрощує процес роботи, забезпечує більш ефективну комунікацію між співробітниками та гарантує точність та своєчасність виконання завдань.

Актуальність теми бакалаврської роботи полягає в тому, що в сучасному світі, коли більшість процесів перенесена в онлайн-режим, онлайн планування завдань стає все більш популярним і важливим. Значний приріст робочих місць, які не пов'язані з географічним розташуванням, спонукає компанії та організації переходити до онлайн-інструментів для планування роботи та керування проєктами. Це дозволяє спростити процес роботи, забезпечити більш ефективну комунікацію між співробітниками та гарантує точність та своєчасність виконання завдань.

Мета роботи — полягає в дослідженні та аналізі фреймворків FARM стеку для розробки проєктів з онлайн-планування завдань. Кожен з фреймворків FastAPI, React та MongoDB має свої особливості та можливості, які потрібно розглянути окремо та в контексті їх використання в єдиному стеку. Основна мета полягає в оцінці можливостей та переваг використання цих фреймворків та їх комбінації для розробки ефективних та надійних онлайн-інструментів для планування завдань.

Для досягнення поставленої мети необхідно виконати такі **завдання**:

- Дослідити фреймворки FARM стеку для онлайн-планування завдань.

- Проаналізувати особливості та можливості кожного фреймворку окремо та в контексті їх використання в стеку.

- Розробити проєкт щодо онлайн планування завдань із використанням фреймворків Fast API, React, та Mongo DB.

Об’єкт дослідження включає фреймворки FARM стеку для розробки проєктів з онлайн-планування завдань, їх можливості та переваги, а також

користувачів проєктів онлайн-планування завдань на основі FARM стеку та їх потреби та вимоги.

Предмет дослідження включає вивчення фреймворків FARM стеку для розробки проєктів з онлайн планування завдань, їх використання та тестування для розробки нових проєктів. Також предметом дослідження є визначення вимог та потреб користувачів до проєктів онлайн планування завдань на основі FARM стеку, а також розробка нових проєктів, які відповідають їхнім потребам та вимогам.

Структура роботи. Дипломна робота складається зі вступу, трьох розділів, висновку, списку використаних джерел, та додатків.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ІСНУЮЧИХ ПІДХОДІВ ДО СТВОРЕННЯ ОНЛАЙН-ПЛАТФОРМИ ПЛАНУВАННЯ ЗАВДАНЬ

1.1. Дослідження сучасних веб-технологій та їх можливостей для створення онлайн платформи для планування завдань

В сучасному світі вебтехнології відіграють ключову роль у розвитку онлайн сервісів, зокрема, платформ для планування завдань. Для створення таких платформ, які були б ефективними та зручними для користувачів, необхідно використовувати сучасні технології та фреймворки, що гарантують високу продуктивність, безпеку та зручність використання.

Фронтенд фреймворки використовуються для розробки користувацького інтерфейсу веб-додатків, що дозволяє розробникам швидко та ефективно створювати додатки з різноманітною функціональністю. Сьогодні серед найпопулярніших фронтенд фреймворків є React, Vue та Angular.

React - це бібліотека для розробки користувацького інтерфейсу, що акцентується на взаємодії з користувачем. Цей фреймворк дозволяє легко створювати компоненти, які можна повторно використовувати, зменшуючи кількість коду та спрощуючи розробку.

Vue - це ще один популярний фронтенд фреймворк, який також ставить перед собою завдання створення компонентів та їх подальшого використання. Однак Vue відрізняється від React своїм простим та зрозумілим синтаксисом, що дозволяє швидше оволодіти ним та розпочати розробку.

Angular - це фронтенд фреймворк від компанії Google, який використовує TypeScript для написання коду. TypeScript - це мова програмування, яка дозволяє зменшити кількість помилок в коді та зробити його більш структурованим. Хоча Angular має багату бібліотеку компонентів та інструментів для розробки, він вимагає від розробників більш високого рівня знань та досвіду у програмуванні.[20]

У таблиці нижче представлено порівняння React, Vue та Angular за різними параметрами:

Характеристика	**React**	**Vue**	**Angular**
Розмір пакету	Середній	Малий	Великий
Швидкість завантаження	Швидкий	Дуже швидкий	Повільний
Швидкість рендерингу	Швидкий	Швидкий	Повільний
Легкість використання	Середня	Дуже легкий	Складний
Наявність документації та підтримки	Хороша	Добра	Відмінна
Компонентний підхід	Є	Є	Є
Синтаксис шаблонів	JSX	HTML-подібний	HTML-подібний
Система маршрутизації	React Router	Vue Router	Angular Router
Система керування станом	Redux, Context API	Vuex	RxJS, NgRx

Можливість використання TypeScript	Є	Є	Є
Наявність бібліотек та плагінів	Широкий вибір	Добрий вибір	Великий вибір
Рівень підтримки спільнотою	Високий	Високий	Високий
Можливість інтеграції з іншими бібліотеками та фреймворками	Є	Є	Є
Сумісність з різними браузерами та пристроями	Добра	Дуже добра	Дуже добра
Можливість підключення до серверної частини за допомогою API	Є	Є	Є

Підсумок порівняння фреймворків React, Vue та Angular свідчить про те, що кожен з них має свої власні переваги та обмеження, тому вибір між ними залежить від конкретних потреб проєкту та індивідуальних уподобань.[5]

React відзначається хорошою швидкістю рендерингу, середнім розміром пакету та якісною документацією. Vue, натомість, славиться дуже швидкою швидкістю завантаження та рендерингу, легкістю використання та активною спільнотою. Angular, хоча має великий розмір пакету та повільнішу швидкість завантаження та рендерингу, проте компенсує це відмінною підтримкою та багатим функціоналом.

Якщо ви шукаєте швидкий та простий у використанні фреймворк з активною спільнотою, Vue може бути найкращим вибором. Якщо ж вам

важлива швидкість та документація, оберіть React. З іншого боку, якщо ви шукаєте фреймворк із багатим функціоналом та відмінною підтримкою, Angular може вам підійти.

У будь-якому випадку, перед вибором фреймворку важливо здійснити ретельний аналіз потреб проєкту та врахувати переваги та недоліки кожного з них.

CSS (Cascading Style Sheets) є мовою опису стилів, яка використовується для форматування та оформлення веб-сторінок. Вона дозволяє відокремити контент сторінки від її візуального представлення, що дозволяє змінювати стиль та вигляд сторінки без впливу на сам контент.[11]

Фреймворки CSS є інструментами, які допомагають розробникам ефективно створювати веб-сторінки, надаючи готові рішення та компоненти для оформлення. Найпопулярніші з них - це Tailwind та Bootstrap.

Tailwind CSS - це фреймворк CSS, що дозволяє швидко та ефективно створювати стилізований контент за допомогою класів, що додаються до HTML елементів. Він дозволяє розробникам швидко змінювати стиль та вигляд сторінки, використовуючи передбачувані класи.[18]

Bootstrap - це фреймворк CSS, який надає готові компоненти та стилі для швидкого створення веб-сторінок. Він забезпечує просту та зрозумілу структуру для розробки сторінок та має багато корисних функцій, таких як мобільне перше оформлення та адаптивність.[5]

Препроцесори CSS є інструментами, які дозволяють розробникам писати CSS з використанням більш простого та зрозумілого синтаксису. Sass та Less - це приклади таких препроцесорів, які дозволяють використовувати змінні, вкладені стилі, міксіни та інші зручні функції для ефективної роботи з CSS.

У таблиці нижче представлено порівняння підходів до розробки стилів, які використовується для відображення та оформлення вебсторінок:

Фреймворк / інструмент	**Чистий CSS**	**Sass**	**Less**	**Tailwind CSS**	**Bootstrap**
Розмір файлу CSS	Залежить від проєкту	Більше, ніж звичайний CSS	Більше, ніж звичайний CSS	Більше, ніж звичайний CSS	Більше, ніж звичайний CSS
Час розробки	Висока	Висока	Висока	Низька	Середня
Кількість рядків коду CSS	Залежить від проєкту	Зменшує кількість рядків	Зменшує кількість рядків	Зменшує кількість рядків	Зменшує кількість рядків
Кількість файлів CSS	Залежить від проєкту	1+	1+	1+	1+
Швидкість завантаження сторінки	Висока	Середня	Середня	Низька	Середня
Сумісність з браузерами	Висока	Висока	Висока	Висока	Висока
Підтримка мобільних пристроїв	Висока	Висока	Висока	Висока	Висока

Кількість налаштувань / опцій	Низька	Висока	Висока	Висока	Висока
Документація / підтримка	Середня	Висока	Висока	Висока	Висока

Розробка стилів є важливою частиною будь-якого веб-проєкту, оскільки CSS дозволяє надати проєкту унікальний вигляд і зробити його зручним та привабливим для користувачів.

Існує декілька фреймворків CSS, таких як Tailwind та Bootstrap, які дозволяють швидко створювати веб-інтерфейси за допомогою зручних класів. Крім того, можна розробляти веб-інтерфейси, використовуючи чистий CSS та HTML для створення більш унікального дизайну.

Препроцесори CSS, такі як Sass та Less, дозволяють використовувати змінні, міксіни та інші зручні функції для зменшення кількості коду та полегшення розробки. При використанні фреймворків CSS, таких як Tailwind та Bootstrap, можна використовувати зручні компоненти та класи для оформлення веб-сторінок, що дозволяє значно прискорити розробку та забезпечити більш однорідний вигляд проєкту.

Проте, під час розробки веб-інтерфейсів важливо пам'ятати про зручність та привабливість для користувачів, а також про безпеку та продуктивність веб-сторінок. Тому на розробку стилів слід відводити достатньо часу та використовувати сучасні технології та підходи.

Одним із ключових елементів будь-якої онлайн-платформи є база даних, яка забезпечує зберігання та організацію даних, використованих у системі, та забезпечує швидкий та ефективний доступ до них. При розробці онлайн-платформи для планування завдань вибір бази даних стає особливо важливим,

оскільки вона повинна задовольняти високі вимоги щодо швидкості та надійності доступу до даних.

Реляційні бази даних (SQL) є традиційним вибором для зберігання структурованих даних. Вони використовують мову SQL (Structured Query Language) для створення таблиць, у яких дані зберігаються у вигляді рядків і стовпців.

У кожній таблиці містяться ключі, які унікально ідентифікують кожен запис, а також зв'язки між таблицями, які забезпечують цілісність даних. Реляційні бази даних мають кілька переваг. Вони забезпечують високу точність і надійність даних, що важливо для бізнесу. Використовувати їх легко, оскільки мова запитів SQL проста для вивчення. Також реляційні бази даних дуже гнучкі і можуть зберігати різні типи даних, такі як числа, текст, і дати.

Але є і недоліки. Вони можуть мати проблеми з масштабуванням, коли даних стає дуже багато. Крім того, зміна структури даних може бути складною і призводити до проблем з точністю даних.

Нереляційні бази даних (NoSQL) підходять для зберігання неструктурованих даних, таких як тексти, зображення, і відео. Вони дуже гнучкі і легко масштабуються, що зручно для великих обсягів даних. Також їх структуру даних легше змінювати.

Проте, NoSQL бази даних мають недоліки. Вони не мають єдиного стандарту для мови запитів, що означає, що для доступу до даних треба використовувати різні методи. Також вони забезпечують нижчий рівень точності і надійності даних порівняно з реляційними базами.

4о

Загалом, вибір між реляційними та нереляційними базами даних залежить від потреб системи та її функціональності. При розробці онлайн платформи для планування завдань, розробники повинні обрати той тип бази даних, який найбільш відповідає потребам системи та її користувачів.

Авторизація користувачів є однією з ключових функцій будь-якої онлайн платформи, оскільки вона забезпечує безпеку та конфіденційність даних користувачів і обмежує доступ до ресурсів. Існують різні підходи до авторизації користувачів:

Соціальний вхід дозволяє користувачам увійти на платформу за допомогою свого акаунта в соціальних мережах, таких як Facebook або Google. Це зменшує кількість інформації, яку потрібно вводити під час реєстрації, що робить процес простішим і швидшим. Однак для цього може знадобитися надати додаткові дозволи на доступ до ваших особистих даних.

Реєстрація по електронній пошті та паролю передбачає створення облікового запису, вказуючи свою електронну пошту та пароль. Цей метод забезпечує більшу конфіденційність, оскільки не потребує доступу до інших акаунтів або даних. Проте, це може бути менш зручним, оскільки потрібно запам'ятовувати новий пароль.

Мультифакторна авторизація вимагає введення двох або більше факторів для входу на платформу, наприклад, пароля і коду, що надсилається на мобільний телефон. Це значно підвищує рівень безпеки, оскільки ускладнює доступ для зловмисників, які намагаються зламати обліковий запис.

Використання біометричних даних для авторизації дозволяє користувачам увійти на платформу за допомогою відбитка пальця або розпізнавання обличчя. Це не тільки забезпечує високий рівень безпеки, але й робить процес авторизації швидким і зручним.

Цей метод передбачає використання біометричних параметрів, таких як сканування відбитку пальця або розпізнавання обличчя, для підтвердження особи користувача. Це забезпечує найвищий рівень безпеки та комфорту, оскільки користувачам не потрібно пам'ятати паролі або вводити додаткові дані для входу. Однак цей метод може вимагати наявності спеціального обладнання, наприклад, сканера відбитку пальця, що може зробити його менш доступним для деяких користувачів.

5. Використання токенів для авторизації

Цей підхід передбачає використання токенів, спеціальних кодів, для підтвердження особи користувача. Токени можуть зберігатись у браузері користувача або на сервері платформи. Це забезпечує безпеку та зручність використання, оскільки користувачам не потрібно вводити паролі або інші дані для входу.

Кожен метод авторизації має свої переваги та недоліки, і вибір методу повинен базуватися на потребах та вимогах конкретної платформи. Також важливо врахувати вимоги до безпеки та конфіденційності даних користувачів при виборі методу авторизації.

Щодо розробки веб-додатків, існує багато різних стеків технологій, кожен з яких має свої особливості та переваги. Найпопулярніші з них:

1. Стек LAMP (Linux, Apache, MySQL, PHP): цей стек є одним з найбільш поширених серед веб-розробників. Він включає операційну систему Linux, веб-сервер Apache, базу даних MySQL і мову програмування PHP. LAMP використовується для створення веб-додатків з відкритим кодом, таких як WordPress, Joomla, Drupal, Magento та інші.

2. Стек MEAN (MongoDB, Express.js, AngularJS, Node.js): цей стек використовується для створення веб-додатків на JavaScript. Він включає базу даних MongoDB, фреймворк Express.js для створення веб-додатків на Node.js, AngularJS для клієнтської частини додатка та Node.js для роботи з сервером.

3. Стек FARM (FastAPI, React, MongoDB): цей стек включає в себе високопродуктивний фреймворк FastAPI для створення веб-додатків на мові програмування Python, бібліотеку React для клієнтської частини додатка та базу даних MongoDB.

4. Стек MERN (MongoDB, Express.js, React, Node.js): цей стек також використовується для створення веб-додатків на JavaScript і включає базу даних MongoDB, фреймворк Express.js, бібліотеку React та Node.js.

5. Ruby on Rails: це фреймворк для створення веб-додатків на мові програмування Ruby, який забезпечує швидку розробку та підтримку веб-додатків.[4]

У таблиці нижче представлено порівняння популярних стеків розробки:

Стек розробки	**Популярність**	**Продуктивність**	**Стратегія розробки**	**Спільнота**
LAMP	Популярний серед веброзробників та компаній, що працюють з WordPress та іншими CMS	Швидкість розробки та відповідної роботи стеку залежить від досвіду розробника та обсягу проєкту. Можливості оптимізації є досить обмеженими.	Використовую ться для створення програмного забезпечення на основі WordPress та інших CMS.	Велика спільнота розробників та користувачів
MEAN	Популярний серед веброзробників та компаній, що працюють з JavaScript	Швидкість розробки та відповідної роботи стеку залежить від досвіду розробника та обсягу проєкту.	Використовую ться для створення програмного забезпечення на основі JavaScript фреймворків.	Велика спільнота розробників та користувачів

FARM	Новітній стек розробки з швидким API та потужним React інтерфейсом користувача	Швидкість розробки та відповідної роботи стеку залежить від досвіду розробника та обсягу проєкту. Можливості оптимізації є досить великими	Використовую ться для створення програмного забезпечення на основі швидкого API, який співпрацює з потужним React інтерфейсом користувача.	Нова та швидко зростаюча спільнота розробників та користувачів
Ruby on Rails	Популярний серед веброзробників та стартапів	Висока продуктивність та швидкість розробки	Стратегія розробки базується на концепції "Convention over Configuration", що спрощує розробку і дозволяє швидко створювати складні вебдодатки.	Велика та активна спільнота розробників та користувачів
MERN	Широко використовується для створення односторінкових додатків та вебсерверів	Швидкість розробки та відповідної роботи залежить від досвіду розробника та обсягу проєкту	Стратегія розробки базується на використанні JavaScript фреймворків та інших вебтехнологій для створення	Велика спільнота розробників та користувачів

			повноцінних вебдодатків.	

Отже, вивчення останніх вебтехнологій показує, що для створення онлайн платформи для планування завдань можна використовувати різні набори інструментів для розробки, такі як LAMP, MEAN, FARM, MERN та інші. Кожен з цих наборів має свої переваги та особливості, тому вибір конкретного набору залежить від вимог та потреб проекту.

Наприклад, LAMP stack ідеально підходить для розробки вебдодатків з відкритим кодом, MEAN та MERN стеки дозволяють створювати вебдодатки на JavaScript з високою продуктивністю та багатофункціональністю, а FARM stack забезпечує швидку роботу з сервером та підтримує асинхронну обробку запитів.

Загалом, використання сучасних вебтехнологій та фреймворків дозволяє створювати ефективні та зручні для користувачів онлайн платформи для планування завдань, що сприяє їх популярності та успіху на ринку.

1.2. Загальний огляд потреб та вимог користувачів

Онлайн платформи для організації робочого процесу та управління часом користувачів набувають все більшої популярності. Однак, для того, щоб такі платформи стали успішними та задовольняли потреби своїх користувачів, необхідно вивчати їхні потреби та вимоги. В даному розділі будуть описані потреби та вимоги користувачів до онлайн платформ для планування завдань.

У першу чергу, користувачі мають потребу у зручному та простому інтерфейсі, який дозволить швидко та легко організовувати їх завдання та плани. Вони бажають, щоб платформа була інтуїтивно зрозумілою та не вимагала великої кількості часу для оволодіння її функціоналом.

Крім того, користувачам необхідно мати можливість працювати з платформою з будь-якого місця та на будь-якому пристрої, що зумовлює необхідність розробки онлайн версії платформи та мобільних додатків.

Користувачі також потребують можливості налаштування своїх профілів та завдань, щоб адаптувати платформу під свої потреби. Наприклад, вони можуть бажати встановлювати терміни виконання завдань, пріоритети, категоризувати завдання та інше.

Користувачам потрібно мати зручну та швидку можливість створювати, редагувати та переглядати завдання. Вони очікують, щоб платформа була легкою в освоєнні та не вимагала значної витрати часу на оволодіння її функціоналом.

Крім цього, користувачі очікують, що платформа буде надійною та безпечною, забезпечуючи захист їхніх даних та їх належне зберігання. Важливо, щоб платформа працювала швидко і не потребувала великої кількості ресурсів.

Також користувачі очікують, що платформа підтримуватиме різні формати завдань, такі як текст, зображення, відео тощо, і надаватиме різноманітні інструменти для редагування та організації завдань.

Крім того, користувачі очікують можливості синхронізації платформи з іншими додатками та сервісами, такими як Google Calendar, Outlook та інші.

Загальною вимогою користувачів є можливість безкоштовно працювати з платформою, а також можливість розширення функціоналу платформи за плату.

Отже, користувачі онлайн платформ для планування завдань очікують наступного:

- Зручного та простого інтерфейсу, який дозволить швидко та легко організовувати свої завдання та плани.

- Можливості працювати з платформою з будь-якого місця та на будь-якому пристрої.

- Можливості налаштування своїх профілів та завдань, щоб підлаштувати платформу під свої потреби.

- Простоти та швидкості створення, редагування та перегляду завдань.

- Надійності та безпеки платформи, що гарантує захист та належне зберігання даних.

- Швидкості та ефективності роботи платформи.

- Підтримки різних форматів завдань, таких як текст, зображення, відео тощо, та наявності різних інструментів для їх редагування та організації.

- Можливості синхронізації з іншими додатками та сервісами, такими як Google Calendar, Outlook та інші.

- Можливості безкоштовної роботи з платформою з можливістю розширення її функціоналу за плату.

Загалом, для задоволення потреб та вимог користувачів важливо забезпечити зручний та інтуїтивно зрозумілий інтерфейс, а також надійну та ефективну роботу платформи з різноманітними форматами завдань та можливістю синхронізації з іншими додатками та сервісами.

1.3. Дослідження можливостей та переваг використання FARM стеку для розробки онлайн інструментів для планування завдань

FARM стек є одним з найбільш популярних стеків технологій для створення вебдодатків на сьогоднішній день, складаючись з трьох ключових компонент: FastAPI, React.js та MongoDB. Кожна з цих технологій має свої унікальні переваги, які в поєднанні зі стеком дозволяють розробникам створювати високопродуктивні та масштабовані вебдодатки.[9]

FastAPI є одним з найшвидших та найпродуктивніших фреймворків Python для створення вебдодатків, пропонуючи багато функціональних можливостей. Цей фреймворк забезпечує високу продуктивність завдяки асинхронному виконанню коду та струминовій передачі даних, що дозволяє обробляти багато запитів одночасно.

Основні переваги використання FastAPI для розробки онлайн інструментів для планування завдань включають:

1. Швидкість та продуктивність — FastAPI пропонує швидке та продуктивне виконання коду завдяки використанню асинхронного виконання коду та струминової передачі даних (streaming).

2. Вбудована валідація даних - FastAPI має вбудовану валідацію даних, що дозволяє перевіряти правильність введення даних на стороні сервера та запобігає введенню некоректних даних.

3. Легка документація API — FastAPI автоматично генерує документацію API за допомогою OpenAPI та JSON Schema, що дозволяє зрозуміти деталі API та забезпечити легку інтеграцію з іншими сервісами.

4. Легка інтеграція з Pydantic — FastAPI легко інтегрується з Pydantic, що дозволяє створювати валідні моделі даних та забезпечує більш впорядкований та читабельний код.

React.js, з іншого боку, є бібліотекою для розробки інтерфейсів користувачів вебдодатків, що була створена Facebook. Вона вражає розробників своєю ефективністю, масштабованістю та зручністю у використанні.[14]

React має кілька важливих переваг. Він забезпечує високу швидкість та ефективність завдяки використанню віртуальної DOM, що зменшує кількість маніпуляцій з реальною DOM і прискорює роботу додатків. React також легко масштабується, підтримуючи розподілений рендеринг, що дозволяє створювати великі та складні додатки, які працюють на різних пристроях.

Ще одна перевага React – його модульність. Розробники можуть розділяти додатки на окремі компоненти, що спрощує процес розробки та підтримки коду. Крім того, React має велику та активну спільноту розробників. Це означає, що є багато корисних пакетів і підтримка бібліотеки, що створює зручне середовище для роботи.

Нарешті, React дуже легкий у використанні. Він дозволяє швидко створювати складні інтерфейси користувачів та легко вносити зміни в код, що робить його зручним для розробників.

Узагальнюючи, React є однією з найпопулярніших та найефективніших бібліотек для розробки інтерфейсів користувача, що забезпечує розробникам зручне та швидке середовище для створення складних вебдодатків.

MongoDB - це база даних орієнтована на документи, яка дозволяє зберігати та обробляти дані у форматі JSON. Ця система є однією з найбільш популярних NoSQL баз даних у світі, оскільки вона пропонує зберігання даних у вигляді документів зі структурованим та гнучким форматом.[9]

MongoDB має кілька важливих переваг. Вона забезпечує гнучкість і горизонтальну масштабованість, дозволяючи додавати нові сервери для збільшення продуктивності. Це спрощує процес масштабування бази даних зі зростанням обсягу інформації.

Ще одна перевага MongoDB – це швидкість і продуктивність. Вона забезпечує швидке зберігання та обробку великих обсягів даних завдяки своїй архітектурі та можливостям оптимізації запитів. MongoDB також відзначається простим та зрозумілим інтерфейсом для розробників, що дозволяє швидко освоїти роботу з базою даних.

Ця база даних широко використовується у великих проєктах, таких як Google, Adobe, і Cisco. Нарешті, MongoDB забезпечує високий рівень безпеки, маючи вбудовані засоби автентифікації та шифрування даних, що гарантує захист інформації.

У контексті розробки онлайн інструментів для планування завдань, використання MongoDB дозволяє зберігати та обробляти великі обсяги даних, що є корисним для зберігання та аналізу інформації про завдання та їх виконання. Крім того, гнучкість та горизонтальна масштабованість MongoDB спрощують процес розширення онлайн платформи зі зростанням користувацької бази та обсягу даних.

Однією з основних переваг використання FARM стеку для створення онлайн інструментів для планування завдань є швидкість розробки. За допомогою FastAPI розробники можуть швидко створювати веб-додатки, React.js дозволяє швидко розширювати функціональність, а MongoDB спрощує зберігання та організацію даних. Це робить процес розробки нових функцій та внесення змін у застосунок простим і швидким.

Іншою перевагою використання FARM стеку є масштабованість. MongoDB дозволяє легко масштабувати базу даних, а FastAPI та React.js — легко масштабувати серверну та клієнтську частини додатка відповідно. Це дозволяє збільшувати обсяг даних та кількість користувачів без втрати продуктивності.

Крім того, FARM стек забезпечує безпеку додатка. FastAPI має вбудовану підтримку аутентифікації та авторизації, що спрощує забезпечення безпеки. React.js дозволяє використовувати бібліотеки для захисту від XSS-атак та інших видів атак.

Також використання FARM стеку дозволяє зменшити витрати на розробку та підтримку додатка. Завдяки швидкості розробки та масштабованості, розробка та підтримка додатка стають більш ефективними та економічно доцільними.

Отже, використання FARM стеку для розробки онлайн інструментів для планування завдань має багато переваг, таких як швидкість розробки, масштабованість, безпека та економічність. З урахуванням потреб користувачів та вимог до онлайн платформи для планування завдань, використання FARM стеку може бути оптимальним вибором для розробки даного додатка.

1.4. Огляд та аналіз аналогічних програмних розробок

"Trello" - це інструмент для управління проектами, який створений компанією Atlassian та доступний як веб-застосунок і мобільний застосунок.

Він організовує ваші проекти у вигляді дошок, де ви можете бачити, над чим працюєте, хто працює над чим, і в якому стані знаходяться завдання.

Основні функції та характеристики:

Управління проектами та завданнями за допомогою дошок, списків та карток. "Trello" використовує цю систему для організації та пріоритезації вашого робочого процесу в зручний та гнучкий спосіб.

Здатність до співпраці та обміну коментарями: Користувачі можуть коментувати картки, що дозволяє обмінюватися думками та ідеями щодо конкретних завдань або проєктів.

Інтеграція з іншими інструментами: Trello може легко інтегруватися з іншими інструментами, такими як Google Drive, Slack, Jira та інші, що робить його потужним інструментом для управління проєктами.

Налаштування сповіщень та дедлайнів: Користувачі можуть встановлювати сповіщення та дедлайни для карток, що допомагає тримати проєкти під контролем та вчасно виконувати завдання.

Підтримка мобільних додатків для iOS та Android: Trello має мобільні додатки для обох платформ, що дозволяє користувачам керувати своїми проєктами з будь-якого місця.

Переваги: Однією з ключових переваг Trello є його простий та інтуїтивно зрозумілий інтерфейс, що спрощує створення, керування та відстеження проєктів. Також, Trello надає велику гнучкість у налаштуванні проєктів, що дозволяє користувачам адаптувати інструмент до своїх потреб. Широкі можливості для співпраці, такі як обмін коментарями та інтеграція з іншими інструментами, також є важливими перевагами Trello.

Недоліки: Проте, Trello має деякі недоліки. Обмежена функціональність у безкоштовній версії може бути перешкодою для деяких користувачів, особливо для тих, хто має великі проєкти або команди. Крім того, відсутність вбудованих інструментів для графічного представлення даних може бути обмеженням для користувачів, які потребують більш детального аналізу своїх проєктів.

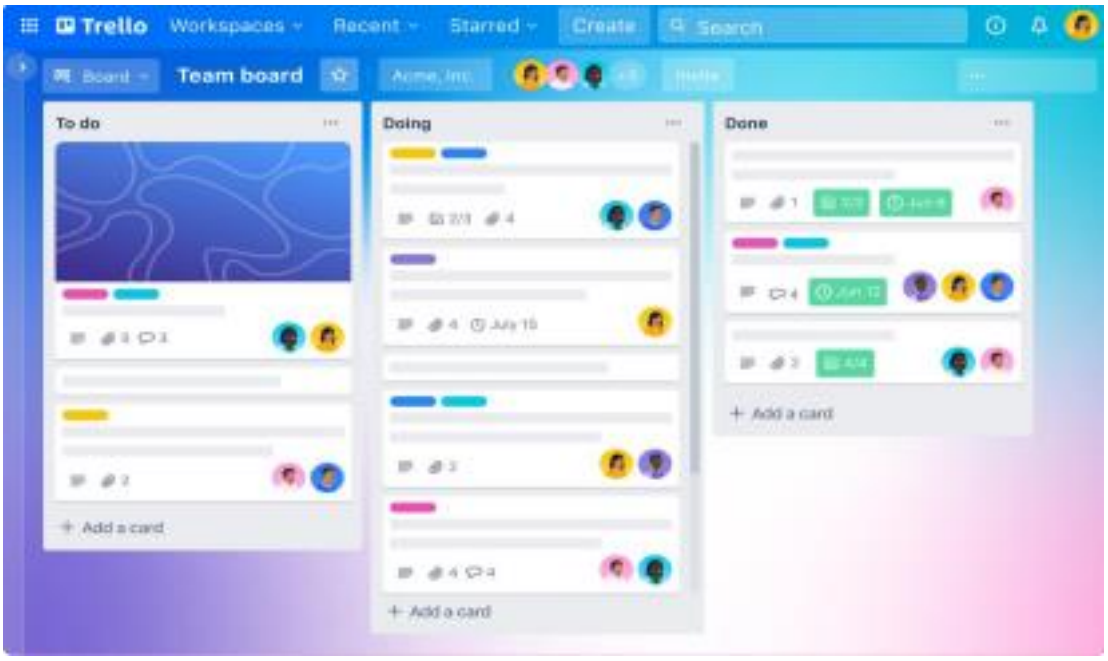

Рисунок 1.1 – Головне меню вебзастосунку «Trello»

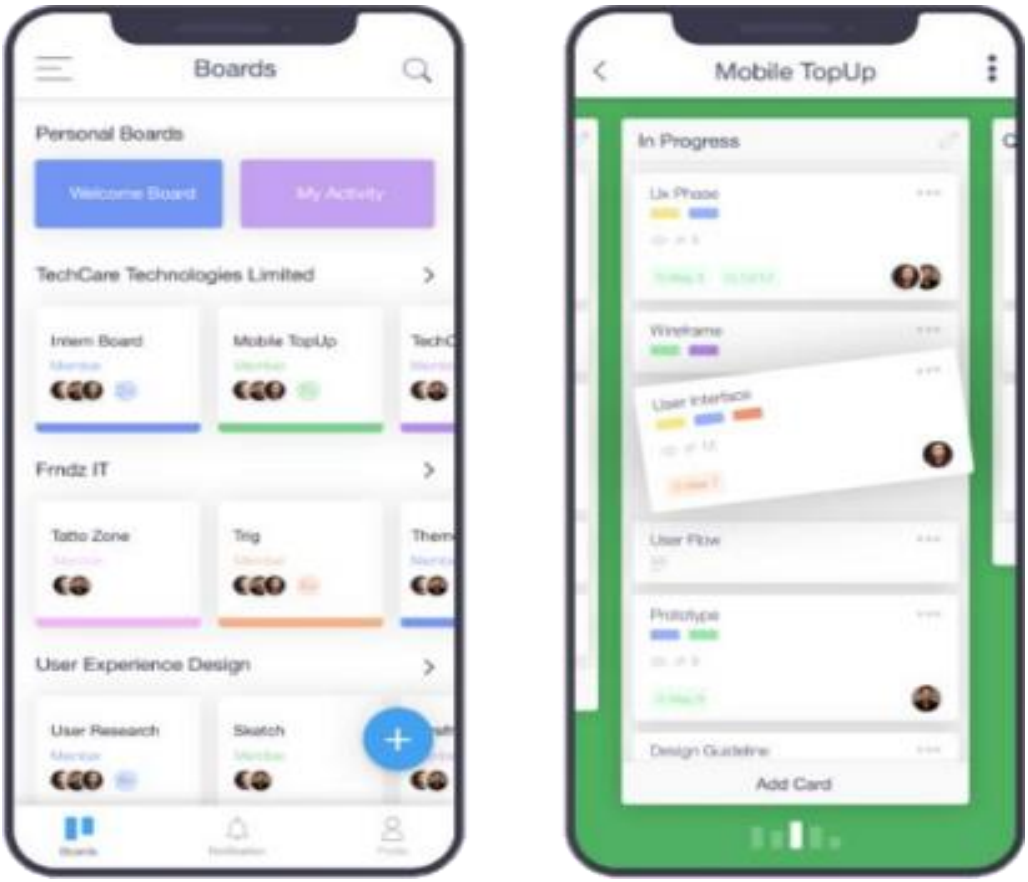

Рисунок 1.2 – Головне меню мобільного застосунку «Trello»

"Asana" - це ще один інструмент для управління проєктами, доступний у вебі та у вигляді мобільного додатка. Він пропонує широкий спектр інструментів для організації, відстеження та управління роботою.

Основні функції та характеристики:

1. Управління проєктами та завданнями через списки, дошки та календарі, що дозволяє створювати та керувати завданнями, встановлювати дедлайни та призначати відповідальних.

2. Можливість співпраці та обміну коментарями, що сприяє ефективній комунікації в команді.

3. Інтеграція з іншими інструментами, такими як Google Drive, Slack, Microsoft Teams та інші, що робить його ще більш потужним для управління проєктами.

4. Можливість налаштування сповіщень та дедлайнів для завдань, що допомагає тримати проєкти під контролем та вчасно виконувати завдання.

5. Підтримка мобільних додатків для iOS та Android, що дозволяє керувати проєктами з будь-якого місця.

Переваги:

Asana пропонує широкі можливості для управління проєктами, що дозволяє створювати складні проєкти з багатьма завданнями та учасниками. Також, вона забезпечує гнучкість у налаштуванні завдань та має вбудовані інструменти для візуалізації даних, що допомагає отримати глибокий аналіз проєктів.

Недоліки:

Проте, Asana може бути складною для нових користувачів через великий набір функцій. Крім того, висока вартість преміум-версії може бути перешкодою для деяких користувачів, особливо для тих, хто має великі команди або складні проєкти.

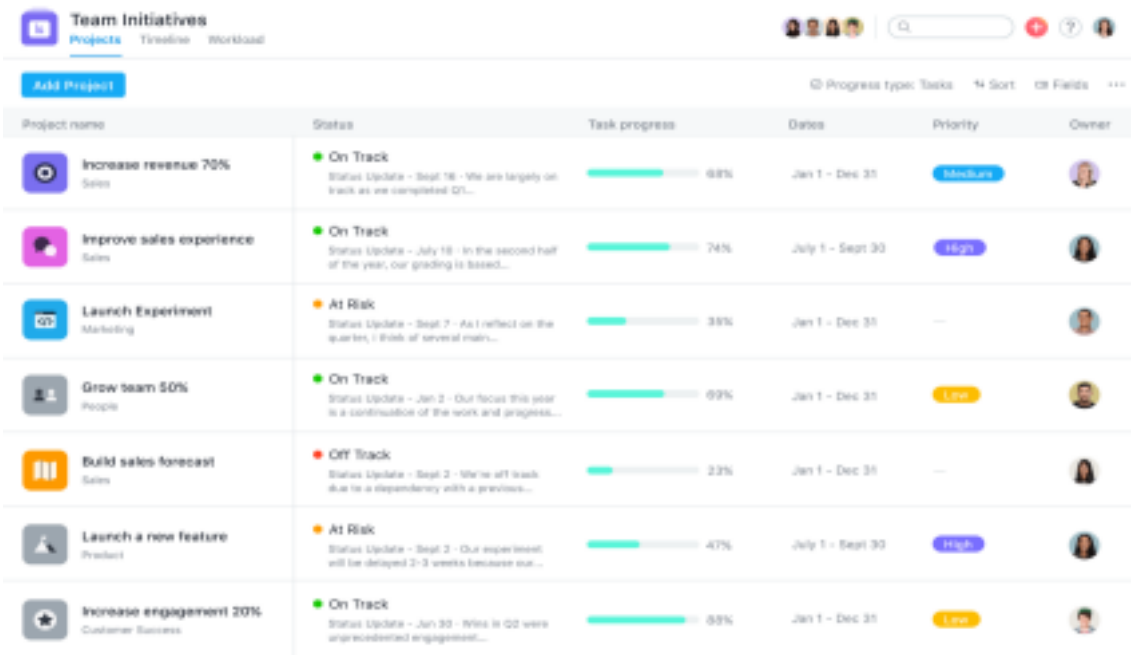

Рисунок 1.3 – Головне меню вебзастосунку «Asana»

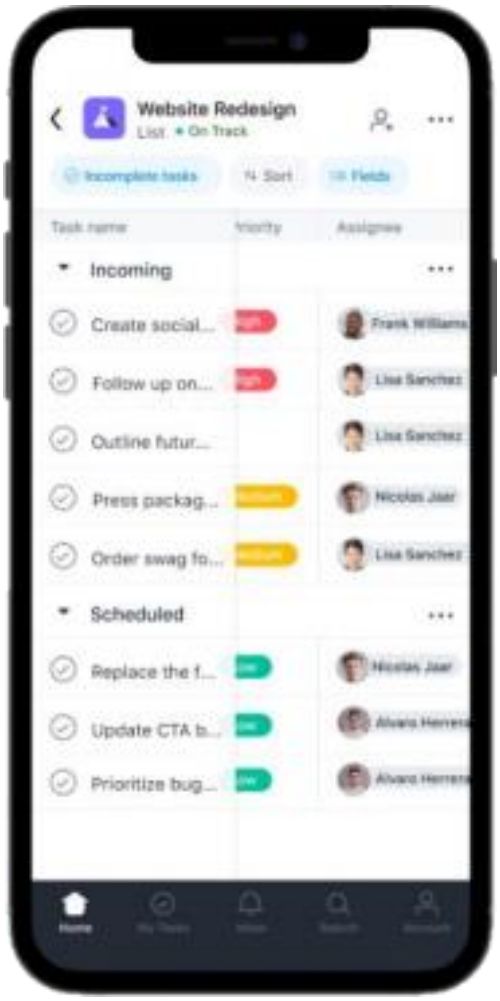

Рисунок 1.4 – Головне меню мобільного застосунку «Asana»

"Jira" - це інструмент для управління проєктами, розроблений компанією Atlassian, і доступний як вебзастосунок та мобільний застосунок. Він є потужним засобом для трекінгу помилок, завдань та інших проєктних вимог.

Основні функції та характеристики:

1. Управління проєктами та завданнями через систему відстеження помилок та завдань, що дозволяє створювати, відстежувати та керувати ними в рамках проєктів.

2. Можливість співпраці та обміну коментарями, що сприяє ефективній комунікації в команді.

3. Інтеграція з іншими інструментами, такими як Confluence, Bitbucket, Bamboo та інші, що робить його більш потужним для управління проєктами.

4. Можливість налаштування сповіщень та дедлайнів для завдань, що допомагає тримати проєкти під контролем та вчасно виконувати завдання.

5. Підтримка мобільних додатків для iOS та Android, що дозволяє керувати проєктами з будь-якого місця.

Переваги:

Jira відома своїми потужними можливостями для управління проєктами, включаючи трекінг помилок та завдань, планування спринтів, управління ресурсами та інше. Вона може бути налаштована для відповідності специфічним потребам команди або проєкту, що робить її дуже гнучким інструментом.

Недоліки:

Однак, Jira може бути складною для нових користувачів через великий набір функцій. Крім того, вона може бути дорога для невеликих команд або індивідуальних користувачів, особливо якщо вони потребують доступу до додаткових функцій або інтеграцій.

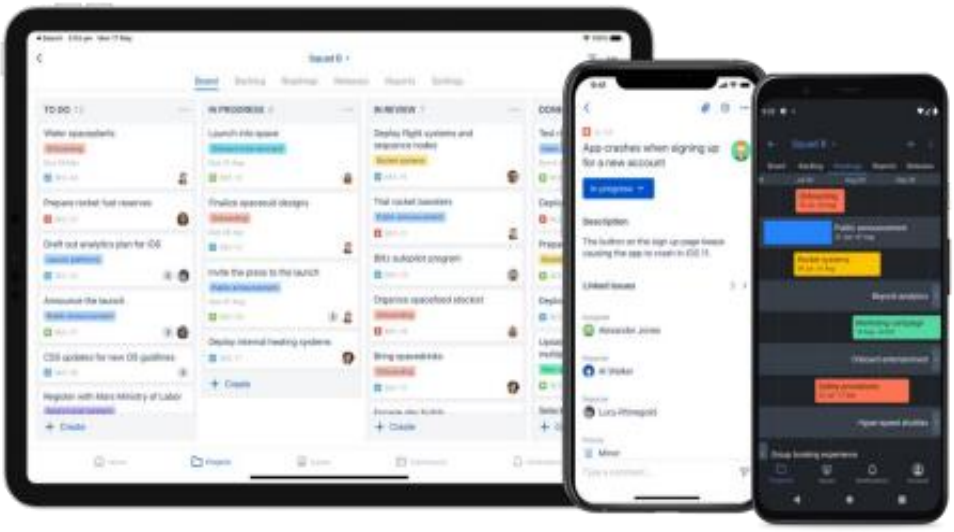

Рисунок 1.4 – Головне меню застосунку «Jira»

Висновки: У цьому розділі було досліджено сучасні веб-технології та їх можливості для створення онлайн-платформи для планування завдань. Зокрема, було розглянуто фронтенд фреймворки, такі як React, Vue та Angular, що дозволяють розробляти ефективні та зручні для користувачів інтерфейси. Особлива увага приділялася можливостям та перевагам використання FARM стеку, який включає фреймворки FastAPI, React та MongoDB для розробки онлайн інструментів для планування завдань. Проведений огляд та аналіз аналогічних програмних розробок допоміг визначити сильні та слабкі сторони існуючих рішень та обґрунтувати вибір інструментальних засобів для розробки власного проєкту.

РОЗДІЛ 2 ДОСЛІДЖЕННЯ ФРЕЙМВОРКІВ FastAPI, React, MongoDB ДЛЯ РОЗРОБКИ ПРОЄКТІВ ЩОДО ОНЛАЙ ПЛАНУВАННЯ ЗАВДАНЬ

2.1. Постановка задачі, призначення та вимоги до програмного засобу

Мета цього дослідження полягає у розробці веб-додатку для онлайн планування завдань, використовуючи стек технологій FastAPI, React та MongoDB (див. рис. 2.1). Основним завданням є створення веб-додатку, що дозволить користувачам створювати, редагувати, видаляти та відстежувати завдання, а також організовувати їх у проєкти. Застосунок також має підтримувати співпрацю між користувачами, дозволяючи їм ділитися проєктами та завданнями та обмінюватися коментарями.

Цей програмний продукт призначений для використання у різних сферах, включаючи бізнес, освіту, наукові дослідження та інші. Він може служити для управління проєктами, координації команд, відстеження прогресу завдань, спілкування та співпраці між учасниками проєкту.

Вимоги до програмного засобу включають кілька важливих аспектів. Застосунок повинен мати повний набір інструментів для керування завданнями та проєктами. Це означає, що користувачі повинні мати можливість створювати, редагувати, видаляти та відстежувати завдання, організовувати їх у проєкти, обмінюватися коментарями та співпрацювати між собою.

Інтерфейс додатку повинен бути простим і зрозумілим для всіх користувачів, незалежно від їх технічної підготовки. Важливо, щоб застосунок працював на різних пристроях, включаючи настільні комп'ютери, ноутбуки, планшети та смартфони, з оптимізованим інтерфейсом для кожного типу пристрою.

Безпека даних користувачів має бути на високому рівні, захищаючи їх від несанкціонованого доступу та втрати. Застосунок повинен захищатися від

SQL-ін'єкцій, XSS-атак та інших загроз безпеки, а також використовувати захищені протоколи передачі даних, такі як HTTPS.

Крім того, застосунок повинен працювати швидко та ефективно, незалежно від обсягу завдань або проєктів. Це включає оптимізацію запитів до бази даних, ефективне кешування та використання асинхронних операцій, де це можливо.

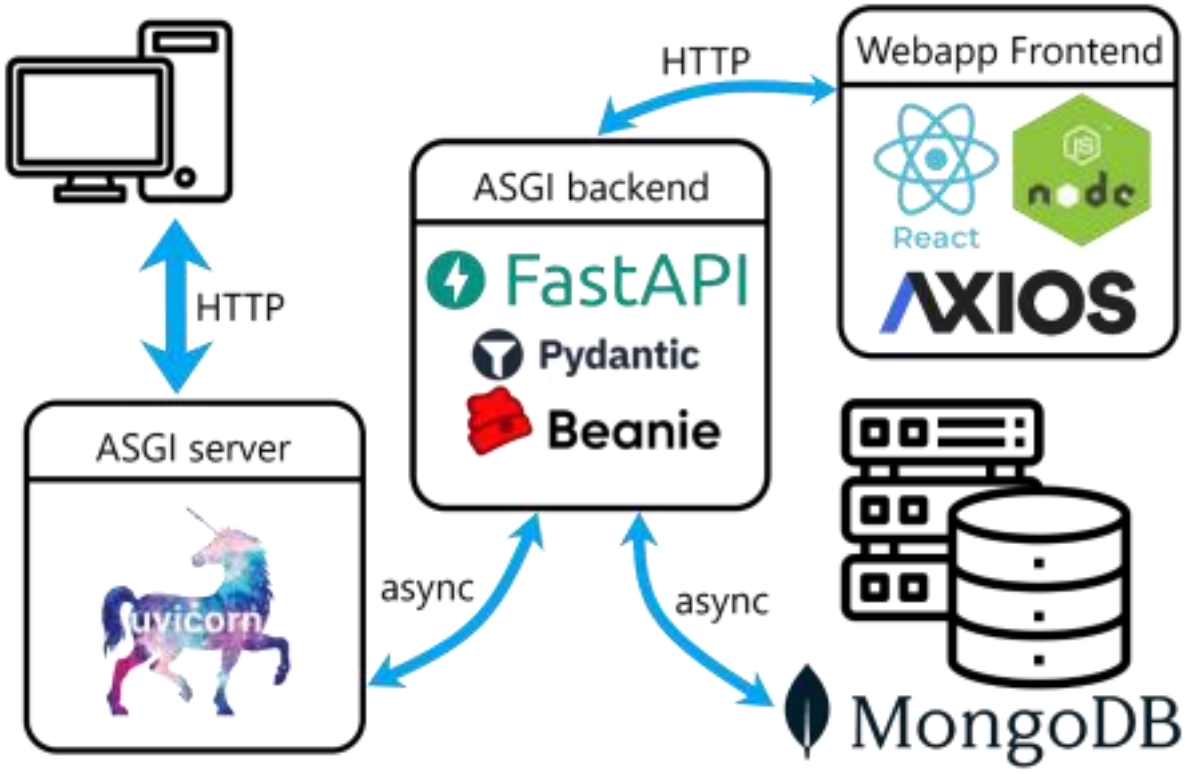

Рисунок 2.1 – Схема архітектури додатку

2.2. Методологія розробки програмного засобу

Методологія, яка буде використана для створення цього вебдодатку, - Solo Scrum. Це адаптація відомої методології Scrum, спеціально адаптована для одного розробника. Ця методологія використовує багато тих же принципів, що й Scrum, таких як ітеративний розвиток, самоорганізація та постійна інспекція та адаптація, але адаптовує їх для умов, коли роботу виконує лише одна людина.

Основні складові Solo Scrum допомагають розробнику ефективно керувати проєктом на одному. Вони включають Product Backlog, який містить всі завдання проєкту, такі як розробка нових функцій або виправлення помилок. Кожен Sprint, що триває від одного до чотирьох тижнів, фокусується на виконанні конкретного набору завдань з Product Backlog. Щоденна коротка зустріч, відома як Daily Scrum, допомагає розробнику відстежувати прогрес, вирішувати проблеми і планувати дії на наступний день. По закінченню кожного Sprint відбувається Sprint Review і Retrospective, під час якого розробник аналізує свою роботу і планує покращення на майбутнє.[2]

Рисунок 2.2 – Схема процесу Solo Scrum

Використання Solo Scrum сприяє підтримці високого рівня продуктивності та організації, навіть у випадку, коли ви працюєте самостійно. Ця методологія також гарантує, що робота над проєктом розвивається у правильному напрямку, а всі важливі завдання отримують належну увагу.

Важливо зауважити, що хоча Solo Scrum спеціально адаптований для одного розробника, він все ж потребує дотримання основних принципів Agile.

Це означає, що розробник має бути готовий до змін, відкритий до нових ідей та постійно шукати шляхи для поліпшення своєї роботи.

2.3. Загальний опис проєкту

Цей проект представляє собою веб-застосунок для онлайн планування завдань, створений з використанням технологічного стеку FastAPI, React та MongoDB. Його основна мета полягає в тому, щоб допомогти користувачам управляти своїми завданнями та проєктами (див. рис. 2.3).

Рисунок 2.3 – Схема взаємодії FastAPI, React та MongoDB

Проект створено з використанням мікросервісної архітектури, що дозволить забезпечити високу масштабованість та простоту управління. Основні компоненти програмного забезпечення включають сервер API на основі FastAPI, який оброблятиме всі запити від користувачів і взаємодіятиме з базою даних MongoDB. [1]

Клієнтський інтерфейс, розроблений з використанням React, буде відображати інформацію для користувачів і надсилати запити до сервера API.

База даних MongoDB буде зберігати інформацію про користувачів, проєкти та завдання. Програма також надасть різноманітні функції для управління завданнями та проєктами, включаючи створення, редагування, видалення та відстеження завдань, управління проєктами, співпрацю та обмін коментарями між користувачами. Проєкт буде розроблено відповідно до методології Agile Scrum з двотижневими спринтами, що включатимуть етапи планування, розробки, тестування та огляду. Крім того, щоденно проводитимуться стоянки для вирішення проблем. [3]

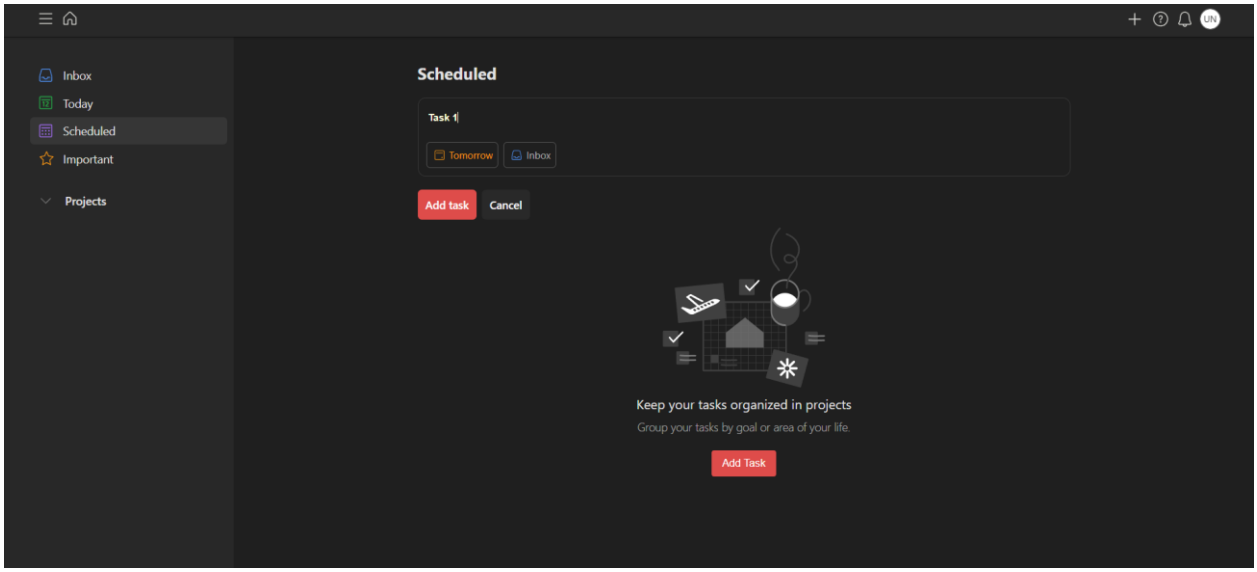

Рисунок 2.4 – Створення нового проєкту

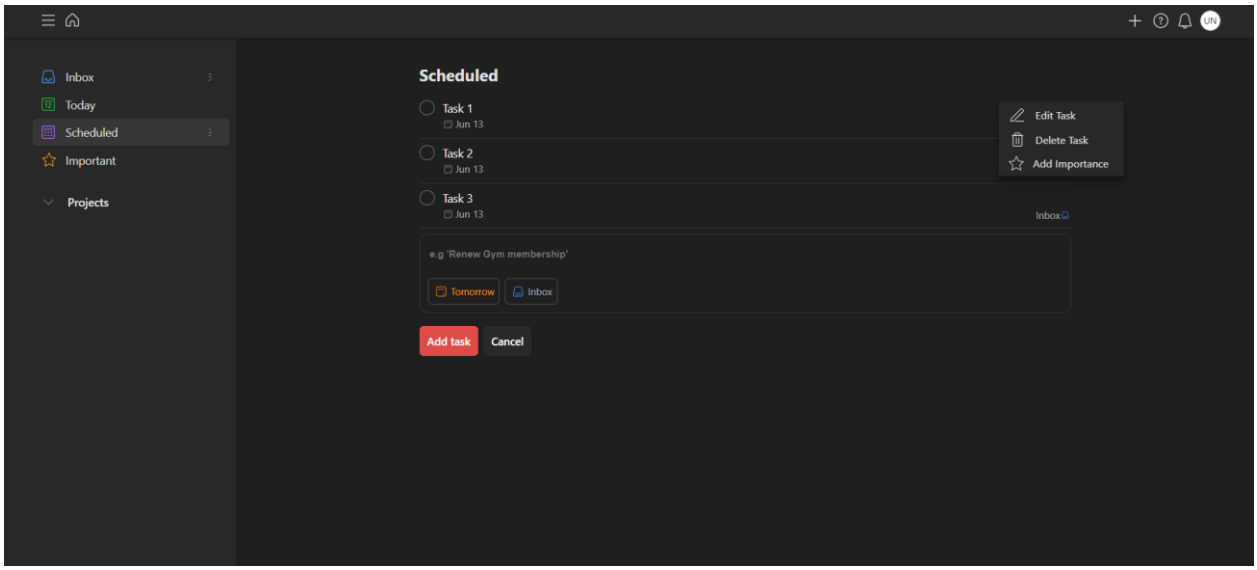

Рисунок 2.5 – Дошка із завданнями

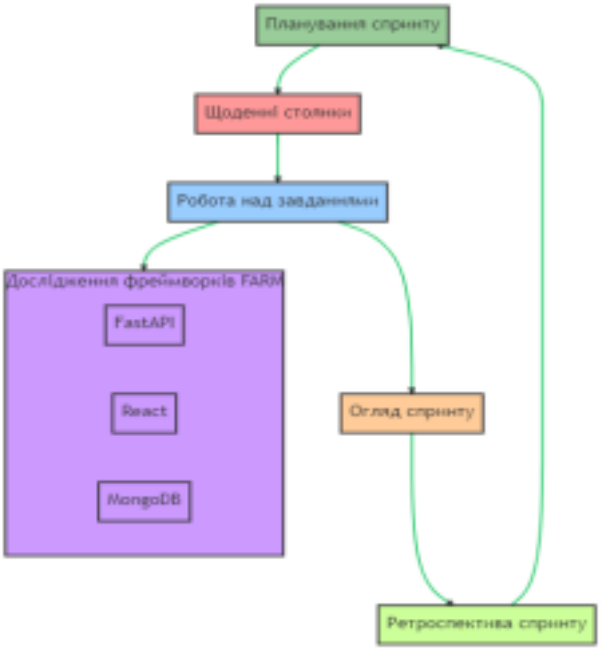

Рисунок 2.6 - План розробки

2.4. Обґрунтування вибору інструментальних засобів розробки

Для створення вебдодатку для онлайн планування завдань було обрано технологічний стек, що включає FastAPI, React та MongoDB. Ці інструменти було відібрано через їх унікальні характеристики, широке визнання в індустрії та відповідність вимогам проєкту.

FastAPI обрано для створення серверного API через його швидкість та високу продуктивність, що дозволяє ефективно обробляти запити від клієнтів. Також простий синтаксис і можливість автоматичної генерації документації зробили його зручним вибором для розробників.

React обрано для клієнтського інтерфейсу завдяки його компонентній архітектурі, що сприяє повторному використанню коду та полегшує підтримку додатків. Активна спільнота розробників і висока продуктивність завдяки віртуальному DOM роблять React відмінним вибором для розробки вебдодатків.[17]

MongoDB обрано для бази даних через її гнучкість у схемі даних і можливість горизонтального масштабування. Вона забезпечує високу продуктивність операцій з даними, що робить її ідеальним вибором для високонавантажених вебдодатків.

Вибір цих технологій було здійснено на основі їх унікальних характеристик, широкого визнання в індустрії та відповідності вимогам проєкту. Вони надають потужні можливості для розробки сучасних, високопродуктивних вебдодатків, а також мають великі спільноти, що забезпечують велику кількість ресурсів для навчання та підтримки.

2.4. Програмна реалізація та основні режими функціонування програмного засобу

Програмна реалізація проєкту має багатошарову структуру, включаючи різні модулі, які відповідають за окремі функції системи. Основні режими роботи програмного забезпечення включають реєстрацію та аутентифікацію користувачів, створення та управління проєктами, а також створення та управління завданнями в межах цих проєктів. Аутентифікація користувачів здійснюється за допомогою електронної пошти та пароля (рис. 2.9). Система надає можливість реєстрації нових користувачів і входу в систему для вже існуючих. Після введення правильних облікових даних користувача система генерує токен доступу (рис. 2.7), який використовується для подальшої аутентифікації запитів від користувача (рис. 2.8).[7]

```python
def verify_password(plain_password: str, hashed_password: str) -> bool:
    """Verify a hashed password and a plain password."""
    return pwd_context.verify(plain_password, hashed_password)


def get_password_hash(password: str) -> str:
    """Hash a password for storing."""
    return pwd_context.hash(password)


def create_access_token(
    subject: str | Any,
    expires_delta: timedelta | None = None,
) -> str:
    """Create a JWT access token."""
    if expires_delta:
        expire = datetime.utcnow() + expires_delta
    else:
        expire = datetime.utcnow() + timedelta(
            settings.ACCESS_TOKEN_EXPIRE_MINUTES,
        )
    payload = {
        "exp": expire,
        "sub": str(subject),
    }
    return jwt.encode(payload, settings.SECRET_KEY, algorithm=ALGORITHM)
```

Рисунок 2.7 - Функції перевірки пароля, генерації хешу пароля та створення JWT токену для аутентифікації користувача

```python
async def authenticate_token(token: str) -> Optional[User]:
    try:
        payload = jwt.decode(
            token,
            settings.SECRET_KEY,
            algorithms=[security.ALGORITHM],
        )
        data = schemas.AuthTokenPayload(**payload)
    except (jwt.JWTError, ValidationError):
        # when the user is not authenticated and auto_error is True.
        raise InvalidTokenError()
    else:
        return await User.get(cast(PydanticObjectId, data.sub))


async def get_current_user(
    token: Optional[str] = Depends(oauth2_scheme),
) -> User:
    """Gets the current user from the database."""
    if token:
        user = await authenticate_token(token)
    else:
        # This is the exception that is raised by the Depends() call
        # when the user is not authenticated and auto_error is True.
        raise InvalidTokenError()
    if not user:
        raise InvalidTokenError()
    return user
```

Рисунок 2.8 - Функції для аутентифікації токену та отримання поточного користувача

```python
@router.post("/login")
async def login(user_credentials: schemas.UserLogin):
    if user := await User.authenticate(email=user_credentials.email, password=user_credentials.password):
        return schemas.AuthToken(authToken=create_access_token(user.id))
    else:
        raise InvalidPasswordOrEmail()


@router.post("/register")
async def register(user_create: schemas.UserCreate):
    if await User.get_by_email(email=user_create.email):
        raise EmailAlreadyRegisteredError()
    elif user_create.password != user_create.password2:
        raise PasswordsDoNotMatchError()
    else:
        user = User(
            name=user_create.name,
            email=user_create.email,
            hashed_password=get_password_hash(user_create.password),
        )
        await user.save()
        return {"success": True, "message": "User created successfully."}
```

Рисунок 2.9 - Endpoints для входу та реєстрації користувача

Після аутентифікації користувачі можуть створювати нові проєкти. При створенні проєкту користувач має можливість вказати електронні адреси інших користувачів, яких слід додати до проєкту. Після створення проєкт зберігається в базі даних, і користувач отримує підтвердження про успішне створення. Крім того, можна отримати детальну інформацію про конкретний проєкт або оновити існуючий проєкт (рис. 2.10).

```python
@router.post("/")
async def post_project(project: schemas.ProjectCreate,
                       user: User = Depends(get_current_user)):
    project_dict = project.dict()
    if emails := project_dict.get("emails"):
        results = await asyncio.gather(*[
            User.get_by_email(email=email) for email in emails
        ])
        users = {user.to_ref() for user in results if user}
        users.add(user.to_ref())
        project_dict["users"] = users
    else:
        project_dict["users"] = [user.to_ref()]
    new_project = Project(**project_dict)
    await new_project.save()
    return {"success": True, "message": "Project created"}

@router.put("/{project_id}")
async def put_project(project_id: PydanticObjectId,
                      project: schemas.ProjectPut,
                      user: User = Depends(get_current_user)):
    project_dict = project.dict(exclude_unset=True)
    project_update = await Project.get(project_id)
    if emails := project_dict.get("emails"):
        results = await asyncio.gather(*(
            User.get_by_email(email=email) for email in emails
        ))
        users = {Link(user.to_ref(), User) for user in results if user}
        project_update = project_update.copy(update=project_dict)
        for user in users:
            if user not in project_update.users:
                project_update.users.append(user)
    await project_update.save()
    return {"success": True, "message": "Project updated"}
```

Рисунок 2.10 - Endpoints для створення та оновлення проєктів

Функція 'post_project' створює новий проєкт на основі вхідних даних і додає користувачів до проєкту. Функція 'put_project' оновлює існуючий проєкт за його ID та може додавати нових користувачів до проєкту.

У межах кожного проєкту користувачі можуть створювати нові завдання. Кожне завдання пов'язане з конкретним проєктом і має набір атрибутів, таких як назва, опис, статус тощо. Користувачі можуть переглядати деталі конкретного завдання, оновлювати існуюче завдання, видаляти завдання або створювати нове завдання (рис. 2.11 – 2.13).

```python
@router.get("/{issues_id}", response_model=schemas.IssueOut)
async def get_issue(issues_id: PydanticObjectId, user: User = Depends(get_current_user)):
    issue = await Issue.get(issues_id)
    return await issue.to_schema()


@router.put("/{issues_id}", response_model=schemas.IssueOut)
async def put_issue(issues_id: PydanticObjectId, issue: schemas.IssuePatch, user: User = Depends(get_current_user)):
    update_data = issue.dict(exclude_unset=True)
    if update_data.get("users"):
        update_data["users"] = [
            Link(DBRef(User.get_motor_collection().name, user["id"]), User) for user in update_data["users"]
        ]
    issue_update = await Issue.get(issues_id)
    if "listPosition" in update_data:
        list_position = int(update_data.get("listPosition")) or 1
        new_status = update_data.get("status")
        await issue_update.calculate_list_position(issue_update.listPosition, issue_update.status,
                                                   list_position, new_status)
    issue_update = issue_update.copy(update=update_data)
    await issue_update.save()
    return await issue_update.to_schema()
```

Рисунок 2.11 - Endpoints для отримання та оновлення інформації про завдання

Функція 'get_issue' повертає деталі завдання за його ID. Функція 'put_issue' оновлює наявне завдання за його ID, використовуючи вхідні дані, що включають оновлення інформації про користувачів та позицію завдання.

```python
@router.delete("/{issues_id}", response_model=schemas.IssueOut)
async def delete_issue(issues_id: PydanticObjectId, user: User = Depends(get_current_user)):
    issue = await Issue.get(issues_id)
    await issue.delete()
    return await issue.to_schema()


@router.post("/", response_model=schemas.IssueOut)
async def post_issue(issue: schemas.IssueIn, user: User = Depends(get_current_user)):
    issue_dict = issue.dict()
    issue_dict["reporter"] = Link(DBRef(User.get_motor_collection().name, issue_dict["reporterId"]), User)
    issue_dict["project"] = Link(DBRef(Project.get_motor_collection().name, issue_dict["projectId"]), Project)
    if issue_dict.get("users"):
        issue_dict["users"] = [
            Link(DBRef(User.get_motor_collection().name, user["id"]), User) for user in issue_dict["users"]
        ]
    if not issue_dict.get("listPosition"):
        issue_dict["listPosition"] = await Issue.get_next_list_position(issue_dict["status"])
    new_issue = Issue(**issue_dict)
    await new_issue.save()
    return await new_issue.to_schema()
```

Рисунок 2.12 - Endpoints для видалення та створення завдань

Функція 'delete_issue' видаляє наявне завдання за його ID. Функція 'post_issue' створює нове завдання, використовуючи вхідні дані, які містять інформацію про користувачів, проєкт та позицію завдання.

```
@router.get("/{issues_id}", response_model=schemas.IssueOut)
async def get_issue(issues_id: PydanticObjectId, user: User = Depends(get_current_user)):
    issue = await Issue.get(issues_id)
    return await issue.to_schema()


@router.put("/{issues_id}", response_model=schemas.IssueOut)
async def put_issue(issues_id: PydanticObjectId, issue: schemas.IssuePatch, user: User = Depends(get_current_user)):
    update_data = issue.dict(exclude_unset=True)
    if update_data.get("users"):
        update_data["users"] = [
            Link(DBRef(User.get_motor_collection().name, user["id"]), User) for user in update_data["users"]
        ]
    issue_update = await Issue.get(issues_id)
    if "listPosition" in update_data:
        list_position = int(update_data.get("listPosition")) or 1
        new_status = update_data.get("status")
        await issue_update.calculate_list_position(issue_update.listPosition, issue_update.status,
                                                   list_position, new_status)
    issue_update = issue_update.copy(update=update_data)
    await issue_update.save()
    return await issue_update.to_schema()
```

Рисунок 2.13 - Endpoints для отримання та оновлення інформації про завдання.Функція 'get_issue' повертає деталі завдання за його ID

Функція 'create_comment' створює новий коментар, 'update_comment' оновлює існуючий коментар за його ID, а 'delete_comment' видаляє коментар за його ID.

При маніпуляції з даними в MongoDB визначені дії, які виконуються автоматично перед або після виконання певних операцій з базою даних, таких як вставка, оновлення або видалення документів. Це включає дії при створенні нового документа, оновленні існуючого або видаленні документа з бази даних (рис. 2.14).

Ця особливість обробки подій дозволяє автоматизувати різні аспекти роботи з даними, що використовується для таких завдань, як автоматичне заповнення полів документа, виконання додаткових перевірок даних або виклик інших частин програми у відповідь на зміни в базі даних. Це додає значну гнучкість до процесу роботи з даними, спрощуючи деякі аспекти розробки програмного забезпечення. Розробники можуть визначити потрібну поведінку на рівні об'єктів Python, які автоматично виконують відповідні операції з базою даних . [6]

```
@before_event(Replace, Update)
def updated_datetime(self):
    self.updatedAt = datetime.utcnow()

@before_event(Insert, Replace, Update, Save)
def set_description_text(self):

    def strip_tags(html):
        return re.sub('<[^<]+?>', '', html)

    description_text = strip_tags(self.description)
    self.descriptionText = description_text

@after_event(Insert)
async def set_issue_link(self):
    await Project.find_one(Project.id == self.project.ref.id) \
        .update(self.push_query())
    if self.users:
        await User.find_many(In(User.id, [user.ref.id for user in self.users])) \
            .update(self.push_query())

@before_event(Delete)
async def del_issue_link(self):
    await Project.find_one(Project.id == self.project.ref.id) \
        .update(self.pull_query())
    if self.users:
        await User.find_many(In(User.id, [user.ref.id for user in self.users])) \
            .update(self.pull_query())
    if self.comments:
        await Comment.find_many(In(Comment.id, [
            comment.ref.id for comment in self.comments
        ])) \
            .delete()
```

Рисунок 2.14 - Методи життєвого циклу для моделі завдання

Методи життєвого циклу автоматично виконуються перед або після певних подій бази даних, таких як вставка, оновлення або видалення. Вони включають встановлення часу оновлення, очищення HTML-тегів з опису, створення та видалення посилань на завдання в проєктах та користувачах.

Цей підхід забезпечує гнучкість та масштабованість системи, дозволяючи користувачам ефективно керувати своїми проєктами та завданнями в межах цих проєктів.

Всі ці функціональні можливості реалізовані за допомогою FastAPI, сучасного високопродуктивного вебфреймворку для Python, який підтримує асинхронні запити та включає систему автоматичної генерації документації (рис. 2.15), що значно спрощує процес розробки та використання API.

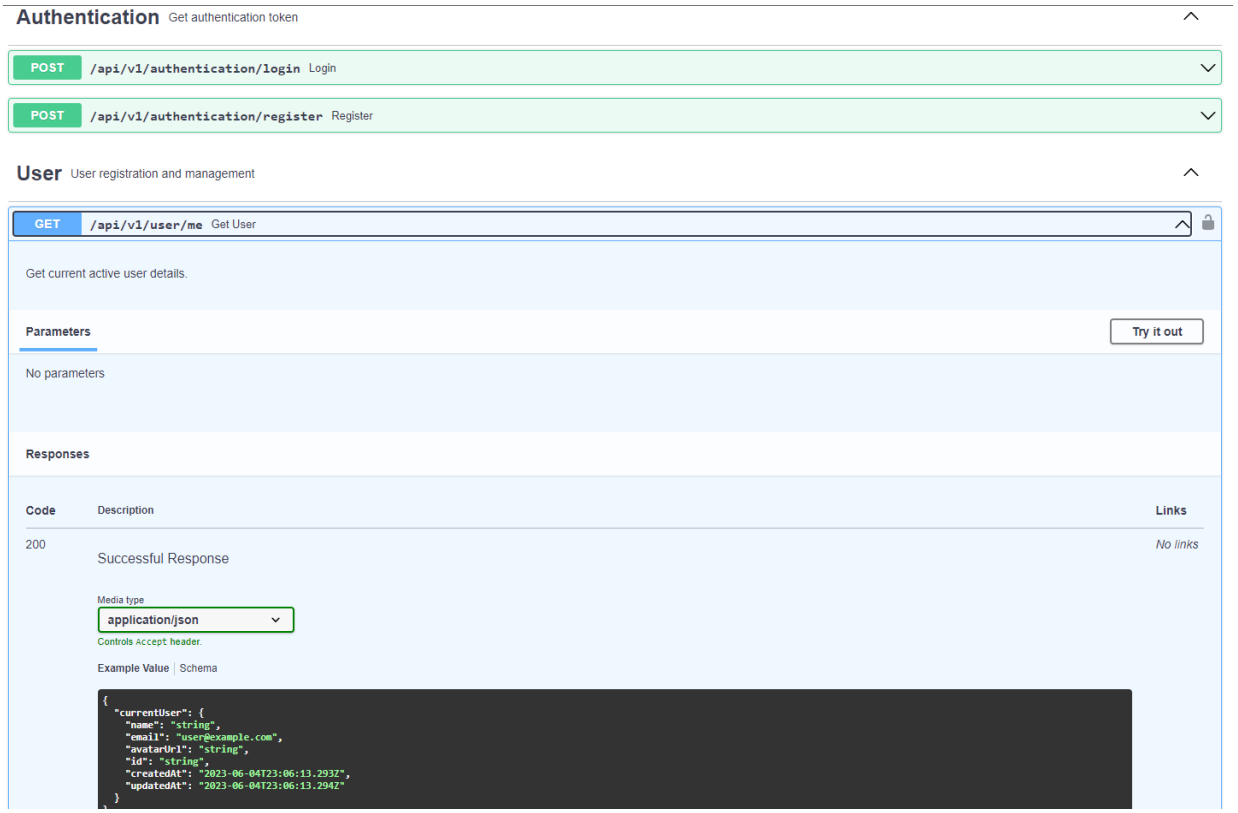

Рисунок 2.15 - Документація API, створена за допомогою OpenAPI. включає в себе схему запитів, відповідей та моделей даних, що використовуються в API

Однією з ключових особливостей програмної реалізації цього інструменту є його компонентна структура, заснована на бібліотеці React.

Кожен аспект вебдодатку поділено на окремі, повторно використовувані компоненти, які можна легко модифікувати та об'єднувати для створення складних інтерфейсів користувача.

Наприклад, файл Routes.js відповідає за маршрутизацію вебдодатку (рис. 2.16). Кожен маршрут (URL) відповідає за відображення певного компонента React, який може включати інші компоненти. Це дозволяє легко розширювати застосунок, додаючи нові маршрути та відповідні компоненти.

```
import React from 'react';
import { Router, Switch, Route, Redirect } from 'react-router-dom';

import history from 'browserHistory';
import Project from 'Project';
import ProjectsList from 'ProjectList';
import LoginForm from 'Project/Authentication/Login';
import Logout from 'Project/Authentication/Logout';
import RegistrationForm from 'Project/Authentication/Register';
import PageError from 'shared/components/PageError';

const Routes = () => (
  <Router history={history}>
    <Switch>
      <Redirect exact from="/" to="/project" />
      <Route path="/authenticate" component={LoginForm} />
      <Route exact path="/register" component={RegistrationForm} />
      <Route path="/logout" component={Logout} />
      <Route path="/project/:projectId" component={Project} />
      <Route path="/project" component={ProjectsList} />
      <Route component={PageError} />
    </Switch>
  </Router>
);

export default Routes;
```

Рисунок 2.16 - Код маршрутизації в файлі Routes.js.

Основний компонент додатку App містить набір підкомпонентів, що використовуються по всьому додатку, таких як NormalizeStyles, BaseStyles, Toast і Routes. Він також імпортує стилі шрифту, які застосовуються в усьому додатку (рис. 2.17).

```
import React, { Fragment } from 'react';

import NormalizeStyles from './NormalizeStyles';
import BaseStyles from './BaseStyles';
import Toast from './Toast';
import Routes from './Routes';

import './fontStyles.css';

const App = () => (
  <Fragment>
    <NormalizeStyles />
    <BaseStyles />
    <Toast />
    <Routes />
  </Fragment>
);

export default App;
```

Рисунок 2.17 - Головний компонент 'App'

Компонент 'App' включає нормалізацію стилів, базові стилі, компонент 'Toast' для відображення повідомлень і маршрутизацію для навігації між сторінками.

2.5. Тестування

Ручне тестування є важливим етапом процесу розробки програмного забезпечення, що забезпечує його якість і надійність. Воно включає перевірку різних аспектів програмного засобу, таких як функціональність, зручність використання, продуктивність тощо. Ручне тестування охопило як бекенд, так і фронтенд частини програмного засобу.

Для бекенду основним методом ручного тестування було використання HTTP-запитів для перевірки різних API кінцевих точок. Завдяки тулкіту FastAPI, документація API автоматично генерується за допомогою Swagger UI, доступного через вебінтерфейс. Це дозволило легко відправляти HTTP-запити

до різних API кінцевих точок і перевіряти їх відповіді. Наприклад, я перевірив правильність витягування, оновлення, видалення та створення завдань, надсилаючи GET-запит на кінцеву точку /issues/{issues_id}, а потім PUT, DELETE та POST-запити до тієї ж кінцевої точки, перевіряючи відповіді на ці запити.

Для фронтенду ручне тестування включало перевірку візуальних елементів інтерфейсу та взаємодію з ними. Я перевірив, чи відображаються всі компоненти коректно, чи реагують вони на взаємодію користувача так, як очікується, та чи взаємодіють з бекендом належним чином. Наприклад, я перевірив компонент маршрутизації в додатку, який визначає, які компоненти відображаються залежно від URL. Цей компонент (Routes) визначає ряд маршрутів, включаючи аутентифікацію, вихід, реєстрацію, список проєктів і конкретний проєкт.

Під час тестування я перевірив наступне:

1. Чи перенаправляється користувач на сторінку проєктів ("/project") при спробі відвідати головну сторінку ("/").

2. Чи відображається форма входу при переході на "/authenticate".

3. Чи відображається форма реєстрації при переході на "/register".

4. Чи працює механізм виходу при переході на "/logout".

5. Чи відображається відповідний проєкт при переході на "/project/:projectId", де ":projectId" - це ідентифікатор конкретного проєкту.

6. Чи відображається список проєктів при переході на "/project".

Я також перевірив взаємодію кожного з цих маршрутів з відповідними компонентами, чи правильно вони викликаються та чи коректно взаємодіють з бекендом. Сценарії включали реєстрацію нового користувача, вхід існуючого користувача, перегляд списку проєктів, перегляд деталей конкретного проєкту, додавання нового проєкту, оновлення існуючого проєкту та видалення проєкту. Я перевірив, чи працюють ці процеси правильно, чи відображають відповідні повідомлення про помилки, коли щось йде не так, і чи оновлюється інтерфейс користувача після виконання дій.

Наприклад, під час тестування форми реєстрації, я перевірив можливість введення даних, валідацію полів, виведення повідомлень про помилки при неправильному введенні даних, успішність реєстрації при правильному введенні даних і відображення нового користувача в системі після реєстрації. Аналогічні сценарії були перевірені для інших функцій додатку, включаючи процеси входу, створення, редагування та видалення проєктів, а також інтерактивні елементи, такі як кнопки та посилання.

Важливо зазначити, що ручне тестування включає не лише перевірку функціоналу, а й відповідність дизайну, стандартам доступності, адаптивність до різних розмірів екрану тощо. Після ручного тестування я дійшов висновку, що фронтенд працює коректно і відповідає всім вимогам до дизайну та функціоналу.

Автоматичне тестування було реалізовано для фронтенду програмного засобу. Проєкт структуровано так, що він містить окремі модулі, які групують схожі функції для спільного тестування. Для цього використовується бібліотека Jest, що дозволяє створювати тести для React компонентів. Кожен модуль має власний набір тестів, які перевіряють його коректну роботу.

Крім того, для автоматичного тестування було використано конвеєр Continuous Integration (CI) та Continuous Deployment (CD). CI/CD автоматизує процеси тестування та розгортання програмного засобу, що підвищує швидкість та ефективність розробки.

Налагодження є важливою частиною процесу розробки програмного забезпечення, яка допомагає знайти та виправити помилки в коді, забезпечуючи правильну роботу програми відповідно до вимог. Для цього існує багато інструментів та методів.

Для налагодження цього проєкту я використовував середовище розробки PyCharm для бекенду та розширення для браузера для фронтенду на React. PyCharm є потужним середовищем для розробки на Python, яке включає вбудовані інструменти налагодження, що дозволяють встановлювати точки

зупинки у коді. Це допомагає призупинити виконання програми і детально проаналізувати її стан у певний момент.[13]

Працюючи над проєктом, я використовував PyCharm для налагодження бекенду, встановлюючи точки зупинки, щоб перевірити правильність обробки даних, виконання запитів до бази даних та функціонування API кінцевих точок. Це дозволило швидко виявляти та виправляти помилки.

Reloadium, плагін для PyCharm, забезпечує функцію гарячого перезавантаження, що дозволяє негайно застосовувати зміни до виконуваного процесу без перезапуску всього додатка. Він також включає інструменти для профілювання пам'яті та таймінгу, що допомагає оптимізувати код і виявляти "вузькі місця".

Для фронтенду я використовував розширення React Developer Tools для браузера, яке допомагає в налагодженні React-додатків. Це розширення дозволяє переглядати стан компонентів React, їх пропси та ієрархію, що допомагає виявляти проблеми з рендерингом або взаємодією компонентів.

Наприклад, під час налагодження я перевіряв правильність відображення списку проєктів за допомогою React Developer Tools, аналізуючи стан компонента ProjectsList та його взаємодію з іншими компонентами. Також я використовував оператор debugger у JavaScript для призупинення виконання коду і покрокового відстеження виконання компонентів, а також метод console.log() для виведення значень змінних, станів та пропсів.

Налагодження допомогло виявити та вирішити проблеми, з якими я стикався під час розробки, забезпечуючи якість коду та правильну роботу програми.

Програмний засіб для онлайн-планування завдань, розроблений на базі фреймворків FastAPI, React та MongoDB, має великий потенціал і широкі можливості для застосування в різних галузях. У цьому розділі наведено рекомендації щодо використання та впровадження програмного засобу. Він може бути корисним у різних сферах діяльності, від освіти до великих

корпорацій. Для оптимального використання важливо адаптувати його до конкретних вимог і потреб кожної організації.

Користувачі програмного засобу мають мати достатній рівень знань та навичок для ефективного використання. Організації повинні забезпечити належну підготовку співробітників для користування цим інструментом.

Необхідно також забезпечити якісну технічну підтримку та обслуговування програмного засобу, щоб уникнути технічних проблем і забезпечити швидке їх вирішення.

Оскільки програмний засіб базується на масштабованих технологіях, організація повинна врахувати можливості його масштабування і адаптації до зростаючих потреб.

З огляду на роботу з чутливими даними користувачів, важливо вжити всі необхідні заходи для захисту цих даних та забезпечення їх інформаційної безпеки.

При правильному впровадженні та використанні цей програмний засіб може значно спростити процес планування завдань, підвищити продуктивність і ефективність роботи.

Висновки: У цьому розділі детально досліджено фреймворки FastAPI, React та MongoDB для розробки проєктів щодо онлайн-планування завдань. Описано постановку задачі, призначення та вимоги до програмного засобу, методологію розробки, загальний опис проєкту та обґрунтування вибору інструментальних засобів.

Основна увага приділялася програмній реалізації та основним режимам функціонування програмного засобу. Також розглянуто питання організації тестування та налагодження програмного засобу.

Розроблено рекомендації щодо використання та впровадження розробленого продукту, що підтверджує ефективність обраних рішень та технологій для досягнення поставленої мети .

ВИСНОВКИ

Під час виконання бакалаврської роботи було проведено детальне дослідження та аналіз фреймворків FARM стеку - FastAPI, React та MongoDB - для створення проєктів з онлайн-планування завдань. Кожен з розглянутих фреймворків має свої унікальні особливості та можливості, які виявилися корисними при розробці інструментів для планування завдань в мережі.

FastAPI виявився ефективним для створення надійних та швидких вебAPI завдяки своїм сучасним, швидким та високопродуктивним можливостям на основі Python. React, як JavaScript бібліотека для створення користувацьких інтерфейсів, дозволив створити інтуїтивно зрозумілі та відповідні інтерфейси користувача. MongoDB, як NoSQL база даних, забезпечила гнучкість у зберіганні даних та швидкість їх обробки.

Використання цих трьох фреймворків разом у FARM стеку дозволило створити ефективні та надійні онлайн-інструменти для планування завдань.

Щодо пропозицій та побажань щодо подальшого дослідження: Хоча FARM стек виявився ефективним для розробки інструментів для планування завдань в мережі, варто розглянути можливість використання інших фреймворків або технологій для порівняння їх ефективності та продуктивності. Зважаючи на швидкий розвиток технологій, важливо постійно відстежувати оновлення та нові можливості цих фреймворків.

Подальше дослідження може включати розробку більш складних проєктів з використанням FARM стеку для глибшого оцінювання його потенціалу та обмежень. Також важливо розглянути питання безпеки при використанні цих фреймворків, особливо при роботі з конфіденційними даними.

Маючи на увазі, що зручність та інтуїтивність інтерфейсу є одними з ключових аспектів успішного планування завдань, подальше дослідження може включати вивчення користувацького досвіду та взаємодії з інструментами, розробленими на основі FARM стеку.

У результаті виконання дипломної роботи досягнуто поставленої мети та розв’язано такі завдання:

1. Досліджено фреймворки FARM стеку для онлайн-планування завдань.. Вивчено такі теми: історія розвитку фреймворків, основні принципи роботи фреймворків, синтаксис фреймворків, компоненти фреймворків, стилі фреймворків, масштабованість фреймворків.

2. Проаналізовано особливості та можливості кожного фреймворку окремо та в контексті їх використання в стеку.

3. Розроблено проєкт щодо онлайн планування завдань із використанням фреймворків Fast API, React, та Mongo DB.

Незважаючи на позитивні результати, виявлені під час дослідження, було також виявлено деякі недоліки та проблеми, зокрема, потребу в глибоких знаннях та досвіді в розробці для ефективного використання цих фреймворків. Один із можливих шляхів вирішення цих проблем - створення детальних інструкцій та керівництва.

Маючи на увазі важливість підтримки та розвитку програмного продукту, подальше дослідження може включати вивчення питань супроводу та розвитку проєктів, розроблених на основі FARM стеку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Архітектура вебдодатків - як вибрати?: стаття з блогу IT-школи Hillel. *Корисні матеріали: Статті та новини IT-індустрії | Комп'ютерна школа Hillel*. URL: https://blog.ithillel.ua/articles/web-application-architecture (дата звернення: 08.06.2024).

2. Agile vs. structured distributed software development: A case study / H.-C. Estler та ін. *Empirical Software Engineering*. 2013. Т. 19, № 5. С. 1197–1224. URL: https://doi.org/10.1007/s10664-013-9271-y (дата звернення: 08.06.2024).

3. Aleksendric M. Full Stack FastAPI, React, and MongoDB: Build Python Web Applications with the FARM Stack. Packt Publishing, Limited, 2024.

4. Angular vs React: Which to Choose for Your Front End in 2024?. *Simform - Product Engineering Company*. URL: https://www.simform.com/blog/angular-vs-react/ (дата звернення: 08.06.2024).

5. Bootstrap. *Bootstrap · The most popular HTML, CSS, and JS library in the world*. URL: https://getbootstrap.com/ (дата звернення: 08.06.2024).

6. Carnes B. Learn the FARM Stack! (FastAPI, React, MongoDB). *freeCodeCamp.org*. URL: https://www.freecodecamp.org/news/learn-the-farm-stack-fastapi-reactjs- mongodb/ (дата звернення: 08.06.2024).

7. Chodorow K. MongoDB: The Definitive Guide. O'Reilly Media, Incorporated, 2013. 432 с.

8. FastAPI. *FastAPI*. URL: https://fastapi.tiangolo.com/ (date of access: 08.06.2024).

9. Introducing FARM Stack - FastAPI, React, and MongoDB | MongoDB. *MongoDB: The Developer Data Platform | MongoDB*. URL: https://www.mongodb.com/developer/languages/python/farm-stack-fastapi- react-mongodb/ (дата звернення: 08.06.2024).

10. JWT.IO. *JSON Web Tokens - jwt.io*. URL: https://jwt.io/ (дата звернення: 08.06.2024).

11. Lalsing V. People Factors in Agile Software Development and

Project Management. *International Journal of Software Engineering & Applications*. 2012. Т. 3, № 1. С. 117–137. URL: https://doi.org/10.5121/ijsea.2012.3109 (дата звернення: 08.06.2024).

12. Manage Your Team’s Projects From Anywhere | Trello. *Manage Your Team’s Projects From Anywhere | Trello*. URL: https://trello.com (дата звернення: 08.06.2024).

13. PyCharm: the Python IDE for Professional Developers by JetBrains. *JetBrains*. URL: https://www.jetbrains.com/pycharm/ (дата звернення: 08.06.2024).

14. Pydantic. *Pydantic*. URL: https://docs.pydantic.dev/ (дата звернення: 08.06.2024).

15. React. *React*. URL: https://reactjs.org/ (дата звернення: 08.06.2024).

16. Reloadium. *Reloadium*. URL: https://reloadium.io/ (дата звернення: 08.06.2024).

17. Sass vs LESS vs Stylus: Вибір препроцесора. *Codeguida*. URL: https://codeguida.com/post/288 (дата звернення: 08.06.2024).

18. Stefanov S. React : up and Running: Building Web Applications. O'Reilly Media, Incorporated, 2021. 222 с.

19. Tailwind CSS - Rapidly build modern websites without ever leaving your HTML. *Tailwind CSS - Rapidly build modern websites without ever leaving your HTML*. URL: https://tailwindcss.com/ (дата звернення: 08.06.2024).

20. What is Ruby on Rails - Ruby on Rails Web Framework Overview 2021 | Railsware Blog. *Blog by Railsware | Blog on Engineering, Product Management, Transparency, Culture and many more...* URL: https://railsware.com/blog/ruby-on-rails-guide/ (дата звернення: 08.06.2024).

21. What is The MEAN Stack? Introduction & Examples. *MongoDB*. URL: https://www.mongodb.com/mean-stack (дата звернення: 08.06.2024).