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

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних інформаційних систем, д.т.н., проф.,
_____Ю.П.Кондратенко
«____»_____2022 року

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

ВЕБЗАСТОСУНОК ДЛЯ МЕНЕДЖМЕНТУ ПРОЕКТІВ У ГАЛУЗІ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Спеціальність 124 «Системний аналіз»

124 – МКР – 607. 21610323

Студент _____В. О. Третяк
«____» лютого 2022 р.
Консультант _____ Е. А. Лисенков
д-р. фіз.-мат. наук, проф
«____» лютого 2022 р.

Миколаїв – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Освітньо-кваліфікаційний рівень **магістр**
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)
Спеціальність **124 «Системний аналіз»**
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____ 20__ р.

З А В Д А Н Н Я
на магістерську кваліфікаційну роботу

Третяку Владиславу Олеговичу

1. Тема магістерської кваліфікаційної роботи «Вебзастосунок для менеджменту проектів у галузі розробки програмного забезпечення».
Керівник роботи Лисенков Едуард Анатолійович, д-р. фіз.-мат. наук, професор.
Затв. наказом Ректора ЧНУ ім. Петра Могили від «__» ____ 20__ р. № _____
2. Строк подання студентом роботи ___ _____ 20__ р.
3. Вхідні (початкові) дані до роботи: загальні відомості про менеджмент проектів та спілкування всередині ІТ-компанії, дані про розробку проекту, їх етапи розробки. Очікуваний результат: вебзастосунок для менеджменту розробки проекту в ІТ-компанії.
4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):
- розкриття теоретичних засад менеджменту розробки у діяльності ІТ-компаній та сучасного стану програмного забезпечення, яке використовується для її підтримки;

- обґрунтуваня вибору технологій і засобів розробки вебзастосунку для забезпечення менеджменту розробки проектів;
- розробка та здійснення програмної реалізації вебзастосунку для забезпечення менеджменту розробки проектів.

5. Перелік графічного матеріалу: презентація, рисунки, таблиці.

6. Завдання до спеціальної частини: Аналіз умов праці та навчання при надзвичайних ситуаціях.

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	д.б.н.,професор Л. І. Григор'єва	
Методична частина	д-р.фіз-мат. наук, професор Е.А. Лисенков	

Керівник роботи __д-р. фіз-мат. наук, проф. Лисенков Е. А._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Третяк В. О._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «_____» _____ 20__ р.

КАЛЕНДАРНИЙ ПЛАН

виконання магістерської кваліфікаційної роботи

Тема: «Вебзастосунок для менеджменту проектів у галузі розробки програмного забезпечення»

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Отримання завдання на магістерську наукову роботу	04.11.2021	11.11.2021	Виконано
2.	Огляд літератури за темою роботи	11.11.2021	22.11.2021	Виконано
3.	Аналіз існуючих ПЗ для забезпечення комунікації та менеджментну розробки	22.11.2021	26.11.2021	Виконано
4.	Аналіз предметної області	01.12.2021	13.12.2021	Виконано
5.	Робота над першим та другим розділом пояснювальної записки	14.12.2021	27.12.2021	Виконано
6.	Створення дизайну, проектування та програмна реалізація вебдодатку	28.12.2021	04.012022	Виконано
7.	Робота над третім розділом пояснювальної записки	04.012022	14.01.2022	Виконано
8.	Тестування системи	18.01.2022	24.01.2022	Виконано
9.	Розробка методичної частини	24.01.2022	25.01.2022	Виконано
10.	Розробка спеціальної частини з безпеки в надзвичайних ситуаціях	25.01.2022	01.02.2022	Виконано
11.	Створення слайдів для захисту та написання доповіді	01.02.2022	15.02.2022	Виконано
12.	Обговорення отриманих результатів з керівником та попередній захист магістерської роботи	28.01.2022	31.01.2022	Виконано
13.	Корегування роботи за результатами попереднього захисту	1.02.2022	15.02.2022	Виконано
14.	Остаточне оформлення пояснювальної записки та слайдів	16.02.2022		Виконано
15.	Захист магістерської роботи	23.02.2022		Виконано

Розробив студент _____Третяк В.О._____ _____

(прізвище та ініціали) (*підпис*)

Керівник роботи __ д-р.фіз-мат. наук, проф. Лисенков Е.А. _____

_____

(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

«____» _____ 202___р.

АНОТАЦІЯ
до магістерської кваліфікаційної роботи роботи

Тема: «Вебзастосунок для менеджменту проектів у галузі розробки програмного забезпечення»

Студент: Третяк Владислав Олегович

Керівник: д-р.фіз-мат. наук професор Лисенков Едуард Анатолійович

Магістерська кваліфікаційна робота присвячена розробці та здійсненню програмної реалізації для менеджменту проектів у галузі розробки програмного забезпечення з функцією чату.

Об'єкт дослідження – інформаційні процеси, моделі менеджменту проектами та інструменти для полегшення не автоматизованих та рутинних задач.

Предмет дослідження – створення логіки, легкої в розширенні, системи для IT компаній, яка допомагає у вирішенні питань з менеджментом проектами та комунікаціями.

Мета дослідження – поєднання функціоналу для команд розробників в IT компаніях шляхом створення онлайн платформи менеджменту у галузі розробки ПЗ , яка надаватиме альтернативу одразу кількома сервісам.

Дипломна робота складається з фахового розділу, методичної і спеціальної частини з охорони праці. Пояснювальна записка дипломної роботи складається зі вступу, трьох розділів, висновків та додатку.

У першому розділі описано предметну сферу, діяльнітсь IT-компаній теоретичні засади життєвого циклу розробки програмного забезпечення, проаналізувано сучасний стан програмного забезпечення, яке використовується для її підтримки та забезпечення.

У другому розділі обґрунтовано вибір технологій і засобів розробки вебзастосунку для менеджменту проектів у галузі розробки програмного забезпечення.

У третьому розділі описано розробку та програмну реалізацію вебзастосунку для менеджменту проектів у галузі розробки програмного забезпечення з функцією вбудованого чату.

У спеціальній частині з охорони праці розглядаються питання аналізу умов праці в офісних приміщеннях.

Дипломна робота містить ___ сторінку (без додатків), ___ рисунків, ___ таблиці, ___ джерел, ___ додаток.

ABSTRACT

for master's scientific work

Subject: “Web-application for communication activities in IT companies”

Student Tretiak Vladyslav Olegovich
Leader: Dr.S., professor Lisenkov Eduard Anatiloivich

Master's thesis is qualifying work is sanctified to development and realization of programmatic realization for the management of projects in industry of software development with the function of chat..

Object of research – informative processes, management models by projects and instruments for the facilitation of the not automated and conservative tasks.

Subject of research – creation of logic easy in expansion, systems for IT of companies, that helps in the decision of questions with a management by projects and communications.

The purpose of the study is combination to the functional for the teams of developers in IT companies by creation on-line of management platform in industry of development of software, that will give alternative at once a few to services.

Diploma work consists of professional division, methodical and special part from a labour protection. The explanatory message of diploma work consists of entry, three divisions, conclusions and to addition.

A subject sphere is described in the first division, діяльнітсь IT-компаній theoretical principles of life cycle of software development, проаналізувано the modern state of software that is used for her support and providing.

In the second division the choice of technologies and facilities of development of вебзастосунку is reasonable for the management of projects in industry of software development.

In the third division development and programmatic realization of вебзастосунку are described for the management of projects in industry of software development with the function of built-in chat.

In the special part from a labour protection

Thesis contains ___ page (without appendices), ___ figures, ___ tables, ___ sources, ___

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему:

«ВЕБЗАСТОСУНОК ДЛЯ МЕНЕДЖМЕНТУ ПРОЕКТІВ У ГАЛУЗІ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

Спеціальність 124 «Системний аналіз»

124 – МКР – 607. 21610323

Студент _____ Третяв В.О.
«__» _____ 20___ р.

Консультант _____ Лисенков Е.А.
д-р. фіз.-мат. наук, проф
«__» _____ 20___ р.

м. Миколаїв – 2022 рік

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

UI – інтерфейс користувача (англ. User Interface)

UX – взаємодія користувача з інтерфейсом (англ. User Experience)

DB – база даних (англ. Data Base)

SPA – односторінковий застосунок (англ. Single-page application)

CC – консольна команда (англ. console command)

DOM – об’єктна модель документу (англ. Document Object Model)

ВСТУП

Актуальність Сучасна галузь розробки програмного забезпечення має проблеми з комунікацією між командами IT спеціалістів, менеджменту моніторингу, та аналітики самого проекту. Водночас, доступ та обмін інформацією у роботі IT спеціалістів, як всередині команди, так і назовні проекту є важливим аспектом реалізації задач, полегшення життя та у цілому збільшення ККД розробників, дизайнерів, менеджерів, тестувальників та замовників. Задачею магістерської роботи є створення вебзастосунку, який буде поєднувати в собі функції Agile дошок, перегляду коду та месенджеру, таким чином вебзастосунок дозволить полегшити комунікацію за допомогою інтеграції чату та системи контролю версіями, яка наддасть можливість швидко переглядати зміни за номером комміта, планувати релізи та спринти, переглядати Canban обо Scrum boards, відсліджувати та редагувати задачі, писати комментарі, надавати права доступу до складових проекту на рівні менеджменту або матеріалів, можливість звітувати та писати документацію, переглядати проектні беклоги.

Інтенсивний розвиток у сфери IT та сучасні тенденції створюють передумови для визначення пріоритетів при розробці ПО та стандартів керування проектами. Налагодженні командні взаємовідносини є одним із найважливіших аспектів у роботі компанії в досягненні поставлених задач.

Кількість інформації, яка збільшується з кожним днем, робить сферу інформаційних послуг досить популярною, створюючи нові можливості для IT компаній.

Актуальність теми магістерської роботи полягає у тому, що сфера інформаційних послуг активно розвивається, IT компанії завжди потребують налагодження автоматизації роботи, яка повторюється, через це існує велика необхідність у створенні продукту, який покривав широкий функціонал, що необхідний для управління та комунікації при розробці.

Метою дослідження є поєднання функціоналу для команд розробників в IT компаніях шляхом створення онлайн платформи менеджменту у галузі розробки ПЗ , яка надаватиме альтернативу одразу кількома сервісам.

Відповідно до мети, виділені наступні завдання магістерської роботи:

Аналіз сфери компаній у галузі IT розробки.

Архітектура та математичні моделі для вирішення задачі створення вебзастосунку.

Програмна реалізація вебзастосунку. Серверна та клієнтська частина вебзастосунку.

Об’єктом дослідження є інформаційні процеси, моделі менеджменту проектами та інструменти для полегшення не автоматизованих та рутинних задач.

Предметом дослідження є створення логіки, легкої в розширенні, системи для IT компаній, яка допомагає у вирішенні питань з менеджментом проектами та комунікаціями.

Наукова новизна одержаних результатів дослідження полягає у тому, що автором: запропоновано та обґрунтовано напрями вдосконалення автоматизації процесів менеджменту проектів у галузі розробки програмного забезпечення, спрямовані на забезпечення їх ефективної взаємодії.

Результати дослідження обговорювалися на Всеукраїнській науково-практичній конференції молодих вчених, аспірантів і студентів «Інтелектуальні інформаційні системи» та отримали схвалення.

Практичне значення отриманих результатів полягає в тому, що сформульовані теоретичні положення та практичні рекомендації щодо підвищення ефективності менеджменту проектів у галузі розробки програмного забезпечення шляхом впровадження розробленого вебзастосунку.

Структура магістерської роботи. Відповідно до мети, завдань і предмета дослідження, магістерська робота містить основну, методичну та спеціальну частини. Основна частина магістерської роботи складається із вступу, трьох розділів, висновку, списку використаних джерел та __ додатків. Загальний обсяг

магістерської роботи – __ сторінок, із них основного тексту основної частини – __ сторінок, методичної частини – __ сторінок, спеціальної – __ сторінок. Кількість використаних джерел – __.

РОЗДІЛ 1. АНАЛІЗ СФЕРИ ІТ-КОМПАНІЙ. ПОСТАНОВКА ЗАДАЧІ

1.1. Опис предметної сфери

Мета проекту є створення вебзастосунку для менеджменту проектів у галузі розробки програмного забезпечення з функціями перегляду Agile дошки, чату та організації часу. На даний момент більшість ІТ компаній використовують системи, що допомагають внутрішнім командам проводити спілкування, менеджерам збирати аналітику об’єму задач та допомагати ефективно використовувати кожного члена команди для полегшення роботи кожної із інших внутрішніх команд на проекті, відстежувати статус завдань та давати зворотній фідбек замовникам.

Модель життєвого циклу програмного забезпечення — це спрощене уявлення процесу програмного забезпечення. Кожна модель життєвого циклу представляє процес з певної точки зору і, таким чином, надає лише часткову інформацію про цей процес. Ці загальні моделі не є остаточним описом програмних процесів. Скоріше, це абстракції процесу, які можна використовувати для пояснення різних підходів до розробки програмного забезпечення. Ви можете розглядати їх як рамки процесів, які можна розширити та адаптувати для створення більш конкретних процесів розробки програмного забезпечення.

Моделі життєвого циклу програмного забезпечення не є взаємовиключними і часто використовуються разом, особливо для розробки великих систем. Для великих систем має сенс поєднати деякі з найкращих функцій водоспаду та моделей з інкрементальним життєвим циклом.

Реальні програмні процеси – це перемежовані послідовності технічних, спільних та управлінських заходів із загальною метою визначення, проектування, впровадження та тестування програмної системи. Розробники програмного забезпечення використовують у своїй роботі різноманітні програмні засоби. Інструменти особливо корисні для підтримки редагування різних типів документів і для управління величезним обсягом детальної інформації, яка створюється у

великому програмному проекті. Нематеріальна та пластична природа програмного забезпечення дозволяє використовувати різноманітні моделі життєвого циклу розробки програмного забезпечення. Вони варіюються від лінійних моделей до ітераційних і гнучких моделей. У лінійних моделях етапи розробки програмного забезпечення виконуються послідовно із зворотним зв'язком та ітераціями за потребою, після чого слід інтеграція, тестування та доставка окремого продукту.

В ітераційних моделях програмне забезпечення розробляється з кроком збільшення функціональності на ітераційних циклах.

У гнучких моделях розробка програмного забезпечення зазвичай передбачає часті демонстрації працюючого програмного забезпечення клієнту або представнику користувача, який керує розробкою програмного забезпечення за короткі ітераційні цикли, які створюють невелике збільшення робочого програмного забезпечення. Інкрементні, ітераційні та гнучкі моделі можуть доставити ранні підмножини працюючого програмного забезпечення в середовище користувача, якщо це потрібно. Якщо модель життєвого циклу програмного забезпечення вибрана правильно відповідно до вимог проекту, розробка проекту йде швидко та продуктивно. Ви завжди повинні враховувати природу вашого продукту. Вам слід проаналізувати, чи працюєте ви над проектом із чіткими вимогами, які ніколи не зміняться, чи ви все ще шукаєте свого клієнта, чи, можливо, ви працюєте над застарілим проектом, де потрібна лише підтримка — усі ці деталі мають значення. Одна модель може полегшити повсякденну рутину, а інша – надмірно ускладнити її і сповільнити.

Життєвий цикл розробки програмного забезпечення – це застосування стандартних бізнес-практик для створення програмних додатків. Зазвичай він поділяється на шість-вісім кроків: планування, вимоги, проектування, збірка, документування, тестування, розгортання, обслуговування. Деякі менеджери проектів поєднують, розділяють або пропускають кроки, залежно від обсягу проекту. Це основні компоненти, рекомендовані для всіх проектів розробки програмного забезпечення.

SDLC – це спосіб вимірювання та покращення процесу розробки. Це дозволяє провести дрібнозернистий аналіз кожного етапу процесу. Це, у свою чергу, допомагає компаніям максимізувати ефективність на кожному етапі. Зі збільшенням обчислювальної потужності це висуває вимоги до програмного забезпечення та розробників. Компанії повинні знижувати витрати, швидше постачати програмне забезпечення та задовольняти або перевищувати потреби своїх клієнтів. SDLC допомагає досягти цих цілей, виявляючи неефективність та вищі витрати та виправляючи їх, щоб вони працювали стабільно. Життєвий цикл розробки програмного забезпечення просто описує кожне завдання, необхідне для складання програмного додатка. Це допомагає зменшити відходи та підвищити ефективність процесу розробки. Моніторинг також гарантує, що проект продовжує виконуватися, і продовжує залишатися вигідною інвестицією для компанії.

На етапі планування керівники проекту оцінюють умови проекту. Це включає розрахунок трудових і матеріальних витрат, створення графіка з цільовими цілями, а також створення команд і керівної структури проекту. Планування також може включати зворотний зв’язок із зацікавленими сторонами. Зацікавлені сторони – це всі, хто може отримати вигоду від програми. Спробуйте отримати зворотній зв’язок від потенційних клієнтів, розробників, експертів у тематиці та торгових представників .Планування повинно чітко визначати сферу та мету застосування. Він накреслює курс і забезпечує команду для ефективного створення програмного забезпечення. Він також встановлює межі, щоб допомогти проекту не розширюватися або не змінювати його початкову мету.

Визначення вимог вважається частиною планування, щоб визначити, що повинна виконувати програма та її вимоги. Наприклад, програма соціальних мереж вимагала б можливості зв’язатися з другом. Для програми інвентаризації може знадобитися функція пошуку. Вимоги також включають визначення ресурсів, необхідних для побудови проекту. Наприклад, команда може розробити програмне забезпечення для керування спеціальною виробничою машиною. Машина є обов’язковою умовою процесу. Управління проектами - це дисципліна, що об'єднує

спеціальні і над професійні знання. Спеціальні знання відбивають особливості тієї сфера діяльності, в якій належати проекти (будівельні, Інноваційні, екологічні, дослідні, організаційні тощо). Це випливе з таких особливо проектної діяльності: значний період від качана реалізації проекту до його завершеною; Великої кількості учасників; складного характеру проектної діяльності, що ставити сукупність простих, елементарної форми (технічної, наукової, комерційної, виробничої, будівельної, фінансової і т.п.).

Управління проектами - це процес управління командою, ресурсами проекту за допомогою спеціальних методів і прийомів з метою успішного досягнення поставленої мети. Фаза проектування моделює спосіб роботи програмного додатка. Деякі аспекти дизайну включають:

Архітектура – визначає мову програмування, галузеві практики, загальний дизайн та використання будь-яких шаблонів або шаблонів

Інтерфейс користувача – визначає, як клієнти взаємодіють із програмним забезпеченням, і як програмне забезпечення реагує на введення

Платформи – визначає платформи, на яких працюватиме програмне забезпечення, наприклад Apple, Android, версія Windows, Linux або навіть ігрові консолі

Програмування – не тільки мова програмування, але й методи вирішення проблем і виконання завдань у програмі

Комунікації – визначає методи, якими програма може спілкуватися з іншими активами, такими як центральний сервер або інші екземпляри програми

Безпека – визначає заходи, вжиті для захисту програми, і може включати шифрування трафіку SSL, захист паролем і безпечне зберігання облікових даних користувача.

Прототипування може бути частиною етапу проектування. Прототип схожий на одну з ранніх версій програмного забезпечення в моделі ітеративної розробки програмного забезпечення. Він демонструє основне уявлення про те, як виглядає і працює додаток. Цей «практичний» дизайн можна продемонструвати зацікавленим

сторонам. Використовуйте відгуки, щоб покращити програму. Змінити фазу прототипу дешевше, ніж переписати код, щоб внести зміни на етапі розробки.

Конкуренція додатків для чату жорстка. Більшість нових додатків для обміну повідомленнями не набувають широкого поширення, не в змозі поєднати надійну продуктивність у масштабі з привабливим, захоплюючим користувацьким досвідом, необхідним для того, щоб програма стала вірусною. Командам, які готуються запустити новий продукт на такий переповнений ринок, потрібно всебічне розуміння сильних і слабких сторін своїх конкурентів, перш ніж вони почнуть визначати пріоритети функцій, приймати технічні рішення або розподіляти інженерні ресурси. Уважне вивчення того, що наведені нижче найпопулярніші програми для чату роблять правильно і де вони не мають успіху має дати цінну інформацію, яка допоможе створити власну програму для чату.

Невеликий проект може бути написаний одним розробником, тоді як великий проект може бути розбитий і над ним працюють кілька команд. На цьому етапі використовуйте програму керування доступом або керування вихідним кодом. Ці системи допомагають розробникам відстежувати зміни в коді. Вони також допомагають забезпечити сумісність між різними командними проектами та забезпечити досягнення цільових цілей.

Процес кодування включає багато інших завдань. Багатьом розробникам потрібно оновити навички або працювати в команді. Знайти та виправити помилки та збої дуже важливо. Завдання часто гальмують процес розробки, наприклад, очікування результатів тестування або компіляція коду для запуску програми. SDLC може передбачити ці затримки, щоб розробники могли виконувати інші обов’язки.

Розробники програмного забезпечення цінують інструкції та пояснення. Документація може бути формальним процесом, включаючи підключення посібника користувача для програми. Це також може бути неформальним, наприклад коментарі у вихідному коді, які пояснюють, чому розробник використав певну процедуру. Навіть компанії, які прагнуть створювати легке та інтуїтивно

зрозуміле програмне забезпечення, отримують переваги від документації.цілей, організувавши деплоймент на боці серверів IT-компанії, яка буде використовувати мережу.

Документація може бути швидким оглядом основних функцій програми, які відображаються під час першого запуску. Це можуть бути відеоуроки для виконання складних завдань. Письмова документація, як-от посібники користувача, посібники з усунення несправностей та поширені запитання, допомагають користувачам вирішувати проблеми або технічні.

Чати надають можливість спілкуватися великій групі людей в реальному часі. Зручне використання переговорів по необхідним питанням, для тих членів проекту, які зайняті в одному процесі [1].

Чат покриває наступні функції:

1. Обговорювання загальних питань та проблем, які вимагають термінового вирішення.

2. За допомогою чату можна нотифікувати групу учасників про подію або швидко дізнатися необхідну інформацію.

3. Соціалізація. Аспект який актуальний для більшості компаній, а саме з віддаленими співробітниками чи декількома офісами. За допомогою чату співробітники компанії мають відчуття причетності до команди і відчувають участь в робочому процесі. Вирішення загальних питань, не пов’язаних з роботою, покращує емоційний стан колективу.

Для підвищення ефективності чатів існують правила для користування чатом, яких рекомендують дотримуватися. Дотримання порад допоможе підняти ефективність спілкування:

1. Використовувати чат виключно в робочих цілях для отримання швидких відповідей, не ініціюйте в них обговорення глобальних тем.

2. При відлученні працівника треба змінити статус, так як, чат передбачає таку можливість.

3. Дотримуватися суті розмови. В гіршому випадку, обговоривши робочий момент, можна закінчити темою, яка не стосується робочого процессу.

4. В разі обговорення задачі з колегою в чаті, рекомендовано перенести її в окрему задачу.

1.2. Переваги чату, вбудованого в систему управління

Вважається багатьма золотим стандартом для додатків однорангового обміну повідомленнями, WhatsApp став повсюдно поширеним у 2010-х роках як альтернатива SMS-повідомленням. Передаючи повідомлення через Інтернет замість мобільних даних, WhatsApp допомагає користувачам уникнути перевищення даних. WhatsApp також був одним з перших мобільних додатків для чату, які пропонували наскрізне шифрування (E2EE), забезпечуючи захист повідомлень, щоб їх не міг прочитати ніхто, крім відправника та одержувача — навіть WhatsApp. Поєднання безкоштовної передачі в Інтернеті та E2EE робить WhatsApp надзвичайно популярним для спілкування з друзями та родиною в різних країнах.

Хоча WhatsApp створив свою репутацію завдяки захищеному обміну повідомленнями, його придбання Facebook у 2014 році змусило деяких користувачів поставити під сумнів цілісність програми, коли мова заходить про захист їхніх особистих даних та вмісту повідомлень. У пошуках більшої конфіденційності та спокою ці користувачі звертаються до менш відомих безпечних програм обміну повідомленнями, таких як Signal і Telegram. Незалежно від того, чи правдиві звинувачення щодо компрометації WhatsApp його цінностей, зміна у сприйнятті користувачів створює можливість для конкурентів, які хочуть отримати частку ринку цього гіганта WhatsApp використовує Extensible Messaging and Presence Protocol (XMPP), одну з найстаріших і найбільш поширених платформ для обміну миттєвими повідомленнями через Інтернет. Варто уточнити, як працює

XMPP, як частина вашого дослідження, щоб вирішити, які фреймворки забезпечуватимуть вашу власну програму чату.

1. Підтримка різних платформ для Android, iOS, Mac, Windows та Інтернету
2. Наскрізне шифрування (E2EE).
3. Групи (канали) до 256 користувачів із настроюваними сповіщеннями.
4. Текстові, фото-, відео- та звукові повідомлення.
5. Голосові та відеодзвінки через Інтернет.
6. Спільний доступ до фотографій із сховища пристрою або безпосередньо за допомогою вбудованої програми камери.
7. Обмін файлами для PDF-файлів, документів, електронних таблиць, слайд-шоу тощо, до 100 МБ.
8. Підтримка платежів, подібних до Venmo, у деяких країнах.

Вбудований додаток для чату від Apple додає значну цінність до його асортименту пропозицій пристроїв, а знайомство та гнучкість iMessage роблять його стандартним для більшості користувачів iPhone. Можливість плавно перемикатися між обміном повідомленнями через Інтернет і SMS означає, що користувачі можуть надсилати один одному повідомлення практично з будь-якого місця, не замислюючись про те, чи є найбільш доцільним на даний момент Wi-Fi, мобільні дані чи SMS. Інтерфейс iMessage є чистим та інтуїтивно зрозумілим у використанні, з функціональними можливостями, розробленими, щоб максимально використовувати сенсорний екран смартфона, а не просто адаптуватися до нього. Користувачі перетягують повідомлення ліворуч, щоб побачити позначки часу, наприклад, натисніть і утримуйте повідомлення, щоб отримати доступ до реакцій, і перетягніть розмову вліво, щоб вимкнути або видалити її.

Завдяки постійним інвестиціям Apple вдається підтримувати iMessage знайомим і свіжим водночас. Користувачі грайливо відкривають нові функції, іноді випадково, а іноді, коли бачать, що їхні друзі починають ними користуватися, в той

час як основні синьо-білі бульбашки повідомлень залишаються заспокійливо послідовними. Ця здатність збалансувати старе з новим — не кажучи вже про те, що воно попередньо встановлено на пристроях Apple — робить iMessage особливо жорстким конкурентом для тих, хто хоче створити власний додаток для чату. Тим не менш, уважне вивчення відкриває кілька можливостей для інших конкурувати.

Один із найочевидніших способів об’єднати iMessage – це підтримка пристроїв Android та Windows, що забезпечує постійну роботу для всіх користувачів, незалежно від їхньої участі в екосистемі пристроїв Apple.На сьогоднішній день досвід обміну повідомленнями між Android та iOS у кращому випадку заслуговує уваги, і користувачі iPhone, які мають комп’ютери з ОС Windows або Chromebook, не можуть насолоджуватися перевагами синхронізації iMessage між пристроями. Іншою можливістю для покращення, принаймні з лютого 2021 року, може бути більш чиста та інтуїтивно зрозуміла функція потоків повідомлень — ми раді, що Apple нарешті додала потоки, але ми чуємо, що багато користувачів вважають поточну версію громіздкою у використанні та приголомшливо дивитися.

1. Синхронізація між пристроями між iOS на iPhone, iPad та Apple Watch і macOS на комп’ютері.
2. Індикатори друку.
3. Групові повідомлення.
4. Підтримка SMS.
5. Додаткові квитанції про прочитання.
6. Реакції.
7. Аудіоповідомлення.
8. Передача тексту.
9. Легкий обмін фотографіями з камери або безпосередньо з інтегрованої програми камери.
10. Інтеграція програм з Apple Pay, Stickers, Apple Music та іншими популярними програмами.

Завдяки Slack концепція командного чату революціонізувала спосіб співпраці працівників знань. Більш спонтанний і гнучкіший, ніж електронна пошта, Slack дозволяє колегам стосуватися проектів, ділитися жартами та мемами та іншим. Притаманний Slack фактор веселощів допоміг розширити додаток у 2010-х роках, і додаток відіграв вирішальну роль у підтримці масового переходу 2020 року до віддаленої роботи.

У ряді способів випадок використання командного чату представляє складніші проблеми розробки порівняно з простішими програмами для обміну повідомленнями один на один. Тим, хто хоче створити клон Slack, потрібно буде підтримувати тематичний формат каналу, який став популярним через Slack, а також індивідуальні прямі повідомлення та спеціальні групові повідомлення. Додаток також потрібно буде спроектувати для багатоквартирного користування, щоб середовище чату кожного клієнта відгородилося від інших.

Командний чат світового класу, як-от Slack, також вимагає ряду розширених функцій, без яких інші програми можуть обходитися. Вони включають глибоку інтеграцію з популярними програмами, такими як Google Calendar, Google Drive і, звісно, Giphy, а також команди косої лінії та можливість використовувати Slack як інтерфейс для розмови для керування іншими програмами. Згадки, реакції, смайли та гарні чисті ланцюжки розмов — усі разом, щоб надати користувачам Slack витончений професійний досвід.

1. Підтримка різних платформ для Android, iOS, Інтернету та комп’ютера.
2. Груповий обмін повідомленнями, канали та прямий обмін повідомленнями.
3. Підтримка численних зовнішніх інтеграцій додатків.
4. Слеш команди.
5. Нитки розмов.
6. Реакції.
7. Згадки.

8. Вкладення файлів.
9. Аудіо та відеодзвінки.
10. Slackbot розмовний AI.
11. Автоматичні нагадування.
12. Закріплені повідомлення.
13. Збережені повідомлення.

Колись Facebook Messenger був простою та надзвичайно популярною функцією чату в додатку, невіддільною від попередніх версій Facebook.com лише для Інтернету. Але цілеспрямовані зусилля Facebook відокремити Facebook Messenger, який був запущений як самостійний додаток у 2011 році, створили більш потужну та динамічну платформу, яка зараз конкурує з іншими лідерами в цій галузі. Перехід на окрему програму Messenger був завершений у 2014 році, коли гігант соціальних мереж змусив користувачів або завантажити Messenger, або взагалі відмовитися від чату в додатку. Як і інші зміни від Facebook, ця зміна спочатку засмутила деяких користувачів, але згодом більшість з них погодилися.

Розширене впровадження Facebook Messenger здебільшого не пов’язане зі створенням найсучаснішого багатофункціонального додатка. Натомість Facebook використовував свою популярність, щоб впливати на те, коли, як і чому люди звертаються до Facebook Messenger, розширюючи варіанти цільового використання за межі випадкового спілкування з друзями. Приклади розширеного використання включають обмін повідомленнями з продавцями на Facebook Marketplace, обмін повідомленнями з місцевими підприємствами щодо продажів і запитів про обслуговування клієнтів, а тепер можливість централізації DM та Facebook в одному додатку.

Messenger отримує переваги від домінування Facebook над світом соціальних мереж — він знайомий, простий у використанні, і він просто там, глибоко інтегрований з Facebook і автоматично синхронізований з вашим списком друзів у Facebook. Але, з іншого боку, це означає, що Messenger не застрахований від підводних каменів Facebook. Оскільки користувачі та уряди в усьому світі

продовжують уважно вивчати політику конфіденційності Facebook, її обробку даних користувачів та зміну її ролі в поширенні знань та інформації, можуть виникнути можливості для конкурентів додатків для чату, які пропонують такий же чудовий продукт з меншим багажем та меншими умовами.

1. «Кімнати» для групових відеодзвінків.
2. Настроювані реакції за допомогою будь-яких смайлів.
3. Ефекти AR, ефекти повідомлень і наклейки для селфі.
4. Користувачі можуть вибрати власні теми та кольори інтерфейсу користувача.
5. Інтеграція з Facebook, WhatsApp, Instagram і Portal.
6. Додаткові push-повідомлення.
7. Відповіді в ланцюжках.
8. Додаткова аутентифікація відбитків пальців пристрою.
9. Підтримка платежів за допомогою кредитної картки або PayPal (лише для США).
10. Продажі та служба підтримки клієнтів спілкуються з компаніями.
11. Можливість надсилати повідомлення друзям в Instagram з Messenger.
12. Додаткова двофакторна аутентифікація (2FA) і наскрізне шифрування (E2EE).

1.3. Постановка задачі

Основна вимога для проекту – створення вебзастосунку для менеджменту проектів. Найкращі програми для обміну повідомленнями забезпечують цілісний досвід, настільки простий та інтуїтивно зрозумілий, що кінцеві користувачі не обов’язково помічають окремі функції. Якщо інші типи програм використовують кілька сторінок і навігаційних меню, чудові програми чату зазвичай просто

відкривають клавіатуру під вікном повідомлення, тому кожна функція виглядає як невід'ємна частина цілого.

Це представляє цікаву проблему для розробників і груп продуктів, які працюють над реверсією та відтворенням успішного додатка для чату: легко бути надмірно впевненим, поспішати на етапі дослідження та почати кодування, не враховуючи всі окремі компоненти, які будуть потрібні. Ви не хочете, щоб у вашому чаті було кілька спринтів, а потім зрозуміли, що ви не включили простий спосіб інтегрувати якусь важливу функцію в майбутньому.

Сучасні програми чату мають бути доступними для користувачів на різних типах пристроїв і операційних системах. Сюди входять як мінімум iOS, Android та Інтернет, хоча настільні програми для Mac і Windows також можуть бути корисними. Створити чат для однієї платформи досить просто, але час, вартість і технічні проблеми збільшуються, коли вам потрібен послідовний досвід на всіх цих платформах. Кожна платформа створює свої унікальні перешкоди і вимагає різного набору навичок розробника. Вам потрібно буде однаково інвестувати в розробку кожної платформи, інакше ви ризикуєте отримати невтішні й суперечливі враження.

Попередньо створені рішення API можуть допомогти вирішити цю проблему, надаючи міжплатформні пакети SDK для чату, які дають змогу досягати успіху без найму спеціального експерта на кожній платформі. Це означає, що команди інженерів можуть використовувати мову за своїм вибором, як-от JavaScript, Python, Ruby, Go, Swift або PHP, замість того, щоб виконувати свої налаштування.

Оскільки громадськість стає більш уважною до питань конфіденційності та безпеки, інтерес до програм для чату, які проактивно вирішують ці проблеми (і ефективно повідомляють про свої зусилля), зростатиме. Ми вже бачили, як висвітлення в новинах сумнівних методів конфіденційності відштовхнуло деяких користувачів від основних програм, таких як WhatsApp, на менші чатові платформи, не пов'язані з Big Tech, як-от Signal, Telegram і Threema. І якщо ваш додаток для чату перетинається з галуззю, яка регулярно має справу з

конфіденційною інформацією, як-от обробка платежів або охорона здоров'я, вона повинна відповідати вимогам відповідності. Широкі рамки, такі як GDPR, ISO 27001 і SOC 2, є універсальними, а галузеві рамки, як-от PCI DSS і HIPAA, додають додаткові вимоги до безпеки.

Як мінімум, усі дані потрібно буде зашифрувати під час передачі та в стані спокою. Справжнє наскрізне шифрування, хоча офіційно не обов'язкове, є хорошою ідеєю. Крім того, відповідні заходи керування ідентифікацією, такі як багатофакторна аутентифікація (MFA), складність пароля та вимоги до ротації, забезпечують розшифрування приватних повідомлень лише потрібним людям.

Вимоги до основного функціоналу:

1. Можливість реалізації Agile boards.
2. Можливість реалізації Scrum boards.
3. Можливість реалізації Canban boards.
4. Створення дошок.
5. Планування спринтів.
6. Трекінг задач.
7. Моніторинг процесу розробки.
8. Можливість додавати коментарі та посилання до задач.
9. Розділення прав на проектному рівні.
10. Беклоги проекту.
11. Інтеграція з CI / CD інструментами.
12. Інтеграція чату для команди, нотифікації.

Платформа, на якій працює команда може змінюватися від проекту або вимог клієнта, прийнято рішення створити веб-додаток та додати підтримку всіх браузерів.

Для реалізації вибрано Single Page Application. Варіант, в якому використовуються бекенд і фронтенд. За допомогою їх взаємодії можна створити додаток, який буде працювати зовсім без перезавантажень сторінки в браузері, це дасть змогу написати код, який буде швидко працювати.

Створення інтерактивних user інтерфейсів користувачів, незалежно від платформи розробки, безсумнівно складне завдання. Для розробки кожного компонента веб-сайту потрібні невтомні зусилля, повна відданість і належна концентрація. Однак React JS — це бібліотека JavaScript, яка покращує створення інтерфейсів користувача та покращує життя розробників. Він уже став масовим і використовується рядом великих імен, включаючи Facebook, Netflix, AirBNB, DropBox, IMDb, PayPal, Tesla Motors, Walmart та багато інших. Створені компоненти можуть бути з легкістю використані повторно. Повторне використання коду підвищує покриття тестами, що, надає більш високий рівень контролю якості [4].

ReactJS це фреймворк JavaScript, який значно полегшив процес розробки. Крім того, він надає якісні програми ReactJS із відповідними інтерфейсами. ReactJS вважається дуже затребуваною альтернативою для розробки зручних і дуже привабливих веб-сайтів і додатків, що надає розробникам ряд можливостей зробити їх більш креативними.Компонентне-орієнтований підхід, можливість з змінювати наявні компоненти і перевикористати код. Компоненти, які були створені під час роботи над тим чи іншим проектом, не мають залежностей. Таким чином, ніщо не заважає використовувати їх знову і знову в проектах різного типу.

Висновок до розділу 1

У цьому розділі аналізується сфера діяльності IT-компанії, процес комунікації в IT-компанії та основні етапи управління проектами. Поставлені завдання та вимоги до платформи. Результати розкривають основні концепції управління проектами, інструменти та кроки для побудови проекту від ідеї до чату, управління завданнями та командою в різних умовах, а також написання документів та звітів. На ринку IT-менеджерам потрібна система, яка забезпечує можливість реалізації основних концепцій, автоматизації етапів концепції, управління стратегічними ресурсами, побудови та аналізу роботи кожного

спеціаліста та забезпечення покращення комунікації та командної роботи. Розроблено чіткий план для автоматизації повсякденних завдань управління проектами та спілкування в команді в єдину систему якості, яка може задовольнити потреби менеджерів, бізнес-аналітиків та розробників.

Система допоможе швидко планувати завдання, спринти, релізи; матиме можливість створювати дошки Scrum або Canban; матиме можливість відстежувати завдання та створювати коментарі; писати звіти та технічні документи; надавати доступ на рівні проекту, покращувати за допомогою інтеграції комунікації послуги та GIT Communication та перегляд коду. Обрано односторінковий додаток (SPA або односторінковий додаток) для типу вебзастосунку. Варіанти використання бекенда та інтерфейсу. Завдяки їх взаємодії можна створити програму, яка працює без перезавантаження сторінки в браузері. Або в спрощеному варіанті, коли переходи між розділами призводять до перезавантаження, вони не потрібні для будь-яких операцій у розділах. Це дозволить написати бізнес-логіку, яка може працювати швидко.

РОЗДІЛ 2. МАТЕМАТИЧНІ МОДЕЛІ ТА АРХІТЕКТУРА ДЛЯ ВИРІШЕННЯ ЗАДАЧІ СТВОРЕННЯ ВЕБЗАСТОСУНКУ ДЛЯ МЕНЕДЖМЕНТУ ПРОЕКТІВ ТА ОРГАНІЗАЦІЇ ЧАСУ

2.1. Огляд на менеджмент проектами

Програми та проекти дуже важливі для багатьох комерційних і державних організацій, завдяки проектам компанії можуть збільшувати прибуток, покращувати процес створення концепцій продуктів, їх розробку та впровадження на ринок, сприяти створенню нових або вдосконалених засобів виробництва, інформації. системи. Проекти є інструментами розвитку та вдосконалення організацій на всіх рівнях: міста, області, країни. За допомогою програм і проектів навчальні та інші заклади впроваджують нові та покращені послуги, які вони вже надають.

Робочий процес будь-якої середньої IT-компанії сильно відрізняється від роботи великої промислової компанії. Управління проектами в таких компаніях здійснюють один або кілька керівників проектів. Кожне замовлення випускається у вигляді нового проекту з чітко визначеними фазами, термінами, результатами та періодичною зарплатою працівника. Проектами керують і контролюються менеджери. Помилка в одному проекті може призвести до втрати прибутку для багатьох інших пов’язаних проектів. Важливо, щоб кожна організація, відповідальна за проект, могла ефективно керувати ним. Імовірність отримання значних результатів від реалізації IT-проекту визначає важливість ролі попередньої оцінки при виборі проекту.

Процеси управління проектами необхідно стандартизувати, а документи, які формалізують ці процеси, називають методами управління проектами. Є спеціальне програмне забезпечення для управління проектами - це системи управління проектами, в тому числі для створення завдань, створення команд, підключення до сервісів співпраці, інтеграції пошти; забезпечення онлайн-

середовища для робочої взаємодії в локальних і розподілених командах, що дозволяє планувати проекти, розставляти пріоритети завдань і відстежуйте їх розклад; співпрацюйте з колегами за допомогою спільних інструментів редагування файлів і відстежуйте прогрес і оновлення кожного завдання команди.

Метою даної роботи є вивчення підвищення продуктивності та зниження ризиків шляхом планування та обліку роботи в автоматизованій системі управління, створення математичної моделі для оптимізації складу команди.

2.2. Побудова математичної моделі

Модель — це сурогатний об'єкт вихідного об'єкта, який забезпечує вивчення деяких властивостей вихідного об'єкта. Використання об’єктів моделі для заміни одного об’єкта іншим для отримання інформації про найважливіші властивості вихідного об’єкта називається моделюванням.

Математичне моделювання буде розуміти процес побудови математичного об’єкта, який відповідає заданому реальному об’єкту, відомий як математична модель, і вивчення цієї моделі для отримання характеристик реального об’єкта. Тип математичної моделі залежить від природи реального об'єкта і завдання об'єкта, а також надійності й точності, необхідних для вирішення проблеми.

В даний час існує багато способів розрахунку вартості створення IT-проекту з урахуванням різних видів ресурсів: технічних, адміністративних, трудових. Може бути створена система підтримки прийняття рішень для міжпроектного переведення співробітників до IT-компаній.

Використання розробленої моделі в задачі розподілу чи перерозподілу працівників на типових проектах дозволяє знизити навантаження на керівників і прискорити процес прийняття рішень. На заданій роботі розглядається задача побудови оптимізаційної математичної моделі управління трудовими ресурсами. врахувати їх кваліфікацію. Щоб вчасно досягти бажаних результатів за певних

грошових витрат або інших значних ресурсів, необхідно керувати якісним проектом розвитку команди.

Правильно сформована команда розробників може бути найбільш продуктивною. Давайте створимо математичну модель, яка буде використовуватися на сторінці «Дошка» програми, тобто створимо дошку для опису завдань у проекті. Постановка проблеми: Є кілька співробітників - розробників і тестувальників (QA-інженер), яких слід розділити в майбутніх проектах. Робота вважається неоднорідною і її кількість вимірюється годинами. Кожне завдання має термін виконання. Завдання поділяються на 2 види: критичні та другорядні. Необхідно побудувати математичну модель на основі історії кожного учасника системи для оптимізації складу команди з урахуванням різних пріоритетів завдань. Розглянуто два оптимальні критерії: час виконання завдання та вартість (вартість заробітної плати). Розглянемо завдання типу critical та введемо позначення:

t_{dt}— час роботи розробника d ($d = 1..m$) над завданнями / під задачами $i(i = 1...k)$ зазначеного типу;

$T_{est(i)}$ – оцінне час виконання завдання / під задачі;

$X_{qi} = \sum_{j=1}^{k} T_{est(j)} / \sum_{j=1}^{m} t_{qj}$ – коефіцієнт ефективності кожного з розробників;

t_{qt} – час роботи QA-engineer q ($q = 1..m$) над завданнями / під задачами j $(j = 1..k)$ зазначеного типу;

$X_{qi} = \sum_{j=1}^{k} T_{est(j)} / \sum_{j=1}^{m} t_{qj}$ - коефіцієнт ефективності кожного з QA-engineer. Величини $t_{dt}, t_{qt}, T_{est(i)}$ відносяться до раніше виконаним завданням типу critical.

Отримана математична модель являє собою задачу нелінійного програмування з оптимальним критерієм і стандартним методом розв’язання. Попередні розрахунки на наборі тестових даних демонструють ефективність

запропонованого методу. Таким чином, вартість кожного другорядного завдання може бути мінімізована, а час на розробку критичних завдань можна зменшити.

Щоб зменшити витрати, ви можете спробувати відмовитися від певних ресурсів під час виконання певної роботи. Але слід мати на увазі, що це збільшує навантаження на інших учасників проекту.

Це, у свою чергу, може призвести до зміни часу виконання завдань або зниження якості виконання. Першим кроком є зниження вартості тих завдань, які є найменш пріоритетними і найменш важливими для проекту. Деякі з них можна просто видалити з проекту. Недоліком такого підходу є погіршення якості реалізації проекту. Цикл реалізації проекту також збільшується, якщо видалені завдання знаходяться на критичному шляху. Дотримуючись цього алгоритму, ви можете автоматизувати процес призначення завдань співробітникам і оптимізувати витрати для вашої IT-компанії. Запропонований підхід можна розширити, розглянувши інші типи ресурсів, що використовуються в IT-проектах. Математичні моделі розподілу трудових ресурсів реалізовані в комплексній розробці програмного забезпечення та методології управління IT-проектами, прецедентних діаграм бізнес-процесів (рис 2.1).

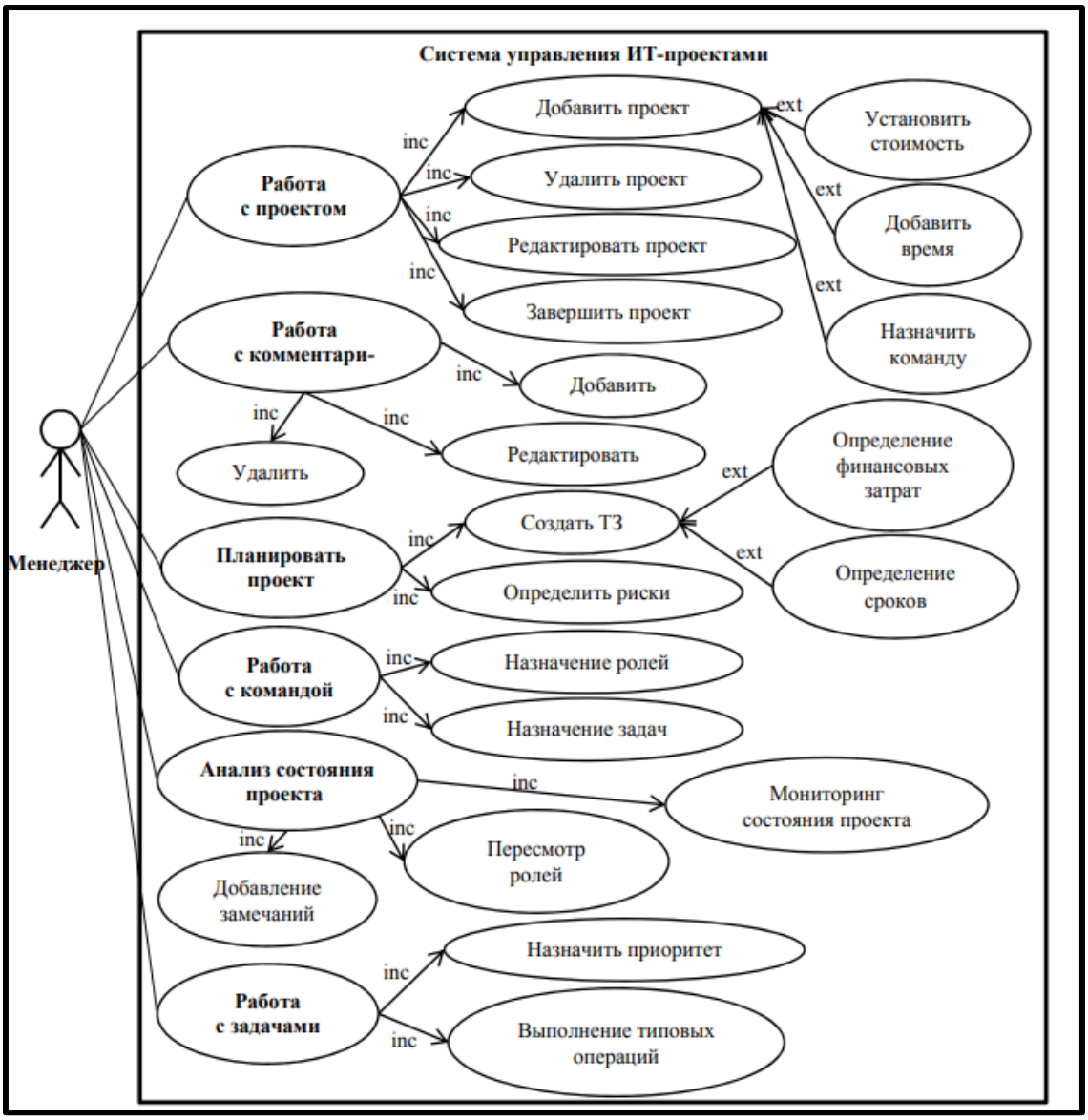

Рис. 2.1. Схема прецедентів бізнес-процесу

2.3. Поняття SPA та архітектура

Моделі в SPA обробляють рівень даних проекту, вони відповідають за створення API виклику, який отримує дані, і заповнення моделі відповіддю JSON, оскільки React не має офіційного способу створення моделі, я вирішив використати Основний JS. Моделі в SPA обробляють рівень даних проекту, як і будь-яка інша програма, вони відповідають за створення викликів API для отримання даних і заповнення моделей відповідями JSON, оскільки React не має офіційного способу

визначення моделей, я вирішив використовувати Backbone. JS для цього. Це також позбавляє мене від необхідності ідентифікувати джерело даних. BackboneJS має два класи для ефективного управління даними, модель і збір. Яка кінцева мета SPA? Щоб надати інтерфейс користувача.

Зрештою, це просто більш складний шар. Якщо ви робите великі обчислення на стороні клієнта, має сенс мати окремий контролер. Але для більшості додатків єдиною метою є забезпечення гладкого інтерфейсу, і щоб виконати це завдання, відправник наполегливо працює. Однак у випадку з React доцільніше виключити контролери, оскільки кожен компонент має власну технологію візуалізації та життєвого циклу. Це майже охоплює всі необхідні розрахунки. Для маршрутизації ми будемо використовувати React Router [5]. Але перед цим давайте поговоримо про URL-адреси. URL-адреса в SPA є ознакою конкретного стану програми, тобто разом з маршрутами та параметрами програму можна завантажити в потрібному стані.

В інтерактивному SPA нормально мати кілька станів на сторінці, і ви хочете отримати доступ до кількох станів вашої програми через URL-адреси, але це означало б брудні схеми URL-адрес, тому краще збалансувати їх. Тепер, коли ви знаєте основи використання React, давайте подивимося на кілька місць. Ми збираємося створити просту односторінкову програму (також відому як SPA) за допомогою React. Ці програми відрізняються від традиційних багатосторінкових програм, які ви бачите скрізь. Найбільша відмінність полягає в тому, що навігація по сторінці не передбачає створення нової сторінки. Натомість ваша сторінка (у цьому контексті часто називається представленням) зазвичай завантажує вбудований рядок у самій сторінці. Найскладніше – переконатися, що односторінковий додаток веде себе так, як це підходить вашим користувачам. Точніше, коли користувачі націлюються на ваш додаток, вони очікують:

1. URL-адреса, що відображається в адресному рядку, завжди відображає те, що вони переглядають.
2. Використання кнопки назад.

3. Можливість переходу до певного виду за допомогою відповідної URL-адреси.

З додатками на кількох сторінках виходять ці три речі. Нічого зайвого не буває. У програмі з однією сторінкою, оскільки ви не переходите на нову сторінку, вам доведеться справді обробляти ці три речі, щоб змусити своїх користувачів працювати. Вам потрібно переконатися, що навігація у вашій програмі коректує URL-адресу правильно.

Вам потрібно переконатися, що історія вашого веб-переглядача належним чином синхронізована з кожною навігацією, щоб користувачі могли використовувати кнопку вперед і назад. Якщо користувач додає закладки до певного представлення даних або копіює/вставляє URL-адресу для подальшого доступу, вам потрібно переконатися, що ваша односторінкова програма переміщує користувача в правильне місце.

Щоб зробити URL-адреси зрозумілішими, я вирішив зберегти їх в окремому файлі (і не використовувати динамічне посилання) для знайомого відчуття MVC. Маршрути, визначені React Router, є просто тегами JSX, і компоненти, які передаються як параметри (рис. 2.2) [6].

```
<Router>
  <div>
    <Route exact path="/search/:keyword" component={ListPage}/>
    <Route exact path="/property/:id" component={SinglePage}/>
  </div>
</Router>
```

Рис. 2.2. Маршрутизація в React

Щоб впоратися з усім цим, існує техніка, яка називається маршрутизацією. Маршрут – це коли ви намагаєтеся зіставити URL-адресу з цільовою сторінкою, яка не є фізичною сторінкою, наприклад, один перегляд у односторонній програмі. Звучить складно, але існує багато бібліотек JavaScript, які можуть допомогти нам у цьому.

Однією з таких бібліотек є JavaScript, React Router. React Router забезпечує функціональність маршрутизації для односторінкових програм, вбудованих в React, що робить його веселим і простим [7]. Анімація переходу під час маршрутизації. Отже, ми маємо ефективну систему маршрутизації.

Тепер давайте анімуємо перехід. Для цього ми будемо використовувати модуль react-transition-group. Ми додаємо анімацію для встановленого стану кожного компонента. Коли ви використовуєте компонент Route всередині Switch для перемикання між компонентами, ви, по суті, встановлюєте та видаляєте відповідні компоненти.

Ми збираємося використовувати react-transition-group в кожному компоненті, який ми хочемо анімувати. Таким чином ви можете використовувати різну анімацію встановлення для кожного компонента.

Я буду використовувати лише один тип анімації для кожного. Оскільки ми маємо справу зі станом змонтованого компонента, необхідно підключити transformAppear і встановити для нього часовий інтервал.

Ви також повинні вимкнути transitionEnter і transitionLeave, оскільки вони працюють лише після встановлення компонента. Вам потрібно буде використовувати ці властивості, якщо ви плануєте анімувати дочірні елементи компонента. Нарешті, додайте унікальне ім’я transitionName, на яке можна посилатися у файлах CSS (рис. 2.3) [8].

```
import React from 'react'
import { CSSTransitionGroup } from 'react-transition-group'
import '../styles/homeStyle.css'


const Home = () =>{
  return(
  <CSSTransitionGroup
      transitionName="homeTransition"
      transitionAppear={true}
      transitionAppearTimeout={500}
      transitionEnter={false}
      transitionLeave={false}>
      <div>
        Home
      </div>
  </CSSTransitionGroup>
  )
}

export default Home
```

Рис. 2.3. Додаємо атрибути для анімації

Також, було імпортовано файл CSS, в якому и визначено CSS переходи (рис. 2.4).

```
.homeTransition-appear{
  opacity: 0;
}

.homeTransition-appear.homeTransition-appear-active{
  opacity: 1;
  transition: all .5s ease-in-out;
}
```

Рис. 2.4. Стилі для анімації

Якщо ви оновите сторінку, вам доведеться побачити повільний ефект компонента Home. Якщо ви робите те ж саме з іншими компонентами маршрутизації, ви зможете побачити їх власні анімації під час навігації по меню. Адресування компонентів. Є два різні варіанти адресації: HashRouter і BrowserRouter. Як випливає з назви, HashRouter використовує хеші для опису ваших посилань — стиль адресації, який підходить для статичних серверів. З

іншого боку, якщо ви використовуєте динамічний сервер, кращим вибором буде BrowserRouter, оскільки з ним ваші URL-адреси виглядатимуть краще. Отже, остаточна версія вашого файлу index.js має виглядати так (рис. 2.5):

```
.homeTransition-appear{
  opacity: 0;
}

.homeTransition-appear.homeTransition-appear-active{
  opacity: 1;
  transition: all .5s ease-in-out;
}
```

Рис. 2.4. Використання HashRouter

Якщо ви використовуєте динамічний сервер і віддаєте перевагу BrowserRouter, єдина різниця – імпортувати BrowserRouter і використовувати його для обгортання компонента <App>. Обертаючи наш компонент <App>, ми передаємо об’єкт історії до нашої програми, щоб інші компоненти реакційного маршрутизатора могли взаємодіяти один з одним [9].

2.4. Поняття Thunk та Redux

Redux пропонує розглядати програму як початковий стан, модифікований серією дій, що, на мою думку, є дуже хорошим підходом для складних веб-додатків, і це відкриває багато можливостей. Redux став однією з найпопулярніших реалізацій керування потоками даних Flux у додатках React.

Однак у процесі дослідження Redux часто виникає ситуація, коли «для дерев нема лісу». Ось простий і продуманий підхід до програм, які використовують Redux. Redux - це більше, ніж просто бібліотека. Це цілісна екосистема. Однією з причин його популярності є здатність писати код з використанням різних шаблонів і підходів проектування. Наприклад, якщо мені потрібно виконати деякі асинхронні операції, чи варто використовувати санки? А може, обіцянка? Або сага?

Який метод правильний? Однозначної і однозначної відповіді немає. І немає «кращого» способу використання Redux. Зрозуміло, широкий підхід зайшов у глухий кут.

Я хочу показати моє особисте користування бібліотекою. Очевидно, що його можна застосувати до різних життєвих сценаріїв, а головне, його легко навчитися. Оскільки ми використовуємо React, почнемо зі створення програми React – офіційного стартового шаблону. Також налаштуйте redux, react-redux і redux-thunk. Використовуйте p у класичному Flux для зберігання програм стану. Відправлення (перехід) дії викликає зміну цього стану. Далі потрібно повторно відобразити подання на основі зміненого стану (рис. 2.6).

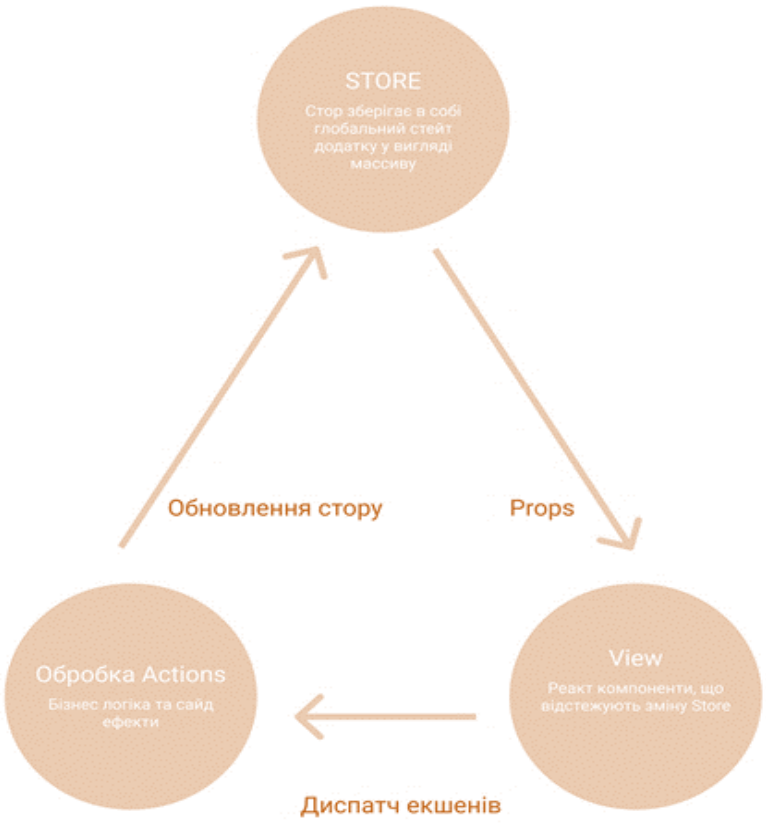

Рис. 2.6. Схема зміни State

Flux спрощує розробку, створюючи односпрямований потік даних. Це зменшує ефект спагетті в міру зростання кодової бази програми. Однією з труднощів у розумінні того, як працює Redux, є багато неочевидних термінів, таких як директор, селектор, санки тощо.

Для більш чіткого розуміння розглянемо розширений цикл Flux (рис. 2.7). Можливо, ви помітили, що інші інструменти Redux, такі як проміжне програмне забезпечення або саги, не відображаються. Це зроблено навмисне, ці інструменти не відіграють значної ролі в нашому додатку.

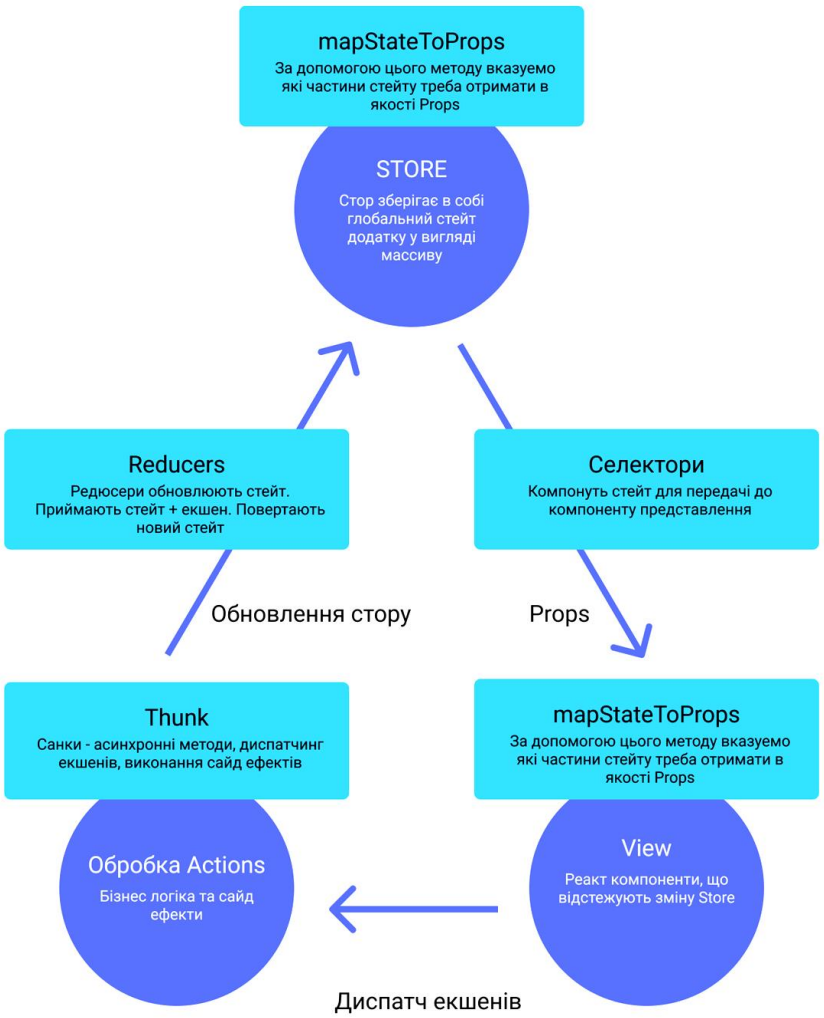

Рис. 2.7. Схема пояснення Redux-інструментів

Селектори один з найголовніших інструментів Redux, про який зазвичай забувають. Селектор - це чиста функція, приймаюча як аргумент глобальний стейт і повертає його в перетвореному вигляді. Селектори тісно пов'язані з редюсерамі і розташовані всередині reducer.js. Селектори дозволяють нам провести деякі розрахунки за даними, перш ніж дані потраплять в уявлення. В майбутньому ми

скористаємося цим прийомом. Кожен раз, коли нам необхідно отримати частину State (наприклад в mapStateToProps), ми повинні використовувати селектори. Чому? Ідея полягає в тому, щоб інкапсулювати внутрішній State програми та перемістити його від уявлення. Уявіть, що пізніше ми вирішили змінити внутрішню структуру. Без селектор нам довелося б вносити зміни в кожен компонент уявлення, що небажано. Використання селектор дозволить проводити рефакторинг, змінюючи тільки редьюсер [10].

Операції Redux за замовчуванням є синхронними, що може бути проблемою для програм, яким потрібно взаємодіяти з API сервера або виконувати інші асинхронні операції. На щастя, Redux надає нам щось на кшталт проміжного програмного забезпечення, яке є десь між ломбардом дії та директором. У Redux є дві найпопулярніші бібліотеки проміжного програмного забезпечення асинхронних дій, Redux Thunk і Redux Saga. Redux Thunk — це бібліотека проміжного програмного забезпечення, яка дозволяє викликати творців дії, повертаючи функцію замість об’єкта. Функція приймає метод диспетчеризації як параметр, щоб його можна було використовувати в тілі функції для диспетчеризації звичайних синхронних операцій після завершення асинхронної операції. Зазвичай Redux-Thunk використовується для асинхронних запитів до зовнішніх API для отримання або збереження даних. Redux-Thunk дозволяє легко відправляти дії, які слідують за «життєвим циклом» запиту, до зовнішнього API [11].Перед початком необхідно додати пакети до нашого проекту (рис. 2.8).

```
$ yarn add redux-thunk

$ npm install redux-thunk
```

Рис. 2.8. Встановлення Redux-Thunk

Необхідно додати middleware, коли буде створений store додатку, за допомогою applyMiddleware, що забезпечує Redux (рис. 2.9).

```
import React from 'react';
import ReactDOM from 'react-dom';
import { createStore, applyMiddleware } from 'redux';
import { Provider } from 'react-redux';
import thunk from 'redux-thunk';

import rootReducer from './reducers';
import App from './App';


const store = createStore(rootReducer, applyMiddleware(thunk));

ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>,
  document.getElementById('root')
);
```

Рис. 2.9. Встановлення Redux-Thunk

Проміжне програмне забезпечення — це запропонований спосіб розширити Redux за допомогою спеціальних функцій. Посереднє програмне забезпечення дозволяє розгорнути способи відправки вашого магазину для задоволення та прибутку. Ключовою особливістю проміжного програмного забезпечення є те, що воно є складеним.

Декілька проміжних програм можна згрупувати разом, коли кожному проміжному програмному забезпеченню не потрібно знати, що відбулося до або після нього в ланцюжку. Наприклад, у нас є проста програма Todo. Коли ми натискаємо «Додати завдання», зазвичай спочатку диспетчується дія, яка сповіщає про початок додавання нового завдання.

Якщо сервер успішно створив і повернув елемент todo, з нашим новим елементом завдання буде відправлена інша операція, і операція успішно завершується. Якщо сервер з якоїсь причини повертає помилку, не додавайте нову дію диспетчеризації завдання з помилкою дії не завершено (рис. 2.10).

```
import { connect } from 'react-redux';
import { addTodo } from '../actions';
import NewTodo from '../components/NewTodo';

const mapDispatchToProps = dispatch => {
  return {
    onAddTodo: todo => {
      dispatch(addTodo(toto));
    }
  };
};

export default connect(
  null,
  mapDispatchToProps
)(NewTodo);
```

Рис. 2.10. Приклад використання Redux-Thunk

У самому дійстві веселіше. Тут ми будемо використовувати бібліотеку Axios для запиту ajax (рис. 2.11).

```
# Yarn
$ yarn add axios

# npm
$ npm install axios --save
```

Рис. 2.11. Приклад використання Redux-Thunk

У функції тіла ми спочатку плануємо звичайну операцію синхронізації, яка повідомляє про те, що ми почали додавати нові елементи завдань за допомогою зовнішнього API. Простіше кажучи - запит відправляється на сервер. Тоді ми фактично використовуємо Axios, щоб зробити запит POST до сервера.

Якщо буде позитивна відповідь від сервера, ми заплануємо операцію синхронізації, використовуючи дані, отримані від сервера. Але якщо з сервером сталася помилка, ми надішлемо іншу операцію синхронізації з повідомленням про помилку. Ми впроваджуємо метод, який буде штучно затримувати розвиток (рис. 2.12).

```
export const addTodo = ({ title, userId }) => {
  return dispatch => {
    dispatch(addTodoStarted());

    axios
      .post(ENDPOINT, {
        title,
        userId,
        completed: false
      })
      .then(res => {
        setTimeout(() => {
          dispatch(addTodoSuccess(res.data));
        }, 2500);
      })
      .catch(err => {
        dispatch(addTodoFailure(err.message));
      });
  };
};
```

Рис. 2.12. Приклад штучної затримки при розробці

Редактор визначає, як змінюється стан програми у відповідь на дії, надіслані на стор. Параметр mapStateToProps — це функція, яка повертає звичайний об’єкт або іншу функцію. Передайте цей параметр connect(), щоб підписати контейнерний компонент на оновлення репозиторію Redux.

Це означає, що функція mapStateToProps викликається щоразу, коли змінюється стан сховища. Якщо вас не цікавить відстеження оновлень стану, передайте connect() як значення undefined або null для цього параметра. Функція mapStateToProps оголошується з двома параметрами, другий параметр необов’язковий. Першим параметром є поточний стан сховища Redux. Другий параметр, якщо він переданий, є об’єктом властивостей, переданих компоненту (рис. 2.13).

```
const mapStateToProps = function(state) {
  return {
    profile: state.user.profile,
    loggedIn: state.auth.loggedIn
  }
}

export default connect(mapStateToProps)(ProfileComponent);
```

Рис. 2.13. Функція mapStateToProps

Якщо з mapStateToProps повертається простий об’єкт, повернутий об’єкт stateProps поєднується з властивостями компонента. Ви можете отримати доступ до цих властивостей компонента таким чином: Якщо mapStateToProps повертає функцію, ця функція буде використовуватися як mapStateToProps для кожного екземпляра компонента. Це корисно для покращення продуктивності візуалізації та покращення пам’яті (рис. 2.14).

```
function ProfileComponent(props) {
  return (
    props.loggedIn
      ? <Profile profile={props.profile} />
      : <div>Please login to view profile.</div>
  )
}
```

Рис. 2.14. Доступ до властивостей компонента

Використовуючи функцію mapDispatchToProps як параметр, програміст повинен подбати про повернення об’єкта dispatchProps, який зв’язує генератор дій за допомогою методу репозиторію dispatch(). Функція приймає метод зберігання dispatch() як перший параметр. Як і mapStateToProps, ця функція також може приймати додатковий другий параметр ownProps, який описує зіставлення з вихідними властивостями, переданими компоненту.

Якщо ця функція повертає іншу функцію, функція, що повертається, використовується як mapDispatchToProps, що корисно для візуалізації та продуктивності пам’яті. Допоміжну функцію Redux bindActionCreators() можна використовувати всередині цієї функції, щоб прив’язати творців дій до методу репозиторію dispatch(). [12].

2.5. Figma як інструмент компонування макету. Структура додатку

Щоб зрозуміти різні етапи розробки, систему необхідно розділити на окремі блоки (сторінки), які можуть визначити нашу бізнес-логіку та полегшити процес розробки та класифікації кожного функціонального модуля.

Попередній дизайн буде створено шляхом створення макета Figma. Що таке Figma? Це графічний редактор для веб-дизайну. У Figma ви можете створити:

1. інтерактивні прототипи для вебсайтів і мобільних додатків;
2. елементи інтерфейсу - значки, кнопки, меню, вікна, форми зворотного зв'язку;
3. вектори.

У Figma всі документи зберігаються в хмарі. Завдяки цьому ви можете разом працювати над макетами в редакторі та відкривати їх за посиланнями без завантаження. Ви можете отримати доступ до Figma через свій браузер або завантажити програму на свій комп’ютер.

Він доступний для Windows і Mac. У настільній версії ви можете працювати в автономному режимі, а зміни синхронізуються під час доступу до Інтернету [13]. Вихідний текст документа зберігається в хмарі.

Немає необхідності завантажувати макети, хмарити та контролювати версії. Якщо дизайнер поспішно припустився помилки, порушив логіку відступів або розмістив різний розмір шрифту в h1 на різних сторінках одного проекту, верстальщик, швидше за все, повторить помилку і опублікує веб-сайт з нею.

Тому фахівці виділяють час на підготовку дизайн-проектів для верстки. Важко зробити його ідеальним з самого початку і не вимагає остаточної збірки, оскільки процес може мати багато редагування, заміни вмісту, налаштувань веб-розробниками та інших незручностей. Коли дизайнер не встигає налаштувати свій продукт для наступного етапу, він передає оригінальну модель або переробляє її

вночі – жоден варіант не найкращий. Для полегшення компіляції параметри, які потрібно перевірити, перераховані перед остаточним поданням проекту.

Процес розкладки може бути полегшений:

1. При створенні файлу встановіть в редакторі колірну схему RGB. Зазвичай це стандартно, але варто перевірити цей варіант.
2. Використовуйте сітки або стовпці, щоб вирівняти розділи та дотримуватися логіки відступів.
3. Збережіть співвідношення зображення при зміні розміру. Для цього просто утримуйте Shift.
4. Використовуйте зображення шрифтів, а не псевдотіла.
5. Якщо об’єкти повинні взаємодіяти один з одним, ви можете створити анімований прототип, щоб продемонструвати всю інтерактивність.

Перед початком роботи над проектом необхідно вказати потрібний формат готового файлу. Це важливо, оскільки перетворення файлів з Figma в Illustrator – не найцікавіше завдання. Щоб спростити використання макета, потрібно створити структуру на панелі «Шари»: видалити пробіли та приховати, згрупувати, написати назви, які відображають вміст.

Для іменування використовується латиниця. Якщо в макеті є приховані шари, деякі програми беруть з них інформацію, що дозволяє вирівнювати блоки тексту та інші елементи по краях прихованої форми. Це ускладнює роботу з відступом і спотворює поля групи при виборі групи. Коли експерт видаляє прихований шар, домени та межі всіх вкладених об’єктів зближуються.

Важко недооцінити тему шрифтів, оскільки вона відкриває так багато можливостей для помилок. Тонкощі починаються з вибору: є список «безпечних шрифтів», які ви можете завантажити безкоштовно — вони найбільш передбачувані та прості у використанні. Крім того, є деякі нерозкриті шрифти. Їх додавання може призвести до блокування безкоштовних облікових записів або цензури авторських прав. Звичайною практикою стало «дряпання» листів із платних наборів – теж погана дорога, оскільки перемальовки рідко бувають різної

якості, але все ж не найбільш законного вмісту. Не забувайте, що багато стилів – не найкраща ідея. Зазвичай достатньо двох-трьох варіантів, використовуючи жирний та інші режими відображення. Важливо визначити кілька стилів тексту для подібних ситуацій. Суміжні абзаци можуть не відрізнятися за розміром шрифту, якщо не незначно. Ці недоліки особливо яскраво виражені у великих веб-проектах з великою кількістю сторінок різного змісту [14].

Висновок до розділу 2

У цьому розділі представлені математична модель та архітектура для вирішення проблеми створення корпоративного чату з можливостями перегляду коду, управління проектами та управління часом.

У дослідженні аналізуються можливості управління людськими ресурсами з урахуванням їхньої кваліфікації при розподілі проектів (завдань). Побудовано математичну модель оптимізації розподілу трудових ресурсів для двох типів завдань. Запропонована модель дозволяє розглянути особливості проекту та його учасників. Використання розробленої моделі в задачі розподілу чи перерозподілу працівників на типових проектах дозволяє знизити навантаження на керівників і прискорити процес прийняття рішень.

Моделі для оптимізації складу команд IT-проектів можна використовувати для зниження ризику, плануючи заздалегідь.

Вибрані основні інструменти для створення архітектур, які дозволять вам легко розгортати, повторно використовувати компоненти та легко керувати веб-додатками загального стану.

Redux пропонує розглядати програму як початковий стан, модифікований серією дій, що, на мою думку, є дуже хорошим підходом для складних веб-додатків, і це відкриває багато можливостей. Одним із принципів хорошої методології є поділ ідей і бізнес-логіки. Де зараз реалізована наша бізнес-логіка?

Вся бізнес-логіка знаходиться в папці src/store/. Більшість із них реалізовано як салазки в actions.js, деякі реалізовані як селектори в редуктор.js. Насправді це має на увазі правило: вся бізнес-логіка повинна бути в обробниках подій (санках), селекторах і редукторах. Redux пропонує широке поле для експериментів.

Створення веб-сервісу ділиться на кілька етапів: збір інформації, дизайн креслення, написання контенту, компонування та програмування. Розробники починають виконувати свої обов'язки лише тоді, коли з'являються готові дизайнерські проекти.

Для максимального спрощення макет являє собою плоске зображення, яке показує майбутні сторінки сайту. Зображення «оживляє» експерт, який за допомогою коду описує зовнішній вигляд веб-сайту – гіпертекстової розмітки та каскадних таблиць стилів. Тому фахівці виділяють час на підготовку дизайн-проектів для верстки.

Важко зробити його ідеальним з самого початку і не вимагає остаточної збірки, оскільки процес може мати багато редагування, заміни вмісту, налаштувань веб-розробниками та інших незручностей. Коли дизайнер не встигає налаштувати свій продукт для наступного етапу, він передає оригінальну модель або переробляє її вночі – жоден варіант не найкращий.

Для полегшення компіляції параметри, які потрібно перевірити, перераховані перед остаточним поданням проекту. Потрібно звернути увагу на розмір ваших зображень – великі, важкі фотографії можуть уповільнити роботу веб-сайтів, тому вони обробляються на спеціальному сервісі стиснення, що дозволяє зберегти якість. При створенні простих зображень рекомендується використовувати векторну графіку, оскільки вона зберігає свої властивості при збільшенні або зменшенні масштабу та не ускладнює веб-ресурси.

Варто пам'ятати, що, враховуючи наведену вище інформацію, веб-дизайнер знижує шанси на помилку, а також демонструє свій професіоналізм та вміння цінувати час своїх колег. На перший погляд список рекомендацій виглядає досить великим, але після виконання кількох проектів його легко запам'ятати.

Проаналізувати потреби у створенні проекту, розробити базову структуру проекту, розбити функціонал на окремі блоки, що відповідають сторінкам програми. Окремі компоненти кожної сторінки описані окремо. На основі цього був створений дизайн майбутнього сервісу за допомогою інструменту Figma.

РОЗДІЛ 3. СТВОРЕННЯ МАКЕТУ. ПРОГРАМНА РЕАЛІЗАЦІЯ СЕРВЕРНОЇ ТА КЛІЄНТСЬКОЇ ЧАСТИНИ ВЕБ-ЗАСТОСУНКУ

3.1. Створення проекту

Ми використовуємо інтегрований набір інструментів для підвищення зручності користувачів і розробників. Розглянемо популярні інструменти React, які допомагають виконувати такі завдання:

1. Масштабування файлів і компонентів.
2. Використання npm бібліотек.
3. Виявлення поширених помилок на ранньому етапі.
4. Спостереження редагування CSS і JS в розробці.
5. Оптимізація коду.

Якщо у вас немає перерахованих вище проблем або ви відчуваєте незручність при використанні інструментів JavaScript, подумайте про те, щоб додати React як простий тег <script> на вашу сторінку HTML, додавши підтримку JSX, якщо потрібно. Це також найпростіший спосіб інтегрувати React в існуючий веб-сайт. Тим часом ви завжди можете додати новий набір інструментів, якщо вважаєте це потрібним!

Розробники React в основному рекомендують наступні рішення:

1. Якщо ви вивчаєте React або створюєте нову односторінкову програму, використовуйте Create React App.

2. Якщо ви використовуєте Node.js для створення веб-сайту для відтворення, спробуйте Next.js.

3. Якщо ви створюєте статичний веб-сайт, орієнтований на вміст, спробуйте Gatsby.

4. Якщо ви створюєте бібліотеку компонентів або інтегруєтеся з наявною кодовою базою, спробуйте більш гнучкий набір інструментів.

Create React App — це зручне середовище для вивчення React і найкращий спосіб розпочати створення нових односторінкових програм React. Він може налаштувати ваше середовище розробки, щоб ви могли використовувати найновіші функції JavaScript, забезпечуючи легку розробку та оптимізовані виробничі програми. Вам знадобиться Node >= 6 і npm >= 5.2. Щоб створити проект, запустіть:

```
npx create-react-app my-app
cd my-app
npm start
```

Рис. 3.1. Команди створення проекту

Create React App не виконує жодної логіки або бази даних. Він просто створює фреймворк інтерфейсу, тому ви можете використовувати його з будь-яким бекендом. Під капотом він використовує Babel і webpack, але вам навіть не потрібно знати про них. Коли програма буде готова до розгортання у виробництво, вам потрібно запустити команду npm run build. Це створить оптимізовану збірку програми в папці збірки.

Додаткову інформацію про Create React App можна знайти за посиланнями «Прочитати мене» та «Посібник користувача». Next.js — популярна легка платформа для статичних і серверних додатків, створених на React.

Він включає в себе готові рішення для стилізації та маршрутизації, якщо ви використовуєте Node.js як середовище свого сервера.

Gatsby — найкращий спосіб створювати статичні сайти за допомогою React. Він дозволяє використовувати компоненти React, але відображає попередньо намальовані HTML і CSS, щоб забезпечити найшвидший час завантаження.

Neutrino – поєднує потужність webpack із простотою встановлення. Включає попередній перегляд програм React і компонентів React.

Nwb особливо корисний для публікації компонентів React у npm. Його також можна використовувати для створення додатків React.

Parcel — це швидкий конструктор веб-додатків без конфігурації, який працює з React.

Razzle — це платформа відтворення на стороні сервера, яка не вимагає жодної конфігурації, але забезпечує більшу гнучкість, ніж Next.js.

3.2. Робота з базою даних

База даних - це просто набір структурованих даних. Наприклад, коли ви створюєте базу даних, саме тут зберігаються дані. Термін «реляційний» відноситься до організації даних, що зберігаються в наборі даних, у вигляді таблиці. Кожна таблиця певним чином пов’язана.

Якщо програмне забезпечення не підтримує реляційну модель даних, просто назвіть його СУБД. MySQL — це система управління базами даних з відкритим вихідним кодом (СУБД) з моделлю клієнт-сервер.

СУБД — це програмне забезпечення або сервіс для створення та керування базами даних на основі реляційної моделі. Тепер давайте детальніше розглянемо кожен термін: Відкритий код означає, що ви можете вільно використовувати та змінювати його. Будь-хто може встановити програмне забезпечення. Ви також можете досліджувати та налаштовувати вихідний код, щоб краще відповідати вашим потребам.

Однак GPL (GPU Public License) визначає, що ви можете робити на основі умов. Якщо вам потрібна більш гнучка права власності та розширена підтримка, доступна версія з комерційною ліцензією. Комп’ютер, який встановлює та запускає програмне забезпечення СУБД, називається клієнтом. Коли їм потрібно отримати доступ до даних, вони підключаються до сервера СУБД.

Це система клієнт-сервер. MySQL є одним із багатьох варіантів програмного забезпечення СУБД. Через популярність MySQL вважається, що СУБД і MySQL — одне і те ж. Назвіть кілька чудових веб-програм, таких як Facebook, Twitter, YouTube, Google і Yahoo! Обидва використовують MySQL для зберігання даних.

Хоча спочатку він був розроблений для обмеженого використання, тепер він сумісний з багатьма важливими обчислювальними платформами, такими як Linux, macOS, Microsoft Windows і Ubuntu.

MySQL і SQL не збігаються. Пам’ятайте, що MySQL є одним із найпопулярніших брендів програмного забезпечення СУБД, що реалізує модель клієнт-сервер. Як же взаємодіють клієнти та сервери в середовищі СУБД? Вони використовують специфічну для домену мову — мову структурованих запитів (SQL). Якщо ви коли-небудь зустрічали інші назви, які включають SQL, наприклад PostgreSQL і Microsoft SQL Server, швидше за все, це бренди, які також використовують синтаксис SQL.

Програмне забезпечення СУБД часто пишеться іншими мовами програмування, але завжди використовує SQL як основну мову для взаємодії з базою даних. Сам MySQL написаний на C і C++. Подумайте про країни Південної Америки, усі вони географічно різні та мають різну історію, але більшість із них розмовляють іспанською.Запит даних: запит конкретної інформації з існуючої бази даних.

Маніпулювання даними: операції додавання, видалення, зміни, сортування тощо для зміни даних, значень або візуальних елементів. Розпізнавання даних: визначення типів даних, наприклад зміна числових даних на цілі. Це також включає визначення схем або зв'язків для кожної таблиці в базі даних.

Контроль доступу до даних: забезпечує технологію безпеки для захисту даних, включаючи рішення, хто може переглядати або використовувати будь-яку інформацію, що зберігається в базі даних. Ви можете використовувати ряд драйверів для використання сервера MySQL у Node.js.

Найпопулярнішими з них є mysql і mysql2. У більшості випадків вони сумісні. У цьому випадку ми будемо використовувати mysql2, оскільки згідно з деякими тестами він забезпечує хорошу продуктивність. Один або кілька пристроїв (клієнтів) підключаються до сервера через певну мережу.

Кожен клієнт може робити запити з графічного інтерфейсу користувача (GUI) на своєму екрані, і якщо обидва сторони зрозуміють інструкції, сервер дасть бажаний результат. Не вдаючись у технічні аспекти, основний процес, який відбувається в середовищі MySQL, той самий. MySQL створює базу даних для зберігання та керування даними, які визначають відносини кожної таблиці.

Клієнти можуть робити запити, вводячи певні команди SQL в MySQL. Серверна програма відповість запитаною інформацією та з'явиться на стороні клієнта. Клієнт зазвичай вказує, який графічний інтерфейс MySQL використовувати. Чим простішим і зручнішим буде графічний інтерфейс користувача, тим швидшою та простішою буде операція з управління даними.

Одними з найпопулярніших графічних інтерфейсів MySQL є MySQL WorkBench, SequelPro, DBVisualizer і Navicat DB Admin Tool. Деякі з них безкоштовні, деякі є комерційними, деякі працюють виключно для macOS, а деякі сумісні з основними операційними системами. Клієнти повинні вибрати графічний інтерфейс відповідно до своїх потреб (рис. 3.2).

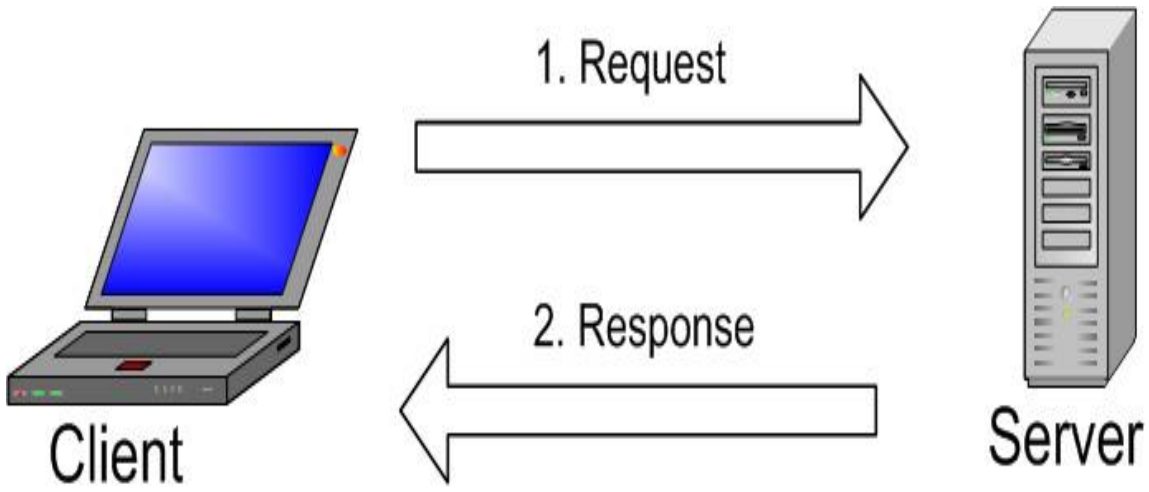

Рис. 3.2. Принцип роботи MySQL

Щоб створити з'єднання, використовуйте метод createConnection(), який приймає параметри підключення як параметри і повертає об'єкт, що представляє

з’єднання.(рис.3.3).

```
const mysql = require("mysql2");

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  database: "usersdb",
  password: "пароль_от_сервера"
});
```

Рис. 3.3. Використання createConnection для підключення

Конфігурація, передана методу configure, може містити багато параметрів. Серед найбільш часто використовуваних:

1. Хост – хост, на якому запущено сервер mysql. "localhost" є за замовчуванням.

2. Порт - номер порту, на якому працює сервер mysql. За замовчуванням "3306" .

3. Користувач – користувач MySQL, який використовувався для підключення

4. Password – пароль для MySQL

5. База даних - ім'я бази даних, до якої встановлено з'єднання. Додатковий параметр. Якщо не вказано, з’єднання зазвичай дефолтне.

6. Charset - кодування, яке використовується для з'єднання, наприклад, за замовчуванням є "UTF8_GENERAL_CI".

7. Часовий пояс - часовий пояс сервера MySQL. Це використовується для перетворення типів значень дати/часу сервера в JavaScript. За замовчуванням "локальний".

Для підключення використаємо метод connect () об'єкта connection (рис. 3.4).

```
const mysql = require("mysql2");

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  database: "usersdb",
  password: "пароль_от_сервера"
});
 connection.connect(function(err){
    if (err) {
      return console.error("Ошибка: " + err.message);
    }
    else{
      console.log("Подключение к серверу MySQL успешно установлено");
    }
 });
```

Рис. 3.4 . Використання connect

Коли ви запускаєте програму та успішно підключаєтесь - коли з'єднання закрито, ми побачимо на консолі, що з'єднання успішне (рис. 3.5).

```
Подключение к серверу MySQL успешно установлено
Подключение закрыто
```

Рис. 3.5. Підключення до сервера MySQL успішно встановлено

Метод end() гарантує, що всі запити, що залишилися, які не були виконані до виклику цього методу, будуть виконані до закриття з’єднання з базою даних. Якщо ми не викличемо цей метод, з’єднання залишиться активним, а програма Node.js продовжуватиме працювати, поки ви не під’єднаєтеся до сервера MySQL. Якщо нам потрібно негайно закрити з’єднання, не чекаючи решти запитів, ми можемо використовувати метод destroy(). [16].

3.3. Створення уявлення бази даних

Представлення (VIEW) – об’єкт бази даних, який є результатом запиту бази даних, визначеного оператором SELECT під час запиту. Представлення іноді називають «віртуальною таблицею». Назва пояснюється тим, що подання доступне

користувачеві як таблиця, але воно не містить даних, і воно витягується з таблиці при зверненні. Якщо дані в базовій таблиці змінюються, користувачі отримають останні дані під час доступу до представлення даних, яке використовує цю таблицю; результати зразка в таблиці кешуються, коли представлення не виконується. У цьому випадку механізм кешування запитів працює на рівні запиту користувача, незалежно від того, чи має користувач доступ до таблиці чи подання.

Представлення можуть бути засновані на таблицях та інших представленнях, тобто вони можуть бути вкладеними (до 32 рівнів вкладення). Переваги використання ідей: в

1. Дозволити гнучке налаштування дозволів на доступ до даних, оскільки дозволи призначені не для таблиць, а для продуктивності. Це дуже зручно, якщо користувачеві потрібно надати дозволи на окремі рядки таблиці або можливість отримати не самі дані, а результат якоїсь операції над ними.

2. Дозволяє розділити логіку зберігання та програмне забезпечення. Ви можете змінити структуру даних, не впливаючи на код програми, вам просто потрібно створити подання, подібне до таблиці, до якої раніше звертався додаток. Це дуже зручно, коли неможливо змінити програмний код або коли є кілька програм, які мають різні вимоги для доступу до бази даних про структури даних.

3. Простота використання за рахунок автоматизації операцій, таких як доступ до певних частин рядків і/або стовпців, отримання даних з кількох таблиць і їх перетворення за допомогою різних функцій

Обмеження в MySQL:

1. Не можна створити тригер на уявлення.

2. Він не може бути виражений на основі тимчасових таблиць, тимчасове агентство не може бути здійснено.

3. Представлення на запит не може використовуватися у визначенні частини FROM.

4. Системні та користувацькі змінні не можна використовувати у визначеннях представлень; у збережених процедурах ви можете використовувати локальні змінні або параметри процедури для визначення представлень.

5. Параметр PREPARE не можна використовувати для визначення представлення.

6. Таблиці та уявлення, присутні у визначенні представлення, повинні існувати.

7. Запити дозволені лише для представлень, які задовольняють кільком вимогам, таким як UPDATE, DELETE та INSERT [17].

Давайте створимо ідею, проаналізувавши інформацію, яка повинна зберігатися в програмі в наступних розділах:

1. Чат (рис. 3.6) та повідомлення (рис. 3.7) на сторінці “Communication”.
2. Поточний пункт на бічній панелі (рис. 3.8).
3. Завдання на сторінці “Board” (рис. 3.9) та коментарі (рис. 3.10).
4. Документи на сторінці “Documents” (рис. 3.11).
5. Звіти на сторінці “Reports” (рис. 3.12).

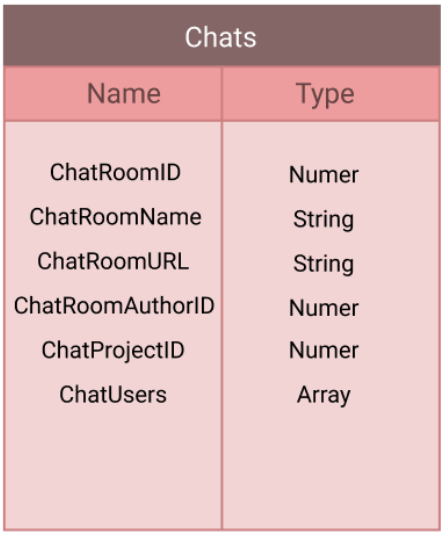

Рис. 3.6. Представлення чату

Messages	
Name	Type
ChatMessageID	Numer
ChatAuthorID	Number
ChatRoomID	Number
ChatText	String
ChatFiles	Array
ChatDate	Date

Рис. 3.7. Представлення повідомлення

Project	
Name	Type
ProjectID	Numer
ProjectName	String
ProjectDesc	String
ProjectURL	String
ProjectStatus	String
ProjectAuthorID	Number
ProjectUsers	Array

Рис. 3.8. Представлення проекту

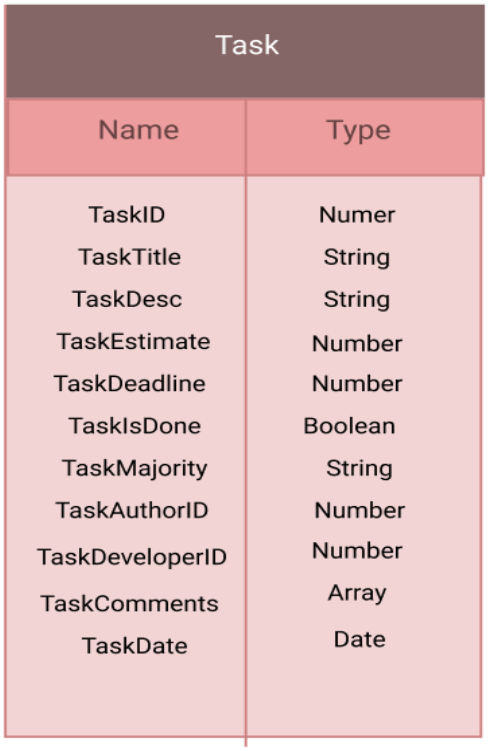

Рис. 3.9. Представлення задачі

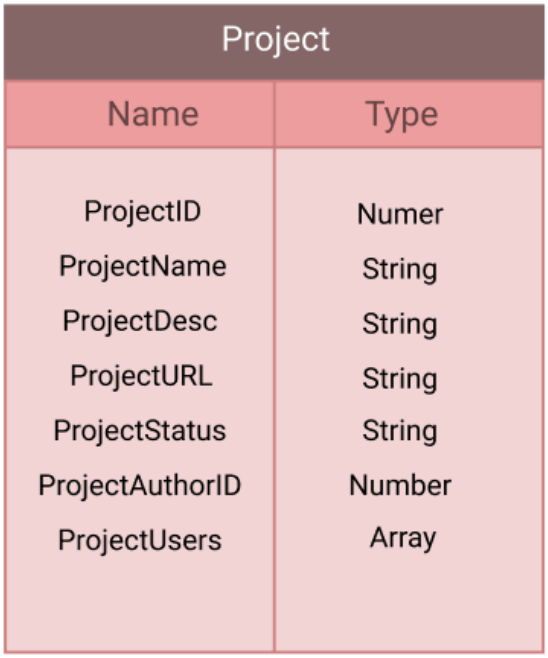

Рис. 3.10. Представлення коментарів

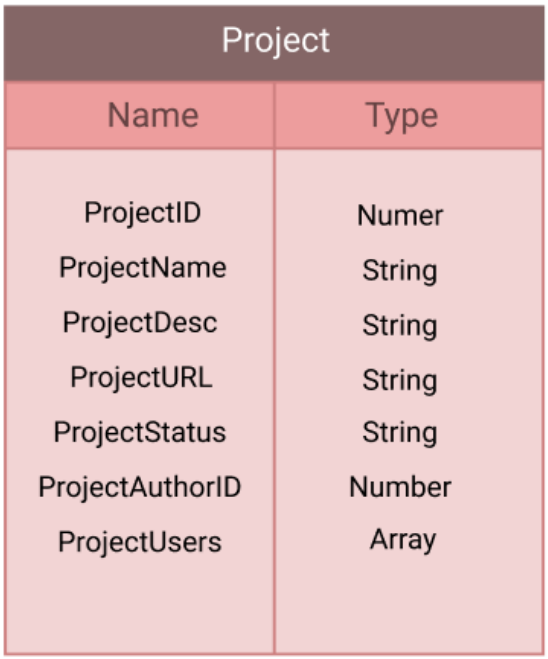

Рис. 3.11. Представлення документів

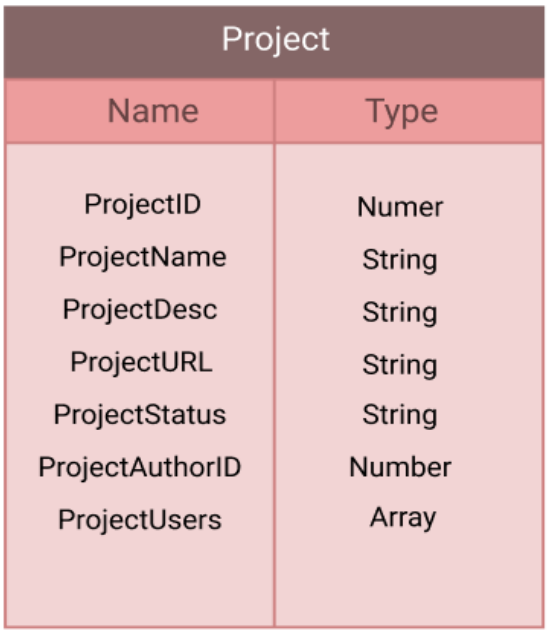

Рис. 3.12. Представлення звітів

3.4. Опис створення макету

Інтерфейс розділу макета складається з кількох блоків (рис. 3.13):

1. Ліва панель показує всю структуру проекту. Ви можете відстежити вкладення елементів і знайти правильний. Кожен тип елемента позначається окремим значком – текст, зображення, компонентний блок.
2. Основний робочий простір по центру - для безпосередньої взаємодії з макетом.
3. У верхній частині сторінки знаходиться головне меню. Більшість із цих інструментів призначені для дизайну, але вам точно знадобиться меню масштабування праворуч.
4. Бічна панель праворуч містить три вкладки, але нам знадобляться лише дві з них – Дизайн і Код. Ці вкладки містять всю доступну інформацію про об’єкт.

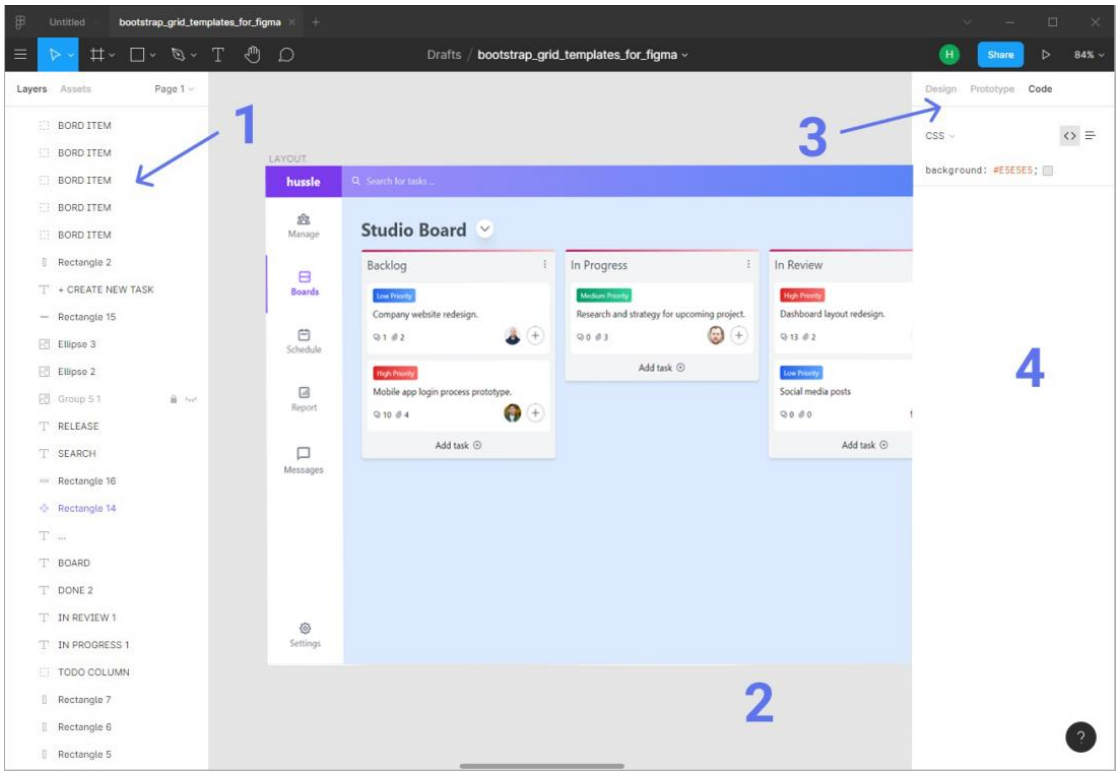

Рис. 3.13. Робочі інструменти Figma.

Розглянемо сторінку профіля користувача (див. рис. 3.24). Основний функціонал, який представлений:

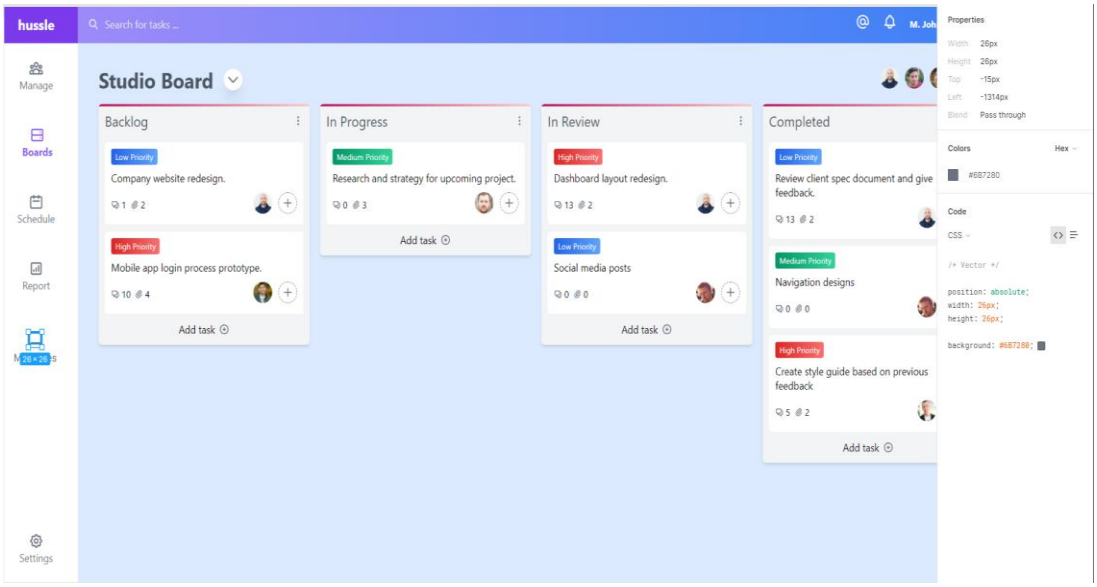

Рис. 3.14. Параметри елементу в Figma

Ви також можете завантажити будь-яке зображення з макета. Для цього перейдіть на вкладку Дизайн і знайдіть останній елемент експорту. Далі у спадному меню виберіть формат, у якому потрібно експортувати зображення. Розмір можна визначити кількома способами. Наприклад, на вкладці «Код» перегляньте значення властивостей «Ширина» і «Висота». Або просто виберіть потрібний блок - розмір відобразиться внизу елемента. Також можна дізнатися відстань між будь-яким елементом і його сусідами. Для цього виділіть його, потім наведіть курсор на інший елемент – з’явиться напрямна та значення [18]. Розробка дизайну створюється із завдань, описаних у вимогах проекту. Створіть домашню сторінку програми (рис 3.15) з усією інформацією про поточний проект і бічну панель для переходу до головного розділу. Створено дошку з усією інформацією про тему, обрану завданням: назва завдання, статус, час, пріоритет, виконавець і команда, пов’язана з цим завданням (рис. 3.16).

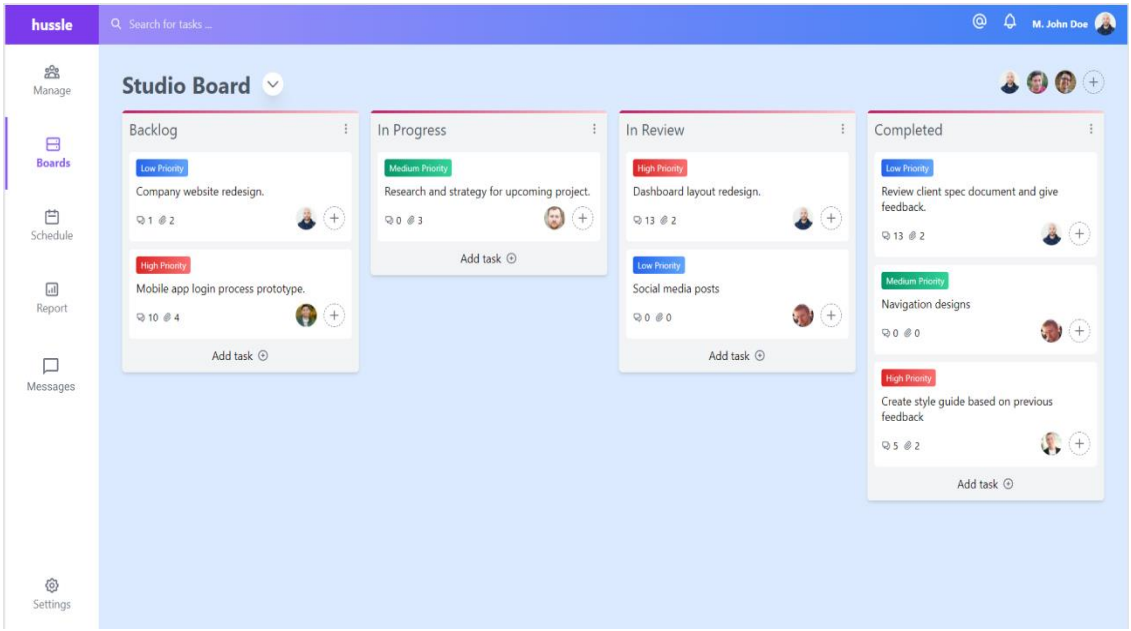

Рис. 3.15. Дизайн дошки з можливостями менеджменту

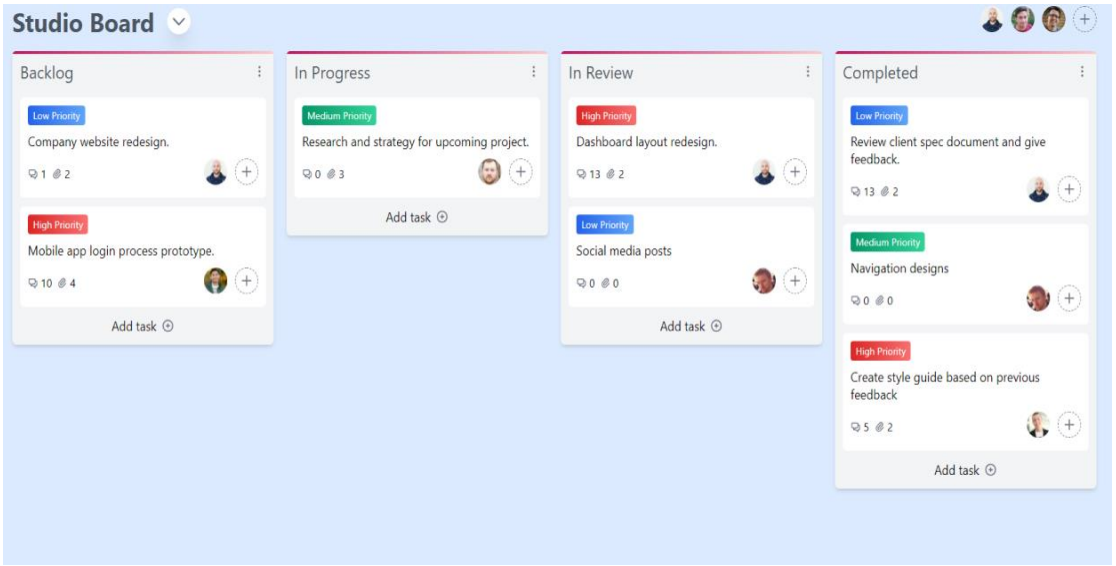

Рис. 3.16. Дизайн дошки, що відображає інформацію про проект

Створено розділ спілкування, який дозволяє створювати нові чати для різних завдань або груп розробників і дозволяє легко обмінюватися інформацією та файлами (рис. 3.17).

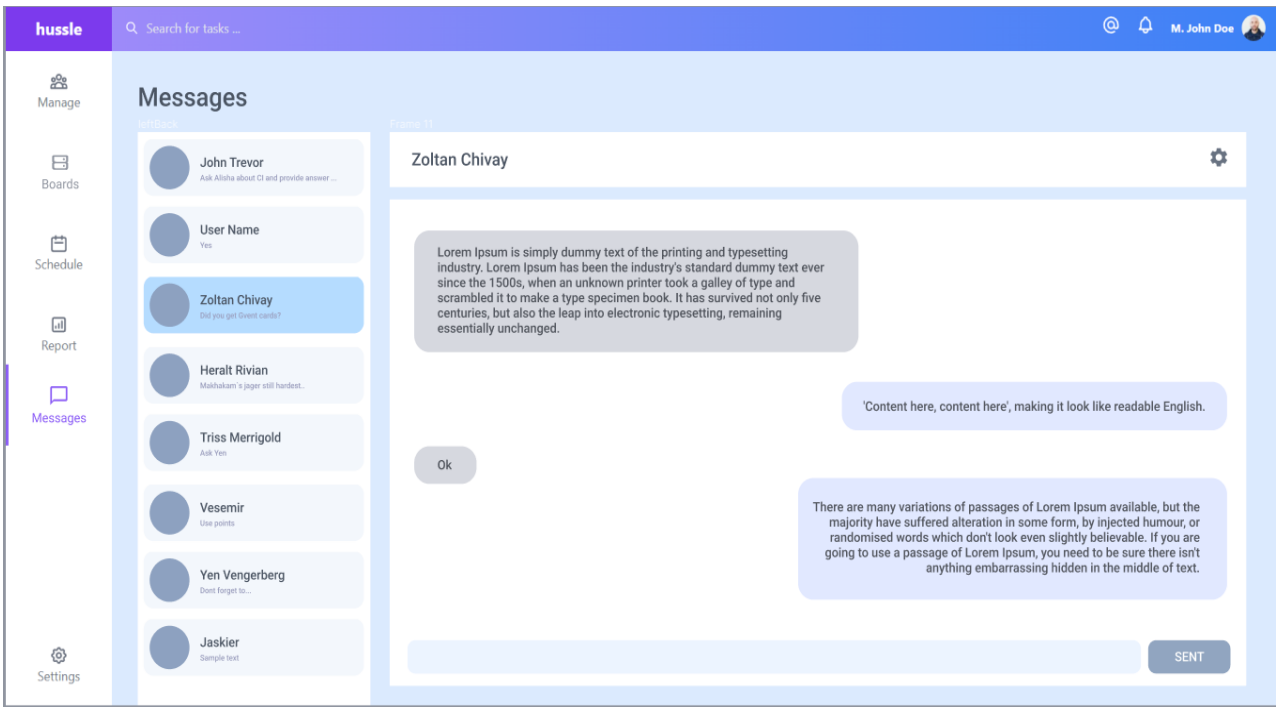

Рис. 3.17. Дизайн корпоративного чату, що має канали для різних груп розробників

3.5. Опис програмної реалізації

Спочатку додайте логотип до капелюха. Для цього в папці src буде створена папка компонентів, яка міститиме папку logo_header. У цю папку завантажуємо логотип і створюємо 2 файли LogoHeader.js і LogoHeader.css (рис. 3.18) [19].

Тепер відобразимо назву кімнати чату. Для цього скористайтеся компонентом RoomTitle, щоб створити папку з назвами кімнат. Ім’я в цьому компоненті ми називатимемо через props, тому пишемо props.chatroomName і тепер перемістимо його сюди (рис. 3.19).

```
import React from 'react'
import logo from './chatix_logo.svg';
import './LogoHeader.css';

function LogoHeader(){
    return (
        <div className="LogoHeader">
            <img src={logo} className="App-logo" alt="Chatix logo" />
        </div>
    );
}

export default LogoHeader;
```

Рис. 3.18. Рендер логотипу

```
import React from 'react';
import './RoomTitle.css';

function RoomTitle(props){
    return (
        <div className="RoomTitle">
            <h1>{props.chatroomName}</h1>
        </div>
    );
}

export default RoomTitle;
```

Рис. 3.19. Створення кімнати для чату

Потім створіть сам компонент заголовка і вставте в нього логотип та назву чату. Негайно передайте ім’я чату дочірньому компоненту через параметр chatroomName. Нагадаю, що ми погоджуємося, що всі дані (стан програми) зберігатимуться в кореневому компоненті програми. Звідти ми спочатку передаємо заголовок у заголовок, а потім із заголовка в заголовок номеру components \ header \ Header.js (рис. 3.20).

```
import React from 'react';
import './Header.css'
import LogoHeader from '../logo_header/LogoHeader';
import RoomTitle from '../room-title/RoomTitle';

function Header(props) {
    return (
        <header>
            <LogoHeader/>
            <RoomTitle chatroomName={props.chatroomName} />
        </header>
    );
}

export default Header;
```

Рис. 3.20. Компонент хедеру

```
import React from 'react';
import './App.css';
import Header from './components/header/Header';

class App extends React.Component {
    constructor(props){
        super(props);
        chatroomName: 'Чат-комната',
        me: {
            is_online: true,
            name: "Алексей",
            uuid: "98s7dfh9a8s7dhf"
        }
    }
    render() {
        return (
          <div className="App">
            <Header
                chatroomName={this.state.chatroomName}
                me={this.state.me}
            />
          </div>
        );
    };
}

export default App;
```

Рис. 3.21. Компонент хедеру

Далі відкрийте файл App.js і додайте компонент Header.js. Потім ми додаємо назву до стану і будимо його за допомогою реквізитів. Вам потрібно додати ім’я поточного користувача в заголовок. Для цього додайте об’єкт користувача до стану і подібним чином пробудіть його в заголовку (рис. 3.21).

Тепер вам потрібно додати в заголовок введення, що містить поточне ім’я користувача, і призначити обробник для внесення змін, щоб ми могли передати нове ім’я користувача компоненту програми. Для цього додайте вхід з ім'ям функції-обробника handleChangeName і викликайте функцію зворотного виклику

props.updateVisitor, куди ми передаємо об'єкт користувача з оновленим ім'ям (рис. 3.22).

```
function Header(props) {
    const [name, setName] = useState(props.me.name ? props.me.name : props.m
e.uuid.substr(-10))

    const handleChangeName = (e) => {
        setName(e.target.value)
        let visitor = {...props.me};
        visitor.name = e.target.value;
        props.updateVisitor(visitor)
    }

    return (
        <header>
            <LogoHeader/>
            <RoomTitle chatroomName={props.chatroomName}/>
            {
                props.me ?
                    <input
                        className='name-input'
                        value={name}
                        placeholder='Ваше имя'
                        onChange={(e) => handleChangeName(e)}
                    />
                : null
            }
        </header>
    );
}
```

Рис. 3.22. Поле для поточного ім'я

Далі створіть компонент компоненти/message-container/MessageContainer.js. Підключіть його до ChatField і прокиньтеся до повідомлень у ньому (рис. 3.23).

```
function Main(props) {
    return(
        <section className="Main">
            <MemberList
                me={props.me}
                members={props.members} />
            <ChatField messages={props.messages} />
        </section>
    );
}
```

Рис. 3.23. Компонент прев’ю повідомлення

Далі ми переглянемо всі повідомлення і для кожного повернемо компонент, який буде його відображати. Давайте створимо його компонент/message/Message.js. де ми відображаємо значок відвідувача, його ім'я або ідентифікатор (якщо ім'я не вказано) і текст повідомлення (рис. 3.24).

```
function Message(props) {

    const getSenderName = () => {
        if (props.sender) {
            return props.sender.name ? props.sender.name : props.sender.uuid.
substr(-10);
        }
        return "Unknown sender";
    };

    return(
        <div className="Message">
            <div className="message-sender-icon">
                <img src={icon} alt="visitor icon"/>
            </div>
            <div className="message-bubble">
                <div className="message-sender-name">{getSenderName()}</div>
                <div className="message-content">{props.message.content}</div
>
            </div>
        </div>
    );
}
```

Рис. 3.24. Компонент повідомлення

Тепер у MessageContainer ми перебираємо всі повідомлення і для кожного повідомлення повертаємо компонент Message, в якому передається об’єкт повідомлення. Створіть компонент із формою для надсилання повідомлень

component/send-message-form/SendMessageForm.js. У ньому ми створимо вхід і кнопку для відправки. Коли ви змінюєте введення, запишіть в ньому текст до стану, а при натисканні кнопки викличте функцію зворотного виклику onSendNewMessage і відправте їй повідомлення зі стану. Пізніше ми створимо функцію onSendNewMessage в компоненті App і передаємо її через props (рис. 3.25). Перевіримо результати (рис. 3.26) [20].

```
currentMessageChanged = (e) => {
    this.setState({message: e.target.value });
}

sendMessageClicked = async (e) => {
    e.preventDefault();
    if (this.state.message.length > 0) {
        await this.props.onSendNewMessage(this.state.message);
        this.setState({...this.state, ...{message : ''}});
    }
}
```

Рис. 3.25. Відправка повідомлення

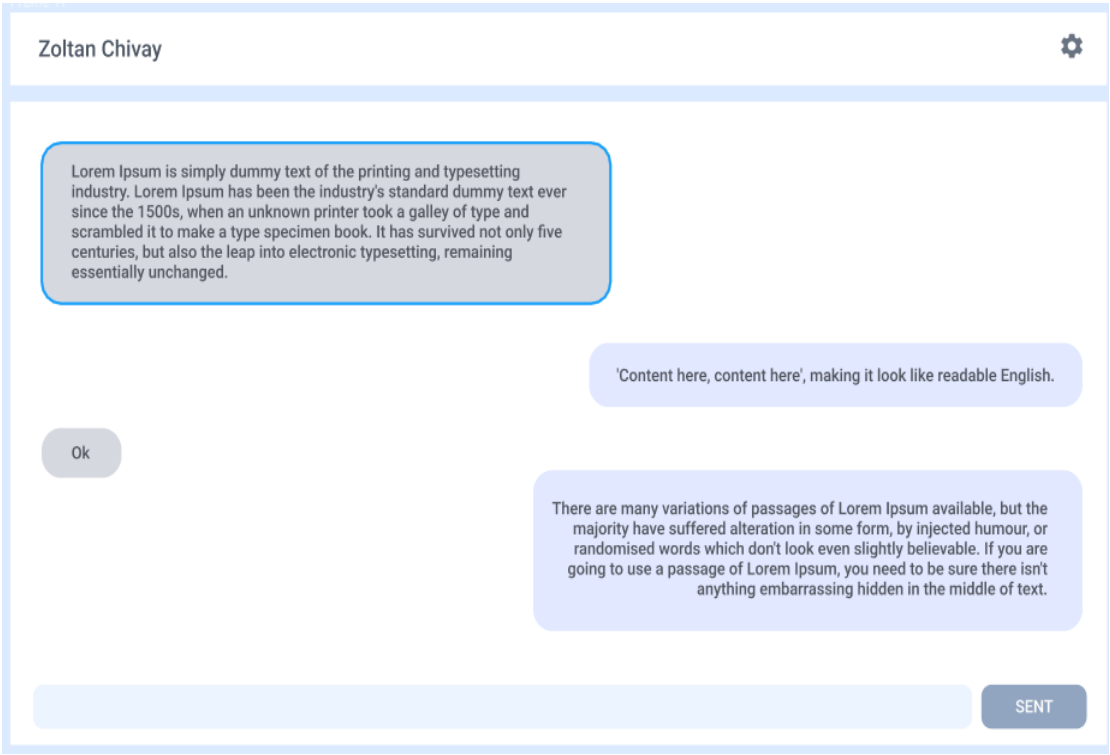

Рис. 3.26. Результат роботи чату

React-trello використовується для створення дощок [21]. Бібліотека дозволяє створювати дошки, заповнювати завдання та контролювати стан команди. серед цих можливостей:

1. Можливість кастомізації.
2. Легкість в підключенні до програми реагування.
3. Підтримка пагінації під час прокрутки смуг.
4. Перетягування карток с завданням на колонках.
5. Можливість редагування функції додавання/видалення карток.
6. Визначте спеціальні елементи для стилів стовпців і карток.
7. Можливість запускати події ззовні (наприклад: додавання або видалення карт на основі подій із бекенда).
8. Вбудоване редагування заголовків.

Компонент Board приймає параметр під назвою data, який містить усі деталі. Функція перетягування за замовчуванням для об’єктів картки. Дані зберігаються у форматі JSON (рис.3.27).

```
const data = {
  lanes: [
    {
      id: 'lane1',
      title: 'Planned Tasks',
      label: '2/2',
      cards: [
        {id: 'Card1', title: 'Write Blog', description: 'Can AI make memes', label: '30 mins', drag
        {id: 'Card2', title: 'Pay Rent', description: 'Transfer via NEFT', label: '5 mins', metadat
      ]
    },
    {
      id: 'lane2',
      title: 'Completed',
      label: '0/0',
      cards: []
    }
  ]
}
```

Рис. 3.27. Формат даних для борди

```
import React from 'react'
import Board from 'react-trello'

export default class App extends React.Component {
  render() {
    return <Board data={data} />
  }
}
```

Рис. 3.28. Формат даних для борди

Ви можете зробити всю дошку доступною для редагування, встановивши для редагованого редактора значення "true". Підтримка цього перемикача дозволить вам видалити наявні картки та відобразить посилання «Додати картку» внизу кожного стовпця. При натисканні відобразиться нова картка з новою редагуванням. Властивості стилю можна передавати як частину даних. Цей метод залежить від використовуваних компонентів картки завдання та стовпця (рис. 3.29).

```
const data = {
  lanes: [
    {
      id: 'lane1',
      title: 'Planned Tasks',
      style: { backgroundColor: 'yellow' },  // Style of Lane
      cardStyle: { backgroundColor: 'blue' } // Style of Card
      ...
};

<Board
  style={{backgroundColor: 'red'}}  // Style of BoardWrapper
  data={data}
  />
```

Рис. 3.29. Кастомізація стилю

Перевіримо результат (рис.3.30).

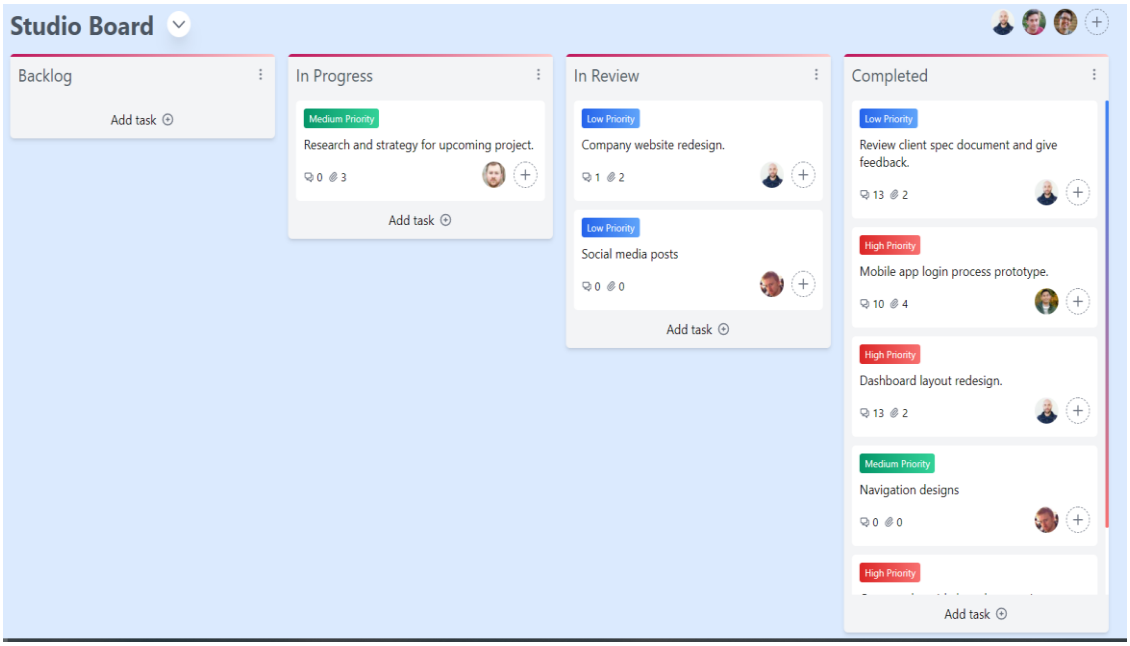

Рис. 3.28. Результат дошки задач

Висновок до розділу 3

У цьому розділі демонструється створення макета, програмна реалізація серверної та клієнтської частин веб-додатка. Для створення макета SPA, тобто розділу чату, дошки завдань, документації, коментарів, використовувався розробник макета Figma. У дизайні використовуються основні принципи UI/UX і досягаються його цілі:

1. Чіткість – інтерфейс однозначний, а текст і структура направляють користувача до мети.

2. Стислий – важливо не перевантажувати інтерфейс підказками, спливаючими вікнами та анімацією. Запитайте себе: «Чи це вам тут потрібно? Навіщо?» Це допоможе вам не перевантажувати сторінку і зосередити увагу користувача на конкретних елементах.

3. Видимість – елементи повинні бути впізнаваними, навіть коли користувачі бачать ваш веб-сайт вперше. Для цього зробіть інтерфейс інтуїтивно зрозумілим. Наприклад, вам не потрібно робити помаранчеву кнопку підтвердження, якщо вона зелена на більшості сайтів. Або, якщо ваші клієнти звикли прокручувати зверху вниз, не використовуйте горизонтальну прокрутку.

4. Чуйність – хороший інтерфейс миттєво реагує на дії користувача. Обов’язково потрібно зрозуміти, що зараз відбувається на екрані: чи здійснено платіж, чи отримав менеджер заявку, чи надіслано повідомлення? За це відповідає відкритий текст.

5. Послідовність – дотримуйтеся послідовності в усіх частинах веб-сайту та програми. Елементи інтерфейсу - меню та повзунки - повинні вести себе однаково на будь-якій сторінці..

6. Естетика - Візуально інтерфейс повинен бути привабливим. Хороший інтерфейс – це той, де користувачеві подобається працювати, не дратуючи і не відволікаючи його від вирішення проблем.

7. Ефективність. Крім зовнішньої привабливості, хороший інтерфейс економить час користувача та доставить його в потрібне місце з мінімальними зусиллями.

8. Поблажливість – навіть із найскладнішим інтерфейсом жоден користувач не застрахований від помилок. Подбайте про інформацію, якщо щось піде не так. Це допоможе заощадити гроші, час і лояльність клієнтів у разі поломки.

Аналізовано та створено представлення бази даних на основі потрібних частин програми. Поля та типи даних кожного представлення розроблені в архітектурі та макеті програми, що дозволяє легко створювати частини сервера та запити між сервером і SPA. Створення розділів сервера, тобто отримання, видалення, додавання та редагування запитів на повідомлення, кімнати чату, дошки завдань, коментарі, документи та звіти. Перевірте, чи запит працює правильно, і додайте перевірки форматування поля. Крім того, програмне забезпечення React, Redux і Thunk реалізує програму та всю функціональність, надану для кожної частини та окремого компонента. Архітектура програми є модульною, що дозволяє легко модифікувати та розширювати програму за потреби. Кожна встановлена ціль розробки була досягнута, а вся функціональність була охоплена та перевірена.

ВИСНОВКИ

У магістерській роботі розроблено програмне забезпечення для корпоративного чату, яке має функції перегляду коду, управління проектами, тайм-менеджменту тощо.

У першій частині аналізується сфера IT-компанії, комунікаційний процес всередині IT-компанії та основні етапи управління проектами. Поставлені завдання та вимоги до платформи та інтерфейсу. Розроблено чіткий план для автоматизації повсякденних завдань управління проектами та спілкування в команді в єдину систему якості, яка може задовольнити потреби менеджерів, бізнес-аналітиків та розробників. Система допоможе швидко планувати завдання, спринти, релізи; матиме можливість створювати дошки Scrum або Canban; матиме можливість відстежувати завдання та створювати коментарі; писати звіти та технічні документи; надавати доступ на рівні проекту, покращувати за допомогою інтеграції комунікації послуги та GIT Communication та перегляд коду.

У другій частині представлено математичну модель та архітектуру для вирішення проблеми створення корпоративного чату з можливостями перегляду коду, управління проектами та тайм-менеджментом. У дослідженні аналізуються можливості управління людськими ресурсами з урахуванням їхньої кваліфікації при розподілі проектів (завдань). Проаналізовано вимоги до створення проекту, розроблено базову структуру проекту, розбитий функціонал на окремі блоки, що відповідають сторінкам програми.Окремі компоненти кожної сторінки засновані на цих компонентах та дизайні майбутній сервіс створюється за допомогою інструменту Figma.

Третя частина демонструє створення макета, програмну реалізацію серверної та клієнтської частин веб-додатка. Програмне забезпечення React, Redux і Thunk реалізує веб-додатки та всю функціональність, надану для кожної частини та окремого компонента. У спеціальному розділі аналізуються місця створення

онлайн-платформ. Визначте критерії шуму, вібрації та випромінювання. Сформульовано правила гігієни праці та виробничої гігієни, техніки безпеки, пожежної безпеки, цивільного захисту, екологічної безпеки та особистої безпеки в надзвичайних ситуаціях. Пропоновані заходи щодо покращення умов праці. Залежно від мети реалізуйте наступні завдання магістерської роботи:

1. Аналіз сфери IT-компаній. Постановка задачі.
2. Математичні моделі та архітектура для вирішення задачі створення корпоративного чату з функціями перегляду коду, менеджменту проектів та організації часу.
3. Створення макету. Програмна реалізація серверної та клієнтської частини веб-застосунку.

Архітектура програми є модульною, що дозволяє легко модифікувати та розширювати програму за потреби. Кожна мета розробки досягнута, вся функціональність охоплена та перевірена.

СПИСОК ПОСИЛАНЬ

1. Сучасний JavaScript: вебсайт URL: https://www.oreilly.com/library/view/modern-javascript/9781492023548/ (дата звернення: 7.01.2021).

2. JavaScript для веб-дизайнерів : вебсайт URL: http://backspaces.net/temp/Ebooks/JavaScript-for-Web-Designers/javascript-for-web-designers.pdf (дата звернення: 10.01.2021).

3. Дронов В. JavaScript у Web-дизайні : довідник, Санкт-Петербург, 2013, 184 с.

4. Дронов В.А. Розробка сучасних Web-сайтів. : довідник, Львів, 2018, 274 с.

5. Фленаган Д. JavaScript. Детальний керівництво. : довідник, Одесса, 2012, 195 с.

6. JavaScript for Web : вебсайт URL: http://backspaces.net/temp/Ebooks/JavaScript-for-Web-Designers/javascript-for-web-designers.pdf (дата звернення: 10.01.2021).

7. Web Security : вебсайт URL: https://www.oreilly.com/library/view/web-security-2016/9781940111452/ (дата звернення: 10.01.2021).

8. Хоумер А. Динамічний HTM: довідник, Київ, 2017, 353 с.

9. Що таке SPA-додатки. : веб-сайт. URL: https://wezom.com.ua/blog/chto-takoe-spa-prilozheniya

10. https://mc.today/popytki-hr-menedzherov-stroit-iz-sebya-psihologov-smeshny-chem-na-samom-dele-dolzhen-zanimatsya-hr/

11. https://www.eternussolutions.com/2020/12/21/redux-thunk-redux-saga/

12. https://dev.to/workshub/state-management-battle-in-react-2021-hooks-redux-and-recoil-2am0

13. https://dev.to/appmapruby/startups-can-still-feel-good-about-choosing-bootstrap-in-2021-15el

14. https://codersera.com/blog/learn-express-js/

15. https://medium.com/selleo/top-trends-in-node-js-to-watch-in-2021-d94ff38cc31e

16. https://www.edureka.co/blog/advantages-and-disadvantages-of-angular/#AdvantagesDisadvantages