МІНІСТЕРСТВО ОСВІТИ НАУКИ УКРАЇНИ
ПРИВАТНЕ АКЦІОНЕРНЕ ТОВАРИСТВО «ПРИВАТНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра інформаційних технологій

ДО ЗАХИСТУ ДОПУЩЕНА

Завідувач кафедри,
д.е.н., доц.
_____ С.І. Левицький

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

ПРОГРАМУВАННЯ АНАЛІТИЧНИХ ЗАСОБІВ ПРОЕКТНОГО УПРАВЛІННЯ В ЗВО ЗА ДОПОМОГОЮ СЛАБКИХ СИГНАЛІВ

Виконав

ст. гр. ІПЗ – 210 _____ В. І. Волков

Керівник

к.е.н., доц. _____ О. В. Шляга

Запоріжжя
2023

ПРИВАТНЕ АКЦІОНЕРНЕ ТОВАРИСТВО «ПРИВАТНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри,
д.е.н., доц.
_____ С.І. Левицький
___.___._____ р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ

студенту гр. ІПЗ-210 ,
спеціальності 121 - «Інженерія програмного забезпечення»
Волкову В'ячеславу Ігоровичу

1. Тема: Програмування аналітичних засобів проектного управління в ЗВО за допомогою слабких сигналів

затверджена наказом № 02-10 від 27 січня 2023 р.

2. Термін здачі студентом закінченої роботи: 12 червня 2023 р.

3. Перелік питань, що підлягають розробці:

1. Здійснити аналітичний огляд за темою дослідження
2. Розглянути особливості системного аналізу інформаційних повідомлень
3. Виконати огляд та порівняння аналогів
4. Вивчити особливості технологій управління програмними проектами
5. Розробити структуру програми АРМ керівника програмними проектами, здійснити підбір інструментальних засобів для створення програмного продукту

6. Реалізувати програму АРМ керівника програмними проектами

7. Оформити звіт за результатами роботи.

4. Календарний графік підготовки кваліфікаційної бакалаврської роботи.

№ етапу	Зміст	Терміни виконання	Готовність по графіку %, підпис керівника	Підпис керівника про повну готовність етапу, дата
1	Формулювання (корегування) теми кваліфікаційної бакалаврської роботи та збір практичного матеріалу за темою	16.01.23-11.02.23		
2	**І атестація** І розділ кваліфікаційної бакалаврської роботи	27.03.23-31.03.23		
3	**ІІ атестація** ІІ розділ кваліфікаційної бакалаврської роботи	24.04.23-28.04.23		
4	**ІІІ атестація** ІІІ розділ кваліфікаційної бакалаврської роботи, висновки та рекомендації, додатки, реферат	22.05.23-26.05.23		
5	Перевірка кваліфікаційної бакалаврської роботи на оригінальність	15.05.23-12.06.23		
6	Доопрацювання кваліфікаційної бакалаврської роботи, підготовка презентації, отримання відгуку керівника і рецензії	29.05.23-12.06.23		
7	**Попередній захист кваліфікаційної бакалаврської роботи**	**12.06.23-18.06.23**		
8	Подача кваліфікаційної бакалаврської роботи на кафедру	за 3 дні до захисту		
9	**Захист кваліфікаційної бакалаврської роботи**	**19.06.23-24.06.23**		

Дата видачі завдання: ___.___._____ р.

Керівник кваліфікаційної

бакалаврської роботи _____ О В. Шляга

Завдання отримав до виконання _____ В. І. Волков

РЕФЕРАТ

Кваліфікаційна бакалаврська робота містить 70 сторінок, 48 рисунків, 33 використаних джерела.

Мета бакалаврської дипломної роботи – розробка АРМ керівника програмними проектами.

Об’єкт дослідження – основні методи і засоби розробки АРМ для цілей проектного управління.

Актуальність проектного управління є очевидною. АРМ керівника програмними проектами відстежує фактичне виконання проекту, що допомагає проектному менеджеру налагоджувати процеси розробки та створити успішну систему для клієнту. Програмування аналітичних засобів проектного управління в закладах вищої освіти за допомогою слабких сигналів може бути цікавим напрямом дослідження та розробки.

В результаті розроблена АРМ керівника програмними проектами, що може бути реалізований в ЗВО.

АРМ, ІНФОРМАЦІЙНА СИСТЕМА, СИСТЕМА УПРАВЛІННЯ ПРОГРАМНИМИ ПРОЕКТАМИ, SINGLE PAGE APPLICATION

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Скорочення	Повна назва	Пояснення / переклад
АРМ	Автоматизоване робоче місце	
БД	База даних	
ОС	Операційна система	
ІС	Інформаційна система	
КПП	Керівник програмного проекту	
ПК	Персональний комп’ютер	
ПЗ	Програмне забезпечення	
ПП	Програмний продукт	
СУБД	Система управління базою даних	
СУ	Система управління	
API	Application Programming Interface	
DOM	Document Object Model	
HTML	HyperText Markup Language	
JS	JavaScript	
JSON	JavaScript Object Notation	
SPA	Single Page Application	

ВСТУП

Актуальність проектного управління є очевидною. АРМ керівника програмними проектами відстежує фактичне виконання проекту, що допомагає проектному менеджеру налагоджувати процеси розробки та створити успішну систему для клієнту.

Програмування аналітичних засобів проектного управління в закладах вищої освіти за допомогою слабких сигналів може бути цікавим напрямом дослідження та розробки.

Слабкі сигнали в контексті проектного управління вказують на ранні ознаки змін або тенденцій, які можуть мати вплив на проекти вищої освіти. Ці сигнали можуть бути відображені в даних, що збираються з різних джерел, таких як соціальні медіа, статистика використання ресурсів, фінансові дані, опитування та інші джерела.

Одним з важливих кроків у програмуванні аналітичних засобів є збір та обробка даних. Збирання може включати автоматичне збирання даних з різних джерел та їх агрегацію. Обробка даних передбачає аналіз, фільтрацію, класифікацію та інтерпретацію слабких сигналів, щоб виявити можливі важливі зміни або тенденції.

Один зі способів реалізації програмного забезпечення для аналізу слабких сигналів – використання методів машинного навчання та штучного інтелекту. Алгоритми машинного навчання можуть бути натреновані на існуючих даних для виявлення кореляцій та закономірностей між слабкими сигналами та результатами проектного управління. Штучний інтелект може допомогти в автоматичному виявленні важливих змін та надавати рекомендації з управління проектами на основі аналізу слабких сигналів.

Після обробки та аналізу даних, аналітичні засоби можуть генерувати звіти, візуалізації та інші інструменти для сприяння прийняттю рішень управління проектами. Ці засоби можуть допомогти виявити можливі проблеми, ризики та можливості, пов’язані з проектами вищої освіти.

Важливим аспектом розробки таких аналітичних засобів є співпраця з фахівцями з проектного управління та стейкхолдерами у вищих навчальних закладах для визначення ключових показників ефективності, слабких сигналів та розробки моделей аналізу даних, що відповідають специфіці їхніх потреб.

Узагальнюючи, програмування аналітичних засобів проектного управління в закладах вищої освіти за допомогою слабких сигналів може допомогти виявити ранні ознаки змін та тенденцій, що впливають на проекти. Воно базується на зборі, обробці та аналізі даних, використанні методів машинного навчання та штучного інтелекту, а також співпраці з фахівцями з проектного управління.

Мета бакалаврської дипломної роботи – розробка АРМ керівника програмними проектами.

Об’єкт дослідження – основні методи і засоби розробки АРМ.

Результати бакалаврської дипломної роботи можуть бути використанні в якості навчального матеріалу для персоналу в ЗВО або на підприємстві, які братимуть участь або управлятимуть різноплановими ІТ-проектами.

Кваліфікаційна бакалаврська робота складається зі вступу, трьох розділів, висновків, переліку посилань (33 найменування). Загальний обсяг роботи становить 72 сторінки комп’ютерного тексту, основний зміст роботи викладено на 63 сторінках.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Сутність проектного управління

Сутність проектного управління полягає в систематичному підході до планування, організації, виконання та контролю проектів з метою досягнення конкретних цілей. Основна мета проектного управління – забезпечити успішне виконання проекту в межах обмежень, таких як обсяг ресурсів, бюджет, терміни та якість.

Основні принципи проектного управління включають:

Чіткість цілей: Визначення чітких та вимірюваних цілей проекту, які повинні бути спільно прийняті всіма стейкхолдерами.

Планування: Розробка детального плану, включаючи розподіл завдань, визначення ресурсів, графік виконання та інші важливі елементи.

Організація: Формування команди проекту, визначення ролей та відповідальностей, забезпечення комунікації та співпраці між учасниками проекту.

Виконання: Реалізація плану проекту, керування ресурсами, виконання завдань та вирішення виникаючих проблем.

Моніторинг та контроль: Систематичне спостереження за прогресом проекту, оцінка досягнень відповідно до запланованих показників, виявлення ризиків та вжиття заходів щодо їхнього управління.

Завершення та оцінка: Завершення проекту, оцінка досягнень, вивчення виводів та набутих досвідів для подальшого вдосконалення процесів управління проектами.

Сутність проектного управління полягає в тому, щоб ефективно використовувати ресурси, керувати ризиками, забезпечувати комунікацію та співпрацю, дотримуючись заздалегідь визначених цілей та забезпечуючи успішне завершення проекту. Це комплексний процес, який вимагає керівництва, планування та вміння ефективно управляти змінами.

Проектне управління в закладах вищої освіти (ЗВО) має свої особливості, оскільки діяльність університетів і коледжів відрізняється від інших галузей. Ось деякі особливості проектного управління в ЗВО:

Багатогалузевість: ЗВО займаються різноманітними діяльностями, такими як навчання, наукові дослідження, студентське життя, інфраструктура, міжнародні програми тощо. Проектне управління в ЗВО повинне враховувати цю багатогалузевість та комплексність в управлінні проектами.

Залученість стейкхолдерів: ЗВО мають різноманітних стейкхолдерів, таких як студенти, викладачі, адміністрація, батьки, донори, випускники тощо. Управління проектами в ЗВО повинне враховувати потреби та очікування цих різних груп стейкхолдерів.

Довгостроковість: Багато проектів в ЗВО мають довгостроковий характер, такі як розвиток нових програм, будівництво кампусів, наукові дослідження тривалої тривалості. Управління такими проектами вимагає врахування довгострокових перспектив, планування та управління змінами.

Академічна свобода: В ЗВО академічна свобода є важливою цінністю, і проектне управління повинне враховувати цей аспект. Рішення та управління проектами повинні сприяти забезпеченню свободи наукових досліджень, творчості викладачів та студентської автономії.

Фінансові обмеження: ЗВО часто операціонують з обмеженими фінансовими ресурсами. Управління проектами в ЗВО вимагає ретельного фінансового планування, ефективного використання ресурсів та пошуку додаткових джерел фінансування.

Велика кількість зацікавлених сторін: Проекти в ЗВО можуть мати велику кількість зацікавлених сторін, які можуть мати різні погляди, потреби та очікування. Ефективне управління комунікаціями та співпраця з різними стейкхолдерами є важливим аспектом проектного управління в ЗВО.

Загалом, управління проектами в ЗВО потребує розуміння специфіки освітнього середовища, врахування потреб різних стейкхолдерів та ефективного використання ресурсів для досягнення мети проекту.

Аналітичні засоби в проектному управлінні використовуються для збору, аналізу та інтерпретації даних з метою прийняття обґрунтованих рішень та покращення результативності проектів. Ось декілька аналітичних засобів, що застосовуються в проектному управлінні:

SWOT-аналіз: Допомагає виявити сильні та слабкі сторони проекту, а також можливості та загрози, що впливають на нього. SWOT-аналіз дозволяє зрозуміти контекст проекту та розробити стратегію, що базується на його потенціалі та викликах.

Аналіз ризиків: Допомагає ідентифікувати потенційні ризики, їх ймовірність та вплив на проект. Аналіз ризиків дозволяє розробити план дій для управління ризиками та запобігання можливим проблемам.

Матриця зацікавлених сторін: Визначає зацікавлених сторін проекту і їхні очікування. Цей інструмент допомагає управляти комунікацією зі стейкхолдерами, розуміти їхні потреби та враховувати їхні внески в проект.

Графіки Ганта та мережеві діаграми: Використовуються для візуалізації послідовності та залежності завдань проекту, оцінки тривалості та ресурсного планування. Ці графіки допомагають зорієнтуватись у графіку виконання проекту та визначити критичні шляхи.

Аналітика витрат: Включає в себе відстеження витрат проекту, бюджетного контролю та прогнозування фінансових ризиків. Допомагає забезпечити ефективне фінансове управління проектом та контроль над бюджетом.

Аналіз продуктивності: Вимірює результативність проекту на основі використання ресурсів, графіка виконання, якості виконаних завдань тощо. Допомагає оцінити ефективність проекту та виявити можливості для його вдосконалення.

Дашборди та звіти: Використовуються для візуалізації ключових показників продуктивності проекту, стану виконання завдань, витрат, ризиків тощо. Дозволяють забезпечити швидкий та зрозумілий огляд стану проекту та приймати обґрунтовані рішення.

Ці аналітичні засоби допомагають проектним менеджерам отримати об’єктивну інформацію про стан проекту, зрозуміти ключові аспекти та виконати аналіз для прийняття обґрунтованих управлінських рішень.

1.2 Методи управління програмними проектами

Серед методів управління програмними проектами вважаємо доцільним зупинитися на таких, як: канбан, скрам та ватерфол.

1.2.1 Канбан

Більшість сучасних систем управління проектами основані на канбан. Завдяки універсальності та простоті канбан, інші методі були витіснені.

Канбан – це метод управління розробкою, який спрямований на виконання завдань вчасно та рівномірно розподіляє навантаження між членами команди. Цей підхід забезпечує прозорість у всьому процесі розробки для всіх учасників. Завдання, по мірі надходження, додаються до окремого списку, з якого кожен розробник може брати задачі, які йому потрібні.

Канбан – наочна система розробки, що показує, що необхідно виробляти, коли і скільки. Метод заснований на однойменному методі в виробничій системі «Тойоти» і ощадливому виробництві.

Канбан заснований на чотирьох основних принципах:

– опора на існуючі методики розробки - канбан починається з існуючих методів розробки і стимулює в них додаткові зміни;

– попередня домовленість про проведення важливих змін - команда розробників повинна враховувати, що постійні зміни - це спосіб поліпшити існуючий процес розробки, проте проведення глобальних змін має великий ризик. Канбан заохочує невеликі і еволюційні зміни.

– повага до існуючого порядку, ролям і обов’язків;

– заохочення ініціативи – заохочення проявів ініціативи кожного розробника.

1.2.2 Скрам

Скрам є одним з можливих способів реалізації гнучкої методології розробки. На відміну від каскадної моделі життєвого циклу ПЗ, відмінною рисою скрам є ітеративний.

Важливу частину скрам методології є система ролей. У скрам прийняті наступні основні ролі:

– власник продукту – представляє інтереси кінцевих користувачів;

– скрам-мастер – той, хто стежить за дотриманням принципів Scrum-розробки і координує команду;

– скрам-команда – програмісти, тестери, аналітики.

– Скрам поділяють на такі етапи розробки:

– створення задач – задачі поділяють по важливості, оцінці об’єму робіт та опис того, як задачу можна протестувати;

– планування спринту, який триває приблизно 2 тижні; команда обирає більш пріоритетні задачі та вирішує яким чином вони будуть вирішуватися;

– робота над спринтом – робота над поставленими задачами;

– тестування та демонстрація продукту;

– ретроспектива та планування наступного спринту.

1.2.3 Ватерфол

Ватерфол – методика управління проектами, яка має на увазі послідовний перехід з одного етапу на інший без пропусків і повернень на попередні стадії.

Ватерфол поділяють на такі етапи розробки:

– аналіз вимог – документування всіх вимог для майбутнього ПЗ;

– проектування програмного забезпечення – створення документу про масштаби і межі проекту; створення документу з вимогами, яким має відповідати створюване програмне забезпечення;

– розробка програмного забезпечення;

– тестування програмного забезпечення;

– технічна підтримка програмного забезпечення.

Таку модель використовують при створенні систем життєзабезпечення, які використовуються у військовій справі, космічних розробках і медицині. Також цю модель можна використовувати для невеликих проектів, але якщо начальному етапі буде допущена помилка, яку виявлять на наступних кроках, ціна виправлення помилки буде дуже високою. Тому рекомендують використовувати даний підхід тільки у тому випадку, якщо всі вимоги до ПЗ прозоро зрозумілі та не будуть змінюватися з плином часу.

1.3 Огляд існуючих систем управління проектами

1.3.1 Trello

Trello – це простий, але класний інструмент управління завданнями (або управління проектами) (рис. 1.1).

Рис. 1.1 – Лого Trello

Інтерфейс Trello дуже простий і мінімалістичний (рис. 1.2), але має все необхідне для команд до 10 членів, включаючи ярлики завдань, вкладення,

управління конкретними завданнями різними користувачами, а також планування.

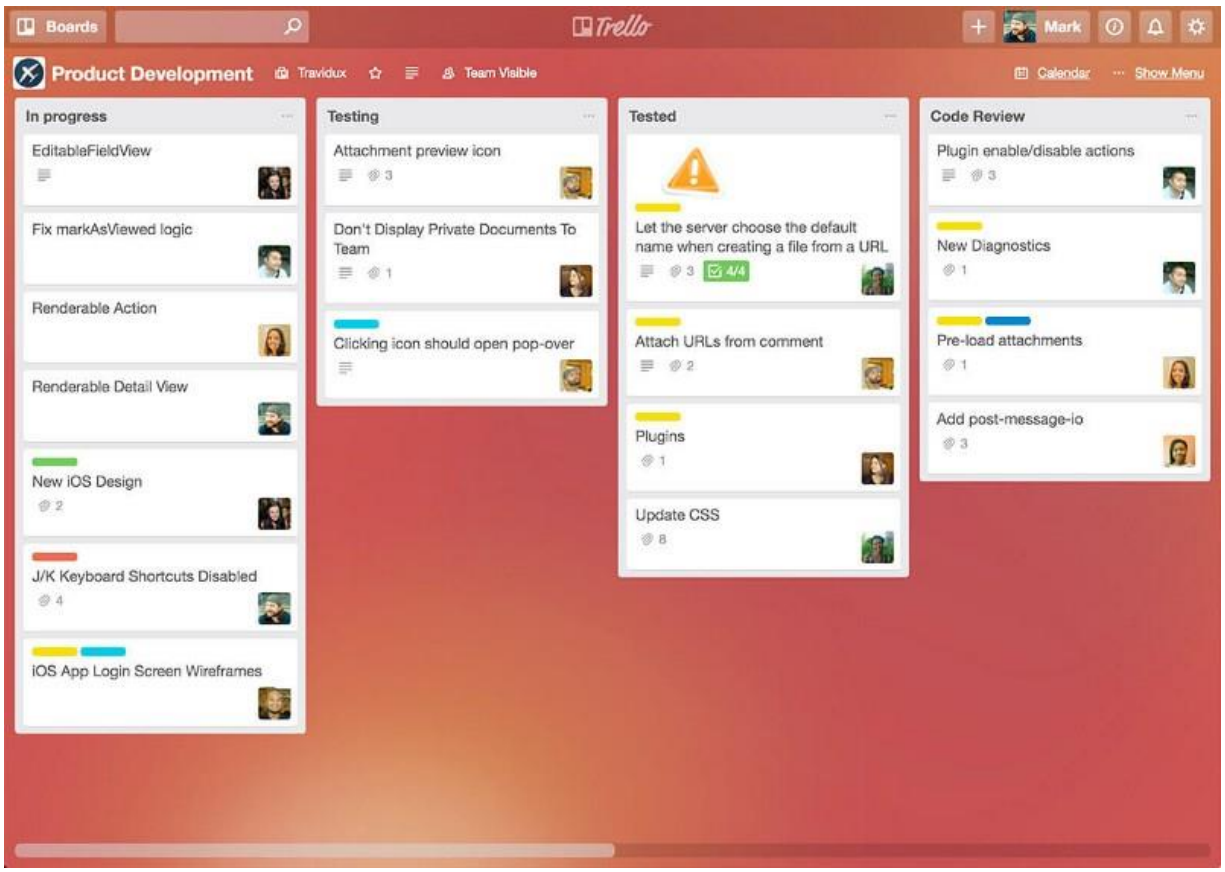

Рис. 1.2 – Інтерфейс Trello

Trello може ідеально підійти як для команд, так і для одинаків. Але варто звернути увагу, що Trello може бути недостатньо для проектів, в яких велика текучка або працює надто багато людей.

Всі основні функції, що надаються Trello, є в безкоштовному плані. Для інтеграції, поліпшення безпеки та підтримки, є плани Business Class і Enterprise, хоча, з розвитком компанії зазвичай починають використовувати щось більш потужне, ніж Trello.

Має такі функції:

– фіксування затраченого часу;
– огляд активних та неактивних проектів різних груп розробників;
– керування складом груп розробників;

– оптимізація роботи груп розробників.

– Має такі переваги:

– прекрасна підтримка з боку розробників;

– максимально оптимізований інтерфейс;

– прив’язка до електронної пошти;

– Також має наступні недоліки:

– відсутня будь-яка локалізація окрім англійської;

– при всій оптимізації, при використанні підключення нижче 3G, завантажує дані довго.

1.3.2 Paymoapp

Paymoapp – конкурент Trello на німецькому та російському ринках (рис. 1.3).

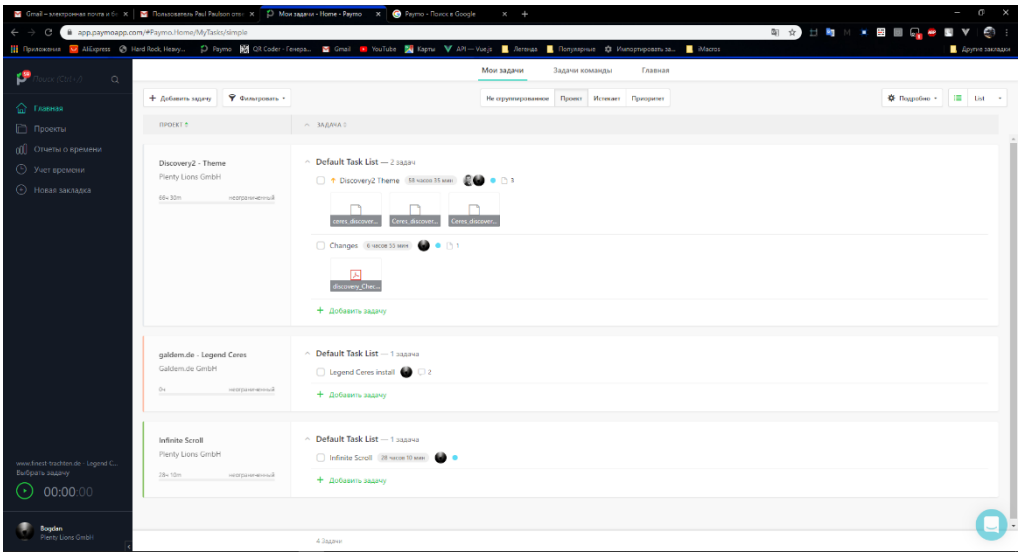

Рис. 1.3 – Інтерфейс Paymoapp

Має повністю аналогічний функціонал Trello, але є декілька відмінностей:

– не має безкоштовного плану роботи;

– має російськомовний варіант інтерфейсу;

– більш класичний дизайн.

Ключовою перевагою Paymoapp є постійна підтримка з боку розробників додатку, тому будь-які помилки вирішуються негайно.

1.4 Веб-додаток як засіб управління проектами

Веб-додаток може бути потужним засобом управління проектами, що надає командам зручність, доступність та спільну платформу для спілкування та керування проектами. Ось деякі способи, які веб-додаток може використовуватися як засіб управління проектами:

Завдання та керівництво проектами: Веб-додатки для управління проектами надають можливість створювати, відстежувати та організовувати завдання для всіх учасників проекту. Вони дозволяють присвоювати завдання, встановлювати терміни виконання, відстежувати прогрес та призначати пріоритети.

Керування командою: Веб-додатки забезпечують засоби для керування командою та спілкування між учасниками проекту. Вони дозволяють обмінюватись повідомленнями, коментувати завдання, спільно працювати над документами та файлами, а також відстежувати активність команди.

Графіки та звіти: Веб-додатки зазвичай надають можливість створювати графіки Ганта, мережеві діаграми та інші візуалізації для планування та моніторингу проекту. Вони також забезпечують генерацію звітів про прогрес, витрати, ризики та інші ключові показники проекту.

Управління ресурсами: Веб-додатки можуть допомагати в управлінні ресурсами проекту, включаючи розподіл завдань між учасниками, відстеження доступності та навантаженості команди, а також планування використання ресурсів.

Спільна робота та колаборація: Веб-додатки забезпечують можливість спільної роботи та колаборації над проектом, незалежно від місцезнаходження

учасників. Вони дозволяють одночасно працювати над завданнями, спільно редагувати документи та забезпечувати взаємодію між учасниками проекту.

Застосування веб-додатку для управління проектами дозволяє зручно та ефективно організувати процеси, спілкування та керування проектами. Він забезпечує зручний доступ до інформації, сприяє спільній роботі та підвищує продуктивність команди.

В кінці 1990-х – початку 2000-х років веб-додатки набули широкого поширення. Одна з важливих переваг побудови веб-додатків для підтримки стандартних функцій браузера полягає в тому, що функціональність може бути реалізована незалежно від операційної системи, на якій працює клієнт. Замість написання окремих версій для різних операційних систем, таких як Microsoft Windows, Mac OS X, GNU/Linux тощо, додаток може бути створений один раз для будь-якої обраної платформи і використовуватися на ній. Однак, різна реалізація стандартів, таких як HTML, CSS, DOM інших, у різних браузерах може створювати проблеми при розробці веб-додатків і їх подальшій підтримці. Крім того, можливість користувачів налаштовувати багато параметрів браузера, таких як розмір шрифту, кольори, відключення підтримки сценаріїв, може вплинути на коректну роботу програми.

Інший підхід, менш універсальний, полягає у використанні технологій, таких як Adobe Flash, Silverlight або Java-аплети, для реалізації повного або часткового інтерфейсу, спрямованого на користувача. Оскільки більшість браузерів підтримує ці технології (часто за допомогою плагінів), Flash- або Java-додатки можуть легко виконуватися. Ці додатки надають більший контроль програмісту над інтерфейсом і дозволяють уникнути багатьох несумісностей у конфігураціях браузерів, хоча несумісність між Java- або Flash-реалізаціями на стороні клієнта може створювати деякі складнощі.

На сьогоднішній день браузери Chrome, Safari та інші популярні браузери не підтримують технологію Adobe Flash. Через архітектурну схожість з традиційними клієнт-серверними додатками, є дебати щодо того,

чи можна віднести такі системи до категорії веб-додатків; існує альтернативний термін «повнофункціональний додаток Інтернету».

Веб-додаток складається з двох частин: клієнтської і серверної, і реалізує концепцію «клієнт-сервер». Клієнтська частина відповідає за користувальницький інтерфейс, формування запитів до сервера і обробку отриманих відповідей. Серверна частина отримує запит від клієнта, виконує обчислення, створює веб-сторінку і надсилає її клієнту за допомогою протоколу HTTP через мережу.

Веб-додаток також може виступати в ролі клієнта для інших сервісів, наприклад, баз даних або інших веб-додатків, розташованих на інших серверах. Яскравим прикладом веб-додатка є система управління вмістом статей у Вікіпедії: велика кількість учасників можуть брати участь у створенні мережевої енциклопедії, використовуючи свої браузери на різних операційних системах (наприклад, Microsoft Windows, GNU/Linux або інші) і не потребуючи додаткового програмного забезпечення для роботи з базою даних статей.

В даний час зростає популярність нового підходу до розробки веб-додатків, відомого як Ajax. З використанням Ajax сторінки веб-додатків не потребують повного перезавантаження, а лише довантажують необхідні дані з сервера, що робить їх більш інтерактивними і продуктивними.

Також зростає популярність технології WebSocket, яка дозволяє створювати двонаправлене з’єднання між клієнтом і сервером без необхідності постійних запитів від клієнта. Це означає, що сервер може відправляти дані клієнту без запиту з його боку. Ця можливість дозволяє динамічно керувати контентом в реальному часі.

Для розробки веб-додатків на стороні сервера використовуються різноманітні технології та мови програмування, які можуть виводити результати до стандартного консольного виводу.

Для реалізації програмного додатку АРМ-менеджера управління проектами на стороні клієнта використовується:

- React – для швидкого рендеру HTML;
- Redux – для зберігання даних в браузері та прозорої архітектури;
- Focal – для пошвидшення роботи додатку;
- SCSS – для оформлення графічного стилю додатку.

Для формування запитів, зберігання даних, створення інтерактивного і незалежного від браузера інтерфейсу:

- nodejs – для отримання користувацьких запитів та взаємодії з БД;
- mongoDB – для зберігання, редагування та видалення даних додатку.

Даний набір програмних засобів обраний не випадково. React, як буде описано в розділі 2, має більшу підтримку зі сторони спільноти, тому розробка буде йти швидше. Redux і Focal обов’язкові в даному ПП, так як потрібні інструменти роботи з великим об’ємом різних даних. SCSS потрібен для налаштування вигляду – це звичайний інструменти будь-якого веб-розробника. Nodejs обов’язковий, так як тільки дана платформа підтримує всі вище перераховані модулі програми. MongoDB – оптимальна NoSQL СУБД для додатку на Nodejs.

РОЗДІЛ 2 ІНСТРУМЕНТИ РОЗРОБКИ

2.1 Формалізація задач, що вирішуються АРМ з метою організації КПП

Додаток має вирішувати наступні задачі:

- збереження і зміна даних про задачі, проекти та користувачів;
- фіксування вирішених/відмінених/прострочених задач та проектів;
- розподілення задач по командам;
- виділення часу на задачі;
- управління активними проектами та задачами.

2.2 Розробка інформаційної моделі для ПЗ КПП

Для зберігання даних для ПЗ КПП використовується БД MongoDB. Оскільки ця БД проста у використанні (оскільки в ній відсутні складні запити та навіть схема) і в інтеграції з додатком. Так само MongoDB заснована на колекціях, через що вона легко масштабована і структура кожного об’єкта прозоро зрозуміла.

Дані в БД зберігаються на кластері, який можна створити за допомогою інтерфейсу на сайті `https://cloud.mongodb.com/`.

2.2.1 Створення та підключення БД до ПЗ КПП

Для створення свого кластера спочатку потрібно зареєструватися в системі. Після реєстрації потрібно натиснути на кнопку «Build a New Cluster» (рис 2.1). Далі вибрати регіон (рис 2.2), ім’я кластера (рис 2.3) і натиснути на кнопку «Create Cluster» (рис 2.4).

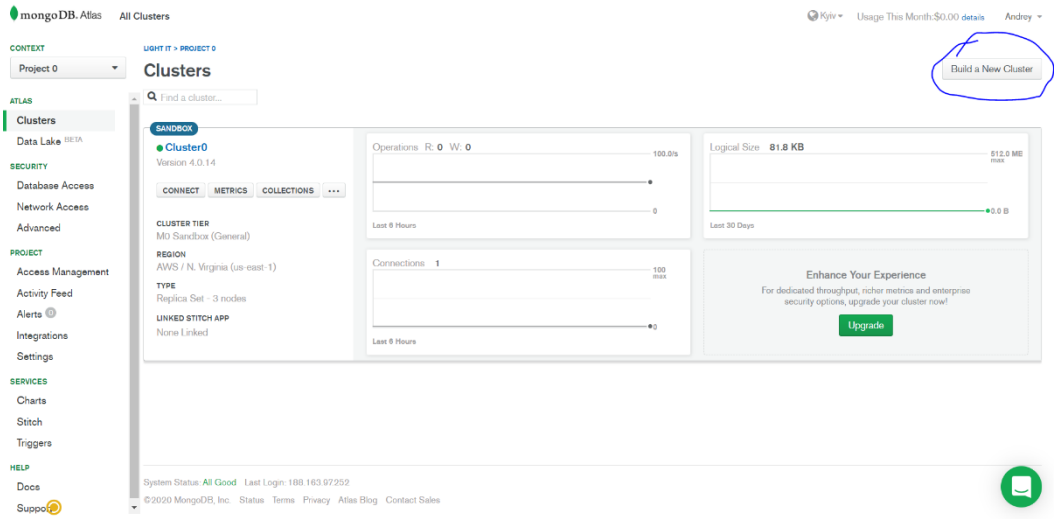

Рис 2.1 – Створення нового кластеру за допомогою інтерфейсу MongoDB

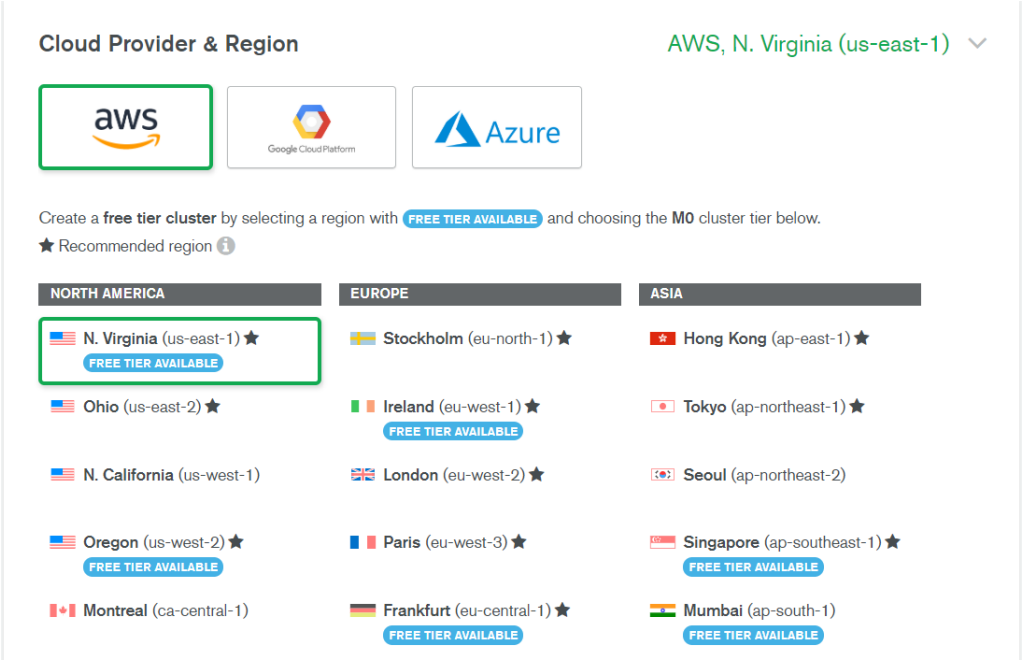

Рис 2.2 – Вибір регіону

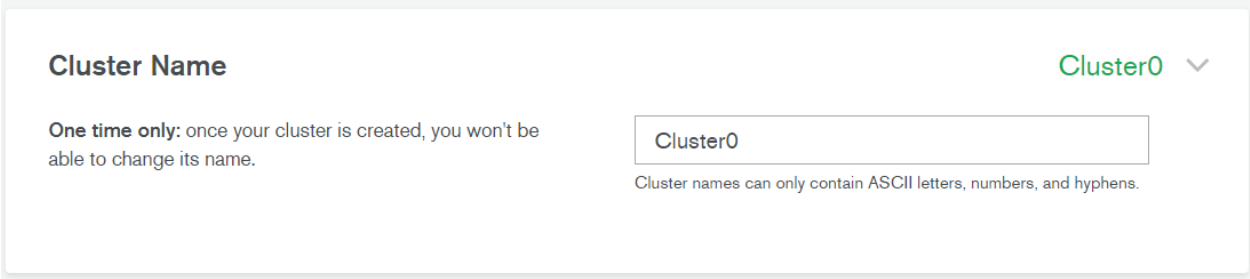

Рис 2.3 – Поле для створення імені кластеру

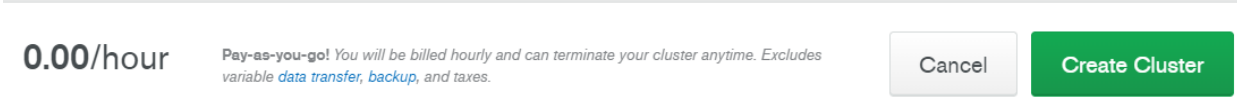

Рис 2.4 – Кнопка для завершення конфігурації та створення кластеру

Після створення кластеру можна побачити його властивості (рис 2.5). На рисунку 2.5 можна побачити кількість операцій, кількість використовуваного об’єму та кількість підключень до бази даних.

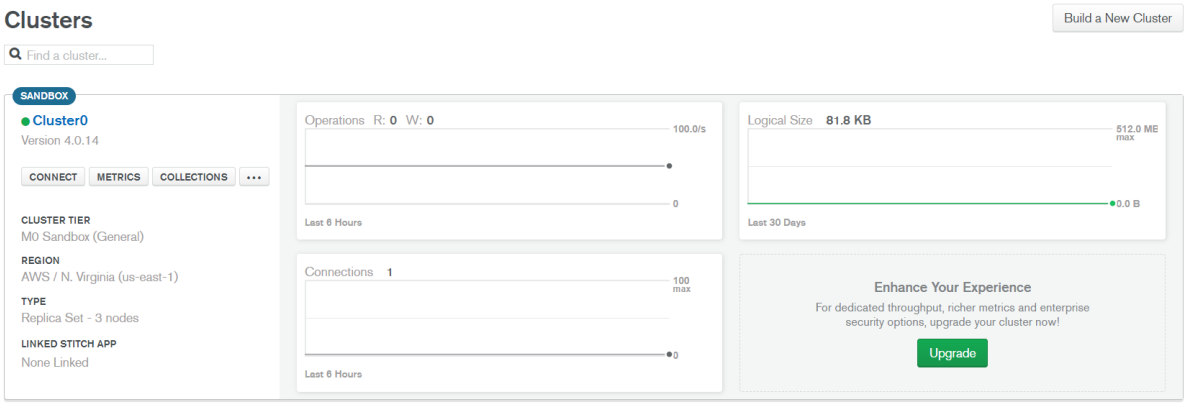

Рис 2.5 – Властивості кластеру

Наступний етап – це етап підключення бази даних до ПЗ КПП. Для цього необхідно натиснути на кнопку «CONNECT» у властивостях кластеру та побачити модальне вікно з прикладами підключень до бази даних (рис 2.6).

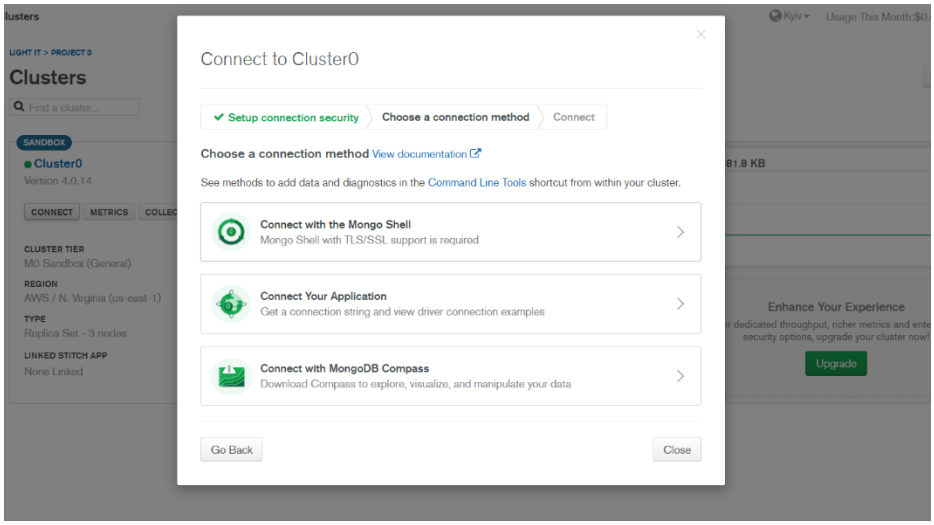

Рис 2.6 – Приклади підключень до MongoDB

Для ПЗ КПП потрібно натиснути на пункт «Connect your application».

У наступному вікні вибираємо технологію, яку використовує додаток на серверній частині та її версію (рис 2.7).

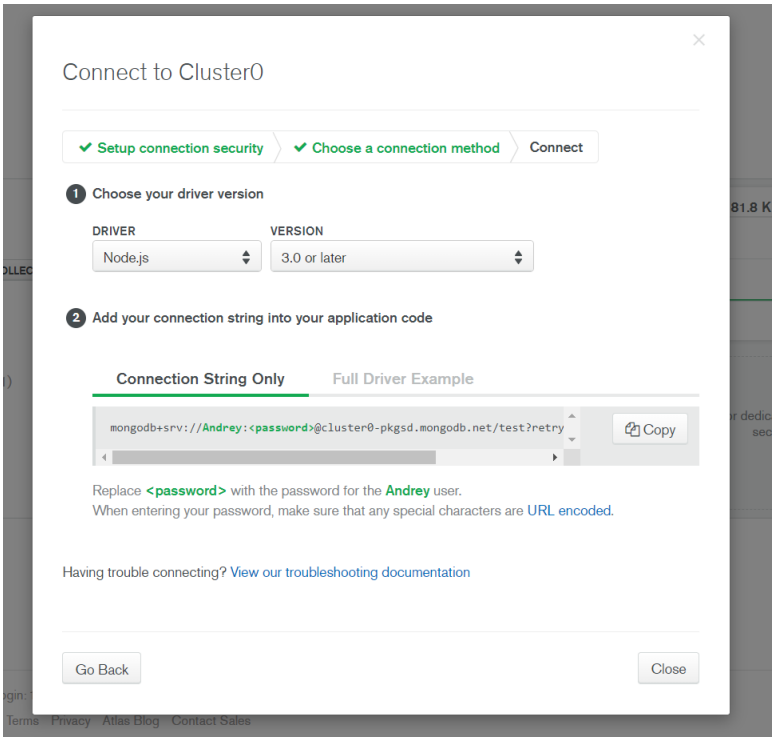

Рис 2.7 – Меню підключення до кластеру

У цьому ж вікні можна побачити строку підключення – mongodb+srv://Andrey:f7vh%wCVhj#nW[{@cluster0pkgsd.mongodb.net/test?retryWrites=true&w=majority. У лістингу 2.1 відображений код, який буде використаний на серверній частині додатку для підключення кластеру (додаток А).

Лістинг 2.1 Код підключення кластеру

```
const uri =
mongodb+srv://Andrey:f7vh%wCVhj#nW[{@cluster0pkgsd.mongodb.n
et/test?retryWrites=true&w=majority;
MongoClient.connect(uri, { useNewUrlParser:
true }).then(client => {
     /* Інший програмний код */
}
```

2.2.2 Створення колекцій з даними

Для додатку КПП можна виділити декілька сутностей, з якими потрібно працювати:

- board – дошка;
- list – список з задачами;
- task – задача;
- user – користувач.

Для наглядного прикладу можна створити схему даних.

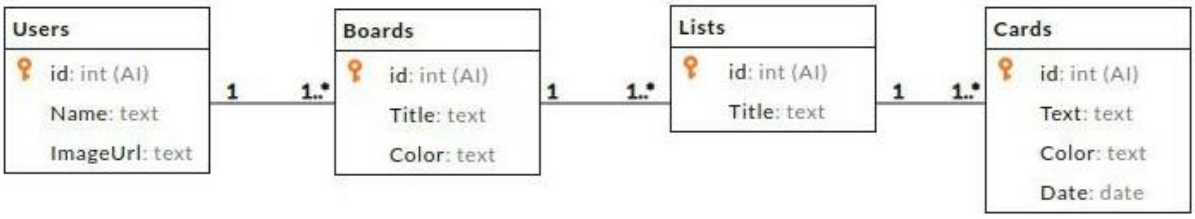

Рис 2.8 – Схема даних для БД КПП

Users – список користувачів, з ідентифікатором, іменем та посиланням на аватар користувача.

Boards – список дошок, що містить ідентифікатор, ім’я та колір кожної дошки.

Lists – список згрупованих задач, що містить ідентифікатор та заголовок.

Cards – список задач, що містить ідентифікатор, текст, колір задачі та термін виконання.

В MongoDB це виглядає наступним чином.

У властивостях кластеру на рис 2.5 потрібно натиснути на кнопку «COLLECTIONS», щоб побачити список створених колекцій (рис 2.9).

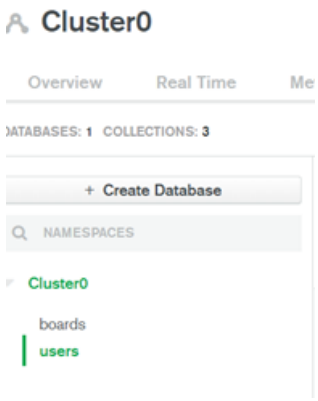

Рис 2.9 – Список створених колекцій

Якщо вибрати колекцію «Users», то буде відображений список користувачів (рис 2.10). На рисунку 2.11 відображено як зберігається користувач.

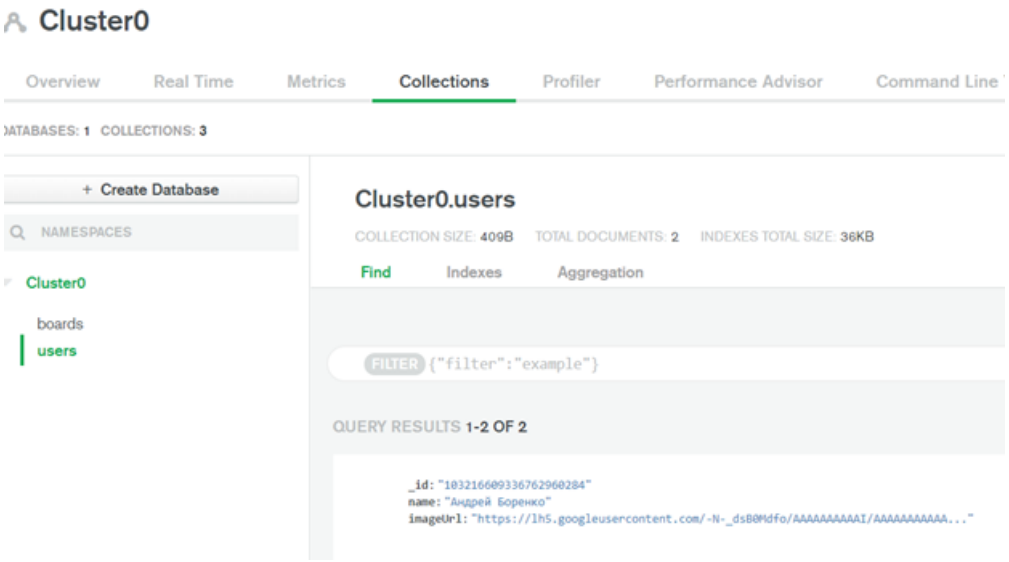

Рис 2.10 – Список користувачів


```
_id: "103216609336762960284"
name: "Андрей Боренко"
imageUrl: "https://lh5.googleusercontent.com/-N-_dsB0Mdfo/AAAAAAAAAAI/AAAAAAAAAAA..."
```


Рис 2.11 – Об’єкт користувача

Якщо вибрати колекцію «Boards», то буде відображений список створених дошок (рис 2.12). На рисунку 2.13 відображено як зберігаються дошки. Можна помітити, що дошка зберігає списки задач та задачі. Цей формат зберігання даних робить більш простим роботу з даними.

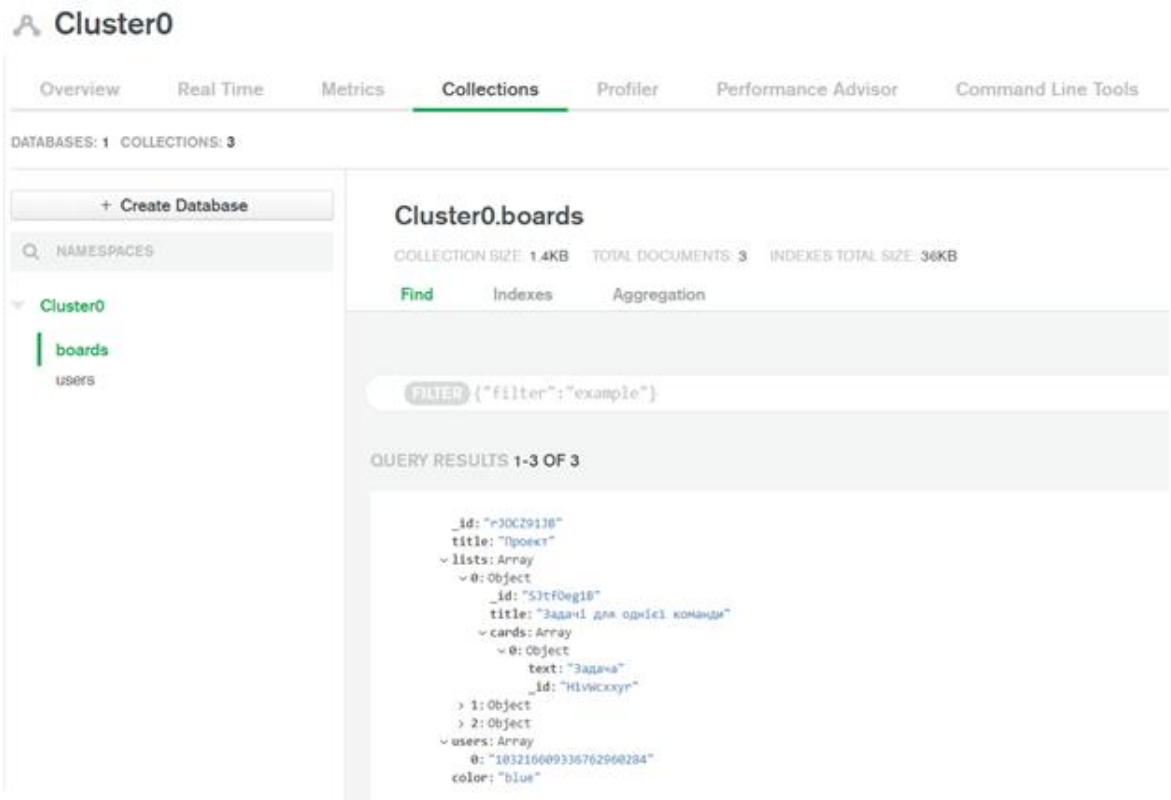

Рис 2.12 – Список дошок

Рис 2.13 – Об’єкт дошки

2.3 Розробка архітектури ПЗ КПП

ПЗ КПП можна поділити на чотири основні модулі:

– Board – модуль для роботи з дошками;

– List – модуль для роботи з групою карток (задач);

– Card – модуль для роботи з карткою (задачею);

– Shared – модуль, який містить компоненти, які неможливо або нема сенсу відокремлювати.

На рисунку 2.14 відображена схема взаємодії цих модулів.

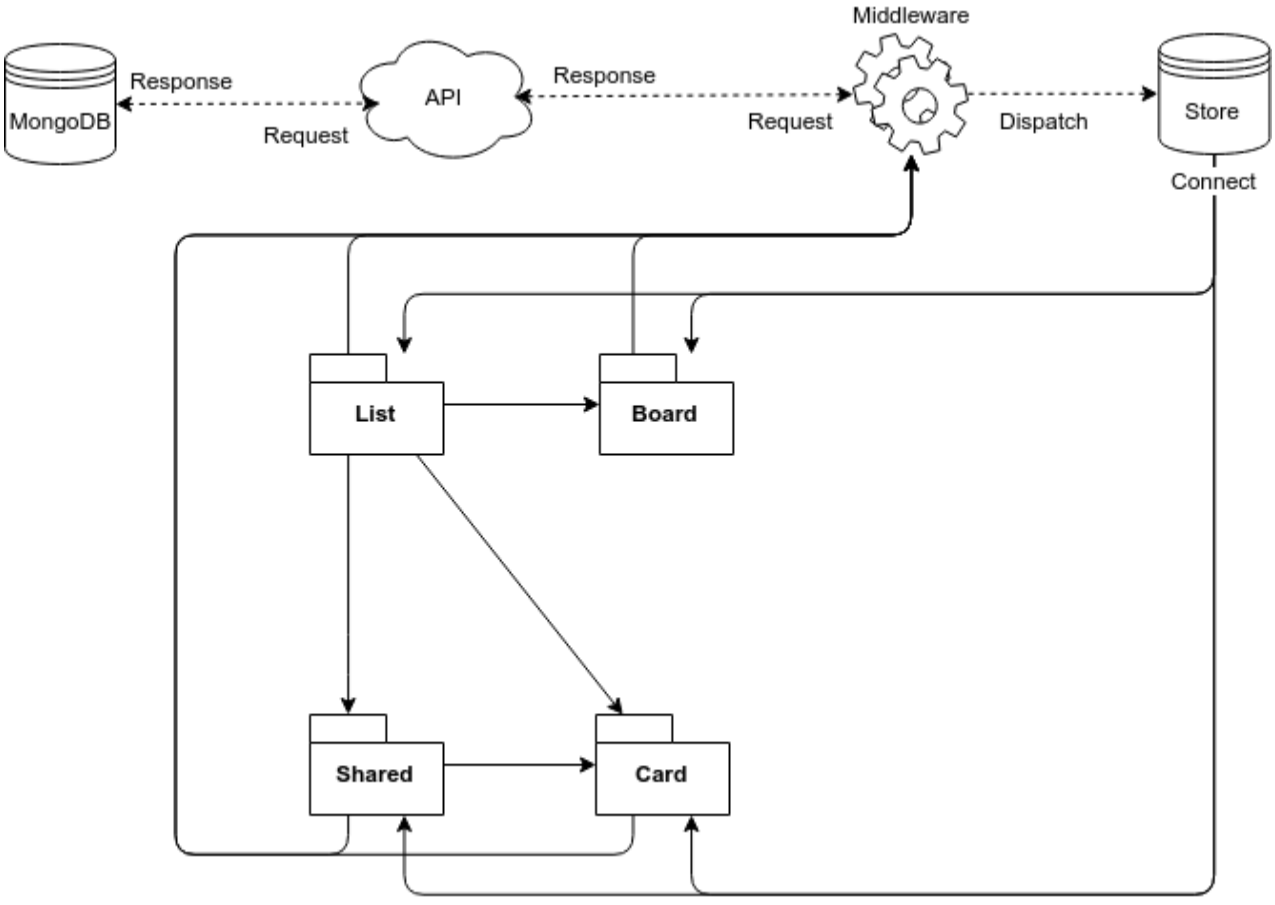

Рис. 2.14 – Схема зв’язку модулів

Взаємодія даних на middleware-частині проекту. Кожен модуль може відправити запит на middleware, і вже там проходить запис чи витяг даних з БД, після чого на middleware приходить відповідь з даними в вигляді

документу JSON. Після чого дані зберігаються у Store та передаються в модулі для подальшої роботи з ними.

Для кожного окремого модулю була створена діаграма компонентів, яка показує розбиття програмної системи на структурні компоненти та залежності між ними. На рисунку 2.15 відображена діаграма компонентів для модуля Board.

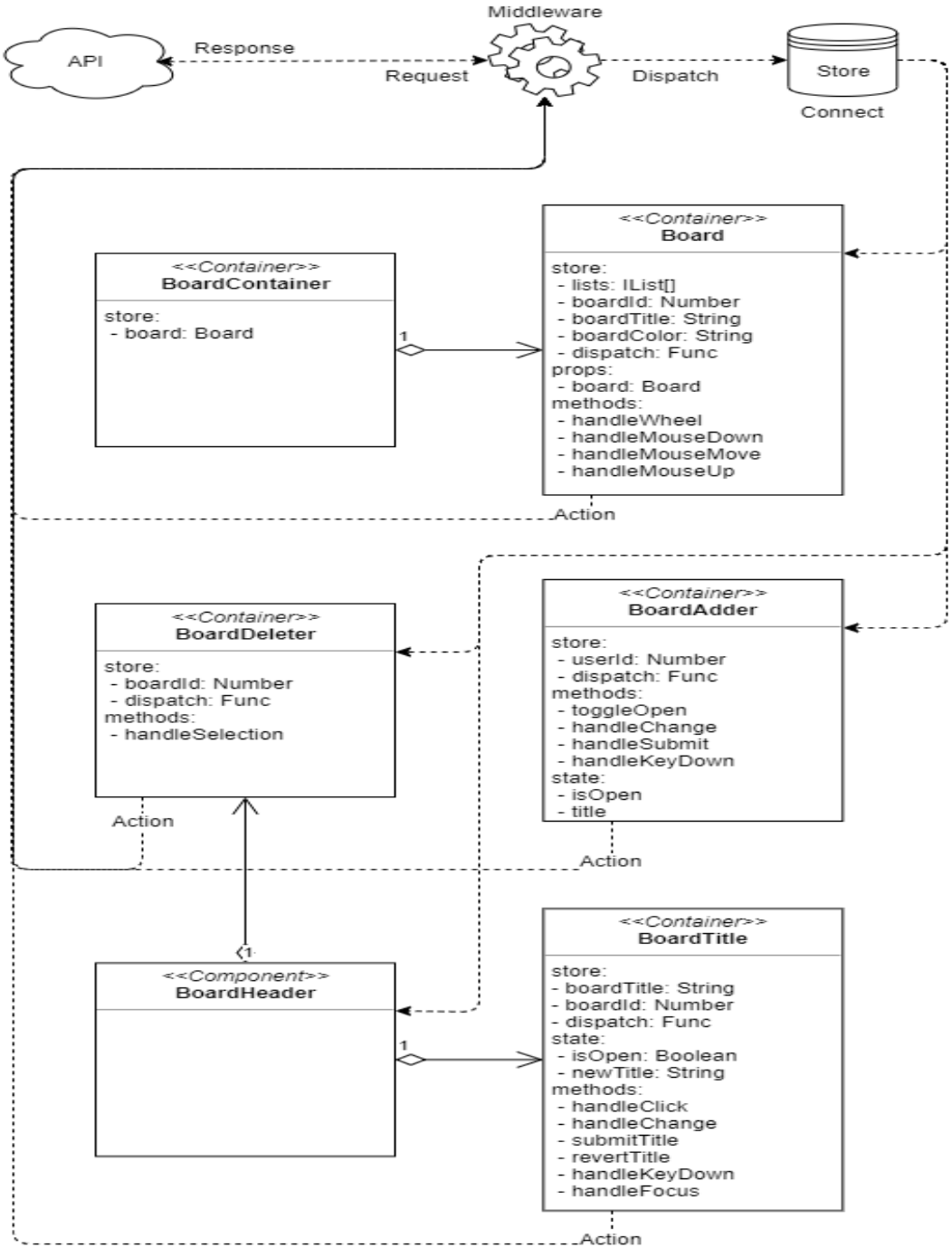

Рис. 2.15 – Діаграма компонентів Board модуля

Згідно з задачами дошка має компоненти для її створення, видалення та відображення.

Компонент для створення дошки – BoardAdder.

Компонент для видалення дошки – BoardDeleter.

Компоненти для відображення дошки: BoardContainer; Board; BoardHeader; BoardTitle.

Далі, на рисунку 2.16 відображена діаграма компонентів для модуля List. Основними компонентами для цього модулю є List і ListHeader для відображення списку карточок та ListAdder для створення нових списків.

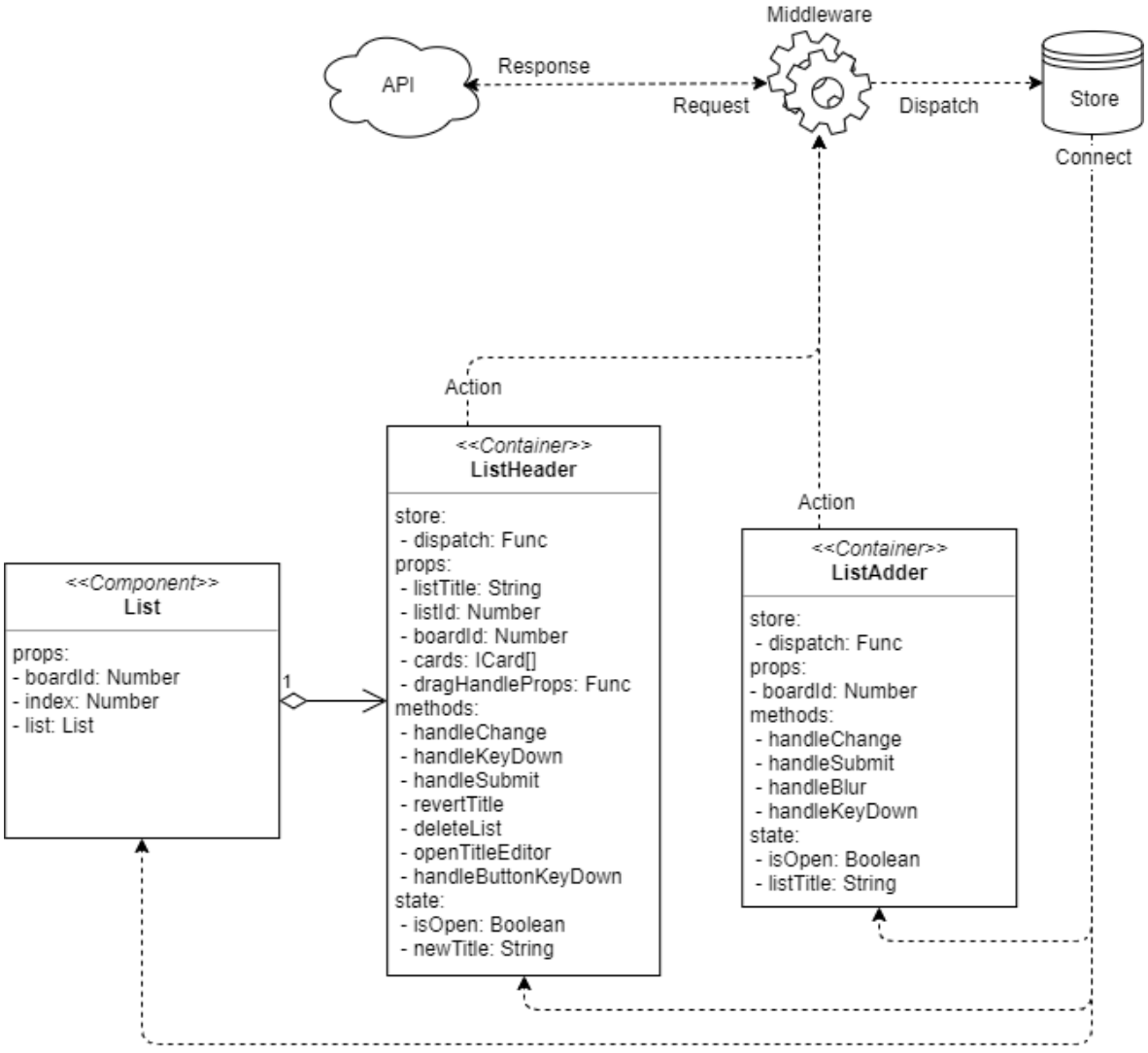

Рис. 2.16 – Діаграма компонентів List модуля

На наступному рисунку 2.17 відображена діаграма компонентів для модуля Card. Основними компонентами для цього модулю є Cards, Card та CardBadges для відображення карточок. Компонент CardAdder потрібен для створення нових карток.

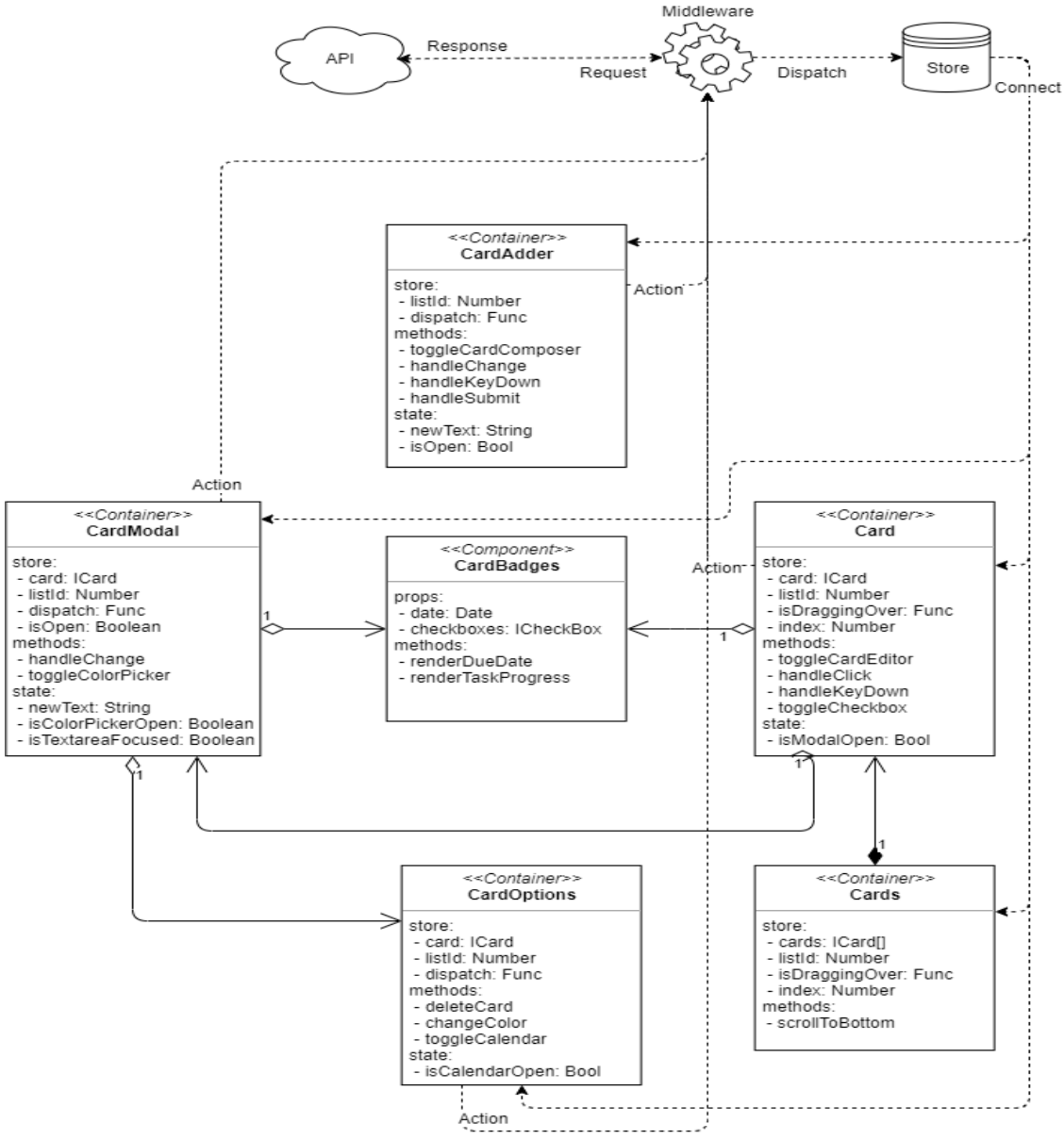

Рис. 2.17 – Діаграма компонентів Card модуля

Компонент CardOptions відповідає за функціонал картки, а CardModal відображає користувачу доступні операції над карткою.

На наступному рисунку 2.18 відображена діаграма компонентів для модуля Shared. Тут знаходяться компоненти для роботи з календарем – Calendar, вибором кольору – ColorPicker, загальний компонент для відображення шапки – Header, та компонент, який обробляє натиснення мишкою за межою компоненту, який використовує ClickOutside.

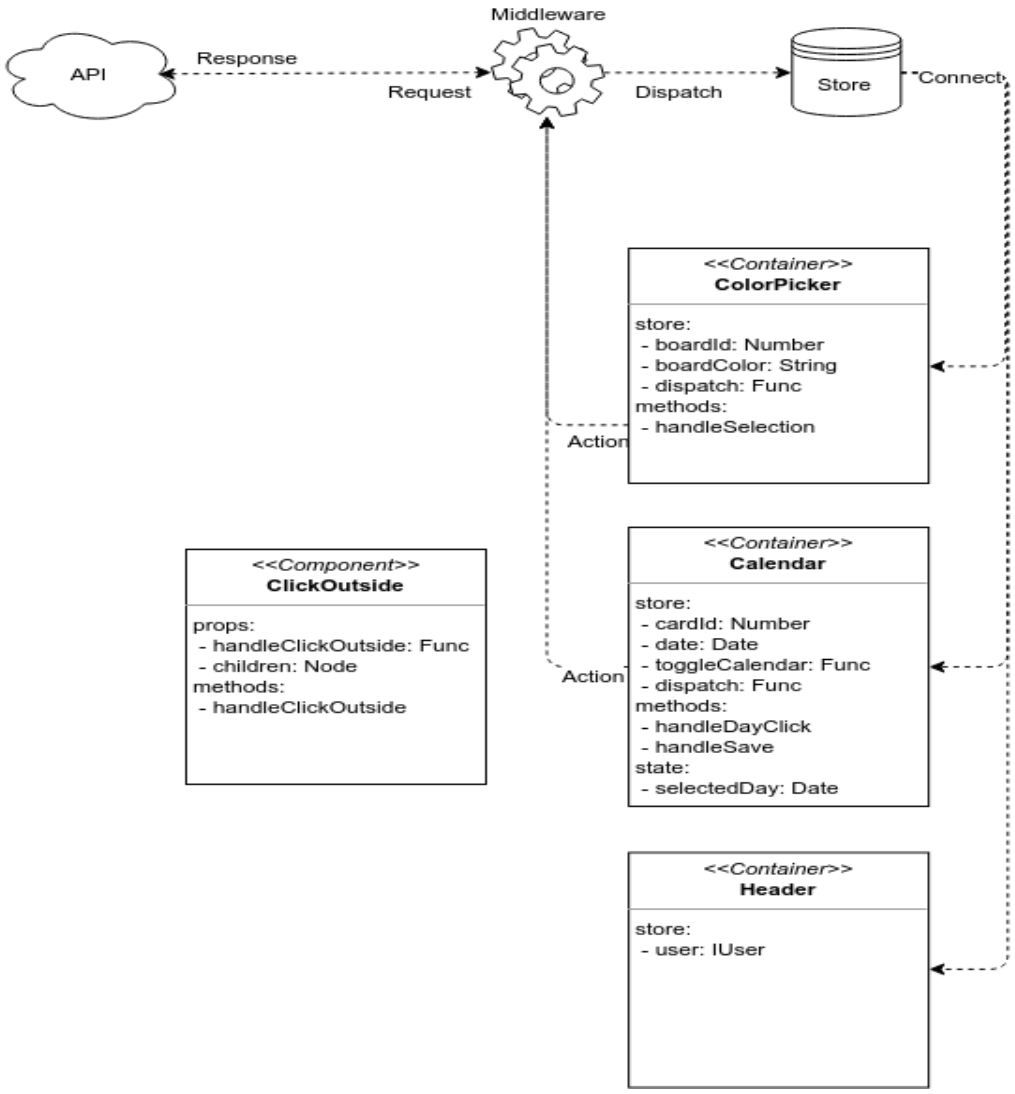

Рис. 2.18 – Діаграма компонентів Shared модуля

Таким чином, у цьому розділі були розглянуті та формалізовані задачі, що вирішуються АРМ з метою організації КПП. Була проаналізована СУБД MongoDB, розроблена інформаційна модель для ПЗ КПП. Була проведена розробка архітектури ПЗ КПП: створена діаграма зв’язку модулів системи; створена діаграма компонентів для кожного окремого модулю.

РОЗДІЛ 3 РОЗРОБКА ДОДАТКУ АРМ КЕРІВНИКА ПРОГРАМНИМИ ПРОЕКТАМИ

3.1 Аналіз і вибір інструментальної системи реалізації КПП

Додатки такого типу повинні мати наступні властивості:

– зручний та максимально оптимізований дизайн – для зручної і при цьому швидкої роботи;

– оптимізовану та добре організовану БД для швидкої роботи з даними, а також їх захист.

Для цього в першу чергу треба вибрати фреймворк (або його відсутність) при розробці фронтальної частини проекту та СУБД (і мову запитів разом з нею).

Після треба підібрати модулі (npm-пакети, бібліотеки), які допоможуть в розробці (dev-залежності) та ті модулі, які стануть частиною проекту.

3.1.1 NodeJS

Node або Node.js – це програмна платформа, яка використовує двигун V8 для перетворення JavaScript з вузькоспеціалізованої мови на загальнопризначену мову. Використання цієї платформи необхідне – саме завдяки ній можливе використання більшості сучасних веб-фреймворків

Рис. 3.1 – Лого NodeJS

Дана платформа – наріжний камінь будь-якого сучасного веб-проекту крім самих спеціалізованих (тобто тих, які не використовують javascript).

Чому саме NodeJS? Немає іншого настільки потужного інструменту для роботи зі скриптовою мовою, а без неї не працюють веб-додатки. Нижче приведені залежності різних фреймворків, бібліотек, СУБД та іншого від даної платформи.

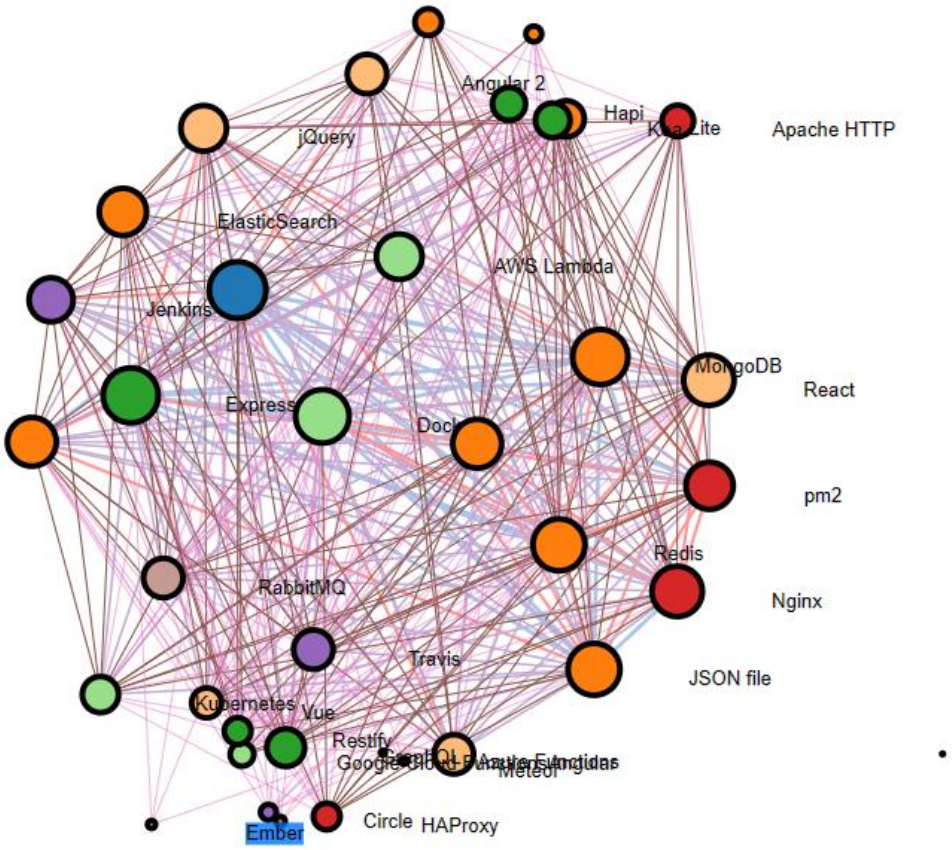

Рис. 3.2 – Залежності від NodeJS

Деякі засоби можуть працювати і без NodeJS:

– Apache HTTP це серверне ПО;

– JQery – незалежна бібліотека, яку можна підключати навіть через посилання на офіційний ресурс;

– JSON – взагалі просто формат файлів.

Але NodeJS може використовувати ці всі засоби в одному проекті, тому вибір простий. Нижче представленні основні принципи роботи з даною платформою.

На лістингу 3.1 представлено створення і запуск HTTP-сервера на Node.js, що видає Hello, world!

Лістинг 3.1 Створення і запуск HTTP-сервера

```
// Завантажуємо модуль http
const  http  =  require ( 'http' );
// Створюємо web-сервер з обробником запитів
const  server  =  http . createServer (( req ,  res )  =>  {
  console . log ( 'Початок обробки запиту' );
  // Передаємо код відповіді і http-заголовки
  res . writeHead ( 200 ,  {
    'Content-Type' :  'text / plain; charset = UTF-8 '
  });
  res . end ( ' Hello world! ' );
});
// Запускаємо web-сервер
server . listen ( 3000 ,  '127.0.0.1' ,  ()  =>  {
  console . log ( 'Сервер запущено
http://127.0.0.1:3000/' );
});
```

Інший приклад скрипта у лістингу 3.1, що створює TCP-сервер, який прослуховує порт 8000 і виводить на екран все, що вводить користувач.

Лістинг 3.2 Скрипт створення TCP-сервера

```
const  net  =  require ( 'net' );
const  server  =  net . createServer ( stream  =>  {
  stream . setEncoding ( 'utf8' );
  stream . write ( 'hello \ r \ n' );
  stream . addListener ( 'data' ,  data  =>  {
    stream . write ( data );
  });
});
server . listen ( 8000 ,  'localhost' );
```

Додаткові пакети сторонніх розробників

До складу Node.js входить власний установник пакетів npm . Установка проводиться за допомогою команди: npm install <packagename>.

Всі доступні для установки пакети і їх короткий опис: npm search.

Цією ж командою можна робити вибірковий пошук пакетів.

3.1.2 Вибір фреймворка

На даний момент на ринку веб-фреймворків представлено багато різних продуктів. Точніше не ринку – майже всі фреймворки безкоштовні. Тому до вибору треба підійти прискіпливо.

Завдяки підтримці та популярності, вибір буде вестись між трьома веб-фреймворками: React, Vue і Angular.

Рис. 3.3 – Логотипи суперників

По популярності React на першому місці, а це означає, що він має:

- саму велику спілку розробників;
- велику кількість модулів npm;
- чудову підтримку в абсолютній більшості браузерів світу.

Далі треба розглянути продуктивність.

Існує річ під назвою DOM, яку можна розуміти як інтерфейс користувача, тобто користувальницький інтерфейс програми. DOM змінюється щоразу, коли оновлюється інтерфейс користувача. Це відображає зміни, внесені в додаток.

Він може використовуватися двома способами, як віртуальний DOM, так і реальний DOM. На них сильно впливає робота фреймворка/бібліотеки.

Angular

Цей Javascript-фреймворк використовує реальний DOM.

Цей підхід має цілий ряд недоліків, а саме:

- дуже важкий та незручний дебагінг програми;
- мала продуктивність;
- великі ресурсовитрати.

React

Ця бібліотека Javascript використовує віртуальний DOM. Це не специфічний для браузера і легкий функціонал. Він надається в пакеті React безкоштовно і виключає проблеми повільної роботи реального DOM.

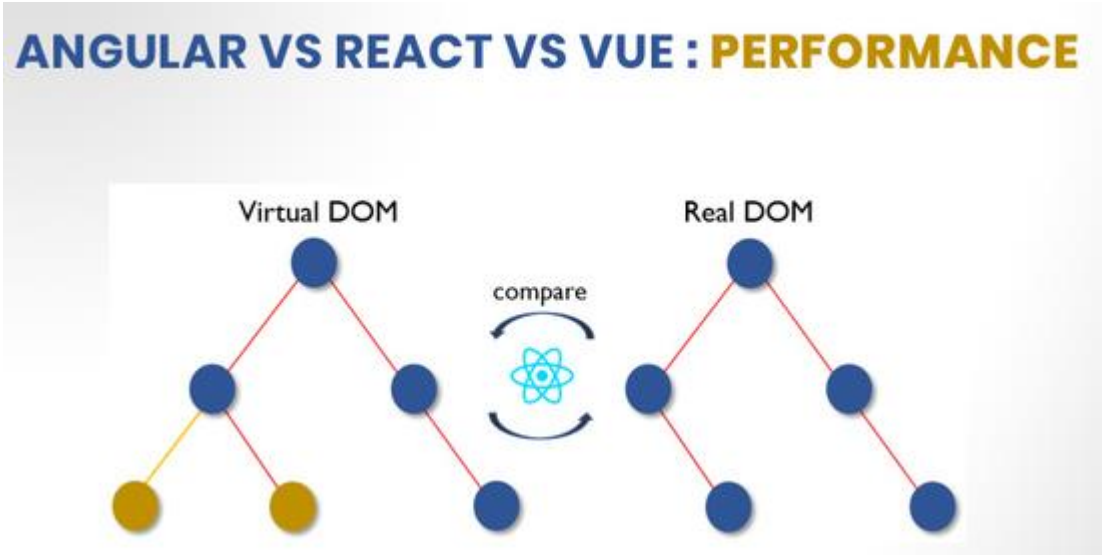

Рис. 3.4 – Графічне представлення концепцій віртуального та реального DOM

Vue

Даний фреймворк самий молодий з даної трійці та створений людиною, яка брала участь в розробці і React, і Angular, тому даний фреймворк має підтримку віртуального DOM.

Іншим параметром э розмір фреймворку. Angular самий важчий завдяки влаштованій підтримці тестів та власній CLI з командами - 500 КБ. React

набагато легший - 100 КБ. Більшу частину функціоналу потрібно вносити в залежності. Vue має всього 80 КБ – в основному модулі тільки саме необхідне.

Angular логічно використовувати для масштабних проектів типу систем документообігу, онлайн-редакторів документів та тому подібного.

Іншим параметром є строгість архітектури додатку.

Angular дуже чутливий до змін в стандартній ієрархії файлівЮ жтому часто для вирішення невеликих задач потрібно створювати повноцінні розділи проекту, що погано впливають на продуктивність додатку. Vue та React мають властивості бібліотек, тому їм не так важлива ієрархія файлів, що дає розробнику більше свободи, але може вплинути на стабільність роботи додатку – нові рішення не завжди краще стандартних, закладених розробниками фреймворку.

На основі дослідження, було прийняте рішення використовувати саме React. Він має саму велику спільноту, тому майже на будь яке питання по фреймворку вже найдена відповідь, а якщо ні, то на будь-якому форумі інші користувачі швидко дадуть пораду. Також завдяки популярності, він самий стабільний в плані підтримки браузерами. І головне, він надає достатню свободу розробнику.

3.1.3 Модулі

Будь-який сучасний веб-проект складається з модулів більше ніж на 50%, так як веб-додатки працюють на однакових платформах, зручно постійно не придумувати велосипед та використовувати уже створене оптимальне рішення для задачі.

Саме для цього створені менеджери пакунків для веб-додатків на NodeJS. Самий популярний з них (і фактично єдиний) це npm.

npm – це менеджер пакетів для JavaScript, який є стандартним для виконавчого середовища Node.js. Він складається з клієнтського командного рядка, також відомого як npm, і реєстру npm, який містить публічні та приватні

пакети. Реєстр npm є інтернет-базою даних, до якої можна отримати доступ через клієнт npm. Перегляд та пошук доступних пакетів можна здійснювати через веб-сайт npm. Компанія npm, Inc. відповідає за управління менеджером пакетів та реєстром.

Рис. 3.5 – Лого npm, Inc

Для роботи був обраний саме цей менеджер, так-як немає необхідності в більш функціональних аналогах типу yarn. Всі менеджери використовують базу npm, але додають свій особливий функціонал, наприклад: хешування пакетів; перевірка хеш-сум; стиснення пакетів.

В даному же проекті не велика кількість залежностей, тому використовувати більш об’ємний доволі неоптимально.

Для роботи з npm потрібно знати основні команди CLI, які представлені у додатку Г.

У даного проекту, як і у будь якого сучасного ВЕБ-додатку, є цілий список npm-залежностей, який зображений на рисунку 3.6-3.7.

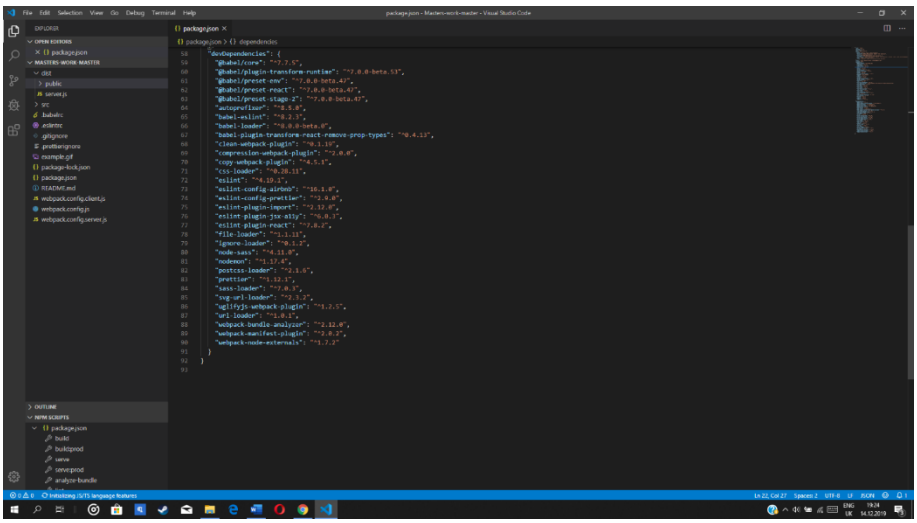

Рис. 3.6 – devDependencies

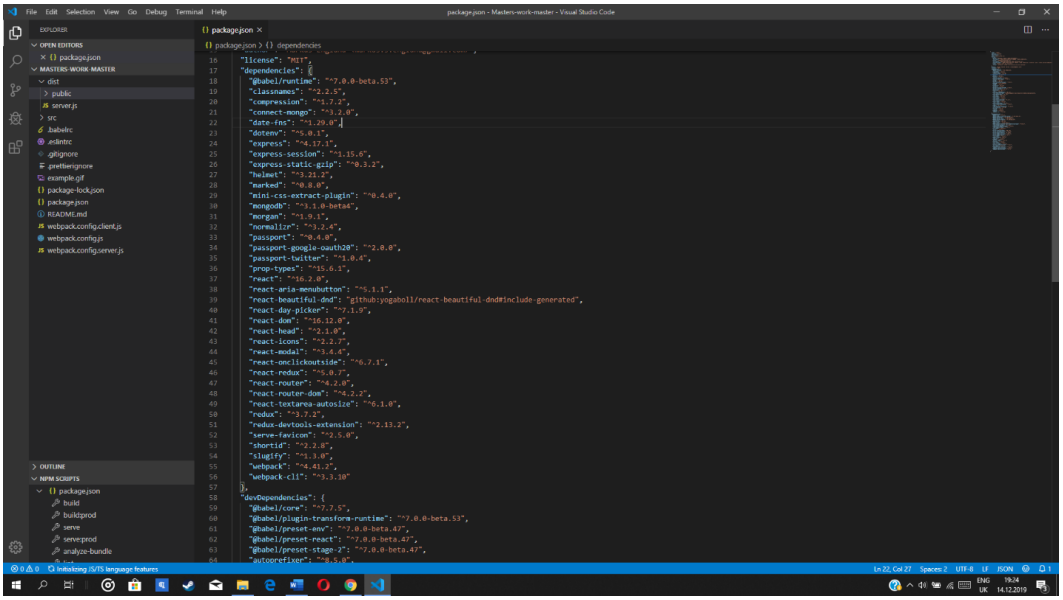

Рис. 3.7 – dependencies

Кожен модуль в цьому списку обраний не просто так. Всі devDependencies допомагають в розробці – форматують код, перевіряють його на помилки, додають префікси до функцій, збирають проект та інше. Dependencies напряму впливають на саме функціонування додатку – це просто бібліотеки з функціями, які використовує програма.

Babel. Хоч час не стоїть на місці, багато користувачів використовують застаріле ПО. Це стосується і браузерів. В кожному оновленні того же Chrome додаються нові функції, виправляються помилки. Але є особливі оновлення – це оновлення підтримки ECMAScript. В кожному новому стандарті цієї мови додаються нові функції, що потім вливается в зміни в JS. Підтримка цих функцій – показник свіжості браузера. Але не завжди у користувача є доступ до нового браузеру, тому для таких ситуацій розробники використовують Babel – складальник, який створює з коду нового стандарту код старого стандарту. Звичайно, такий код важко назвати оптимальним, зате він працює на кожному пристрої. І деякі функції він відключає.

Наприклад, стрілкові функції, які вказані в ES6, перетворюються на регулярні декларації функцій. Нестандартний синтаксис JavaScript, такий як JSX, також може бути перетворений.

Рис. 3.8 – Лого Babel

Babel надає поліфіли для підтримки функцій, які повністю відсутні у середовищах JavaScript.

Webpack – просто складач, який перетворює проект на різних фреймворках, з залежностями та медіафайлами в єдиний проект, який можна запустити на сервері. Webpack - один з найпотужніших і гнучких інструментів для збірки frontend.

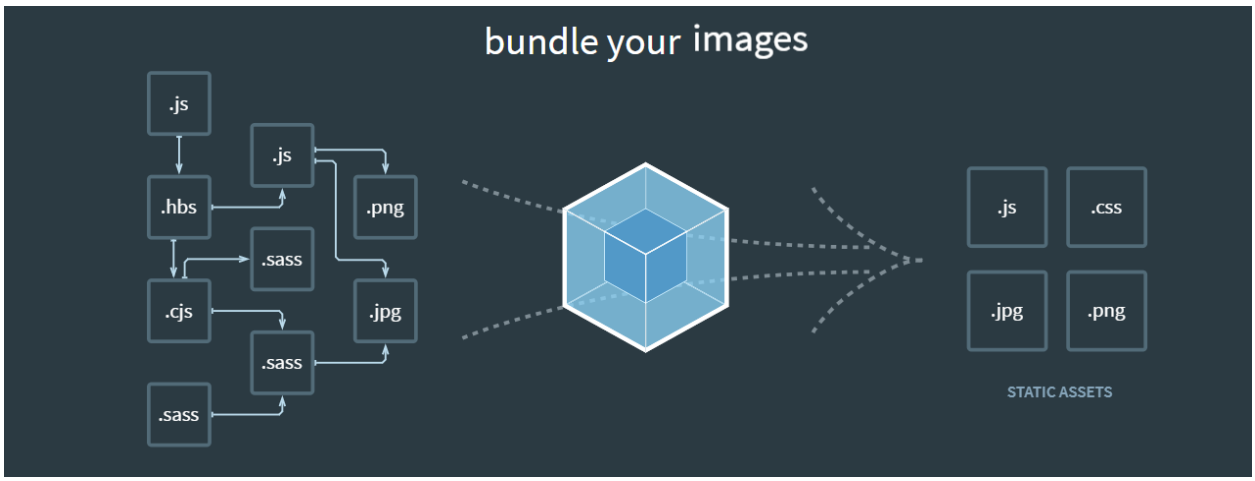

Рис. 3.9 – Робота Webpack

Webpack використовується для керування залежностями та створення графа залежностей, що дозволяє веб-розробникам використовувати модульний підхід при розробці веб-додатків. Його можна використовувати через командний рядок або налаштовувати за допомогою файлу конфігурації

під назвою webpack.config.js. Цей конфігураційний файл використовується для визначення правил, плагінів та інших параметрів для проекту. Webpack може бути розширений за допомогою правил, що дозволяють розробникам виконувати спеціальні завдання під час об'єднання файлів.

Рис. 3.10 – Лого Webpack

Webpack використовує механізм розділення коду на вимогу, використовуючи названі частини. Група 39 технічного комітету ECMAScript працює над стандартизацією функції "динамічного імпорту", яка дозволяє завантажувати додатковий код за запитом. Крім того, Webpack має вбудований сервер розробки під назвою Webpack Dev Server, який може використовуватися як HTTP сервер для обслуговування файлів під час розробки. Він також підтримує гарячу заміну модулів, що дозволяє безперервно оновлювати змінені модулі без перезавантаження всього додатку.

Форматування коду – важлива частина розробки. Код повинен буди правильним – з відступами, адекватними назвами змінних, оптимальними рішеннями, наприклад використання стрілкових функцій в колбеках та інше. Самим зручним і популярним інструментом для цього є ESLint.

ESLint – це інструмент статичного аналізу коду JavaScript, призначений для виявлення проблемних частин у програмному коді. Його було розроблено Ніколасом К. Закасом у 2013 році. Правила в ESLint можна настроювати, а також створювати та завантажувати спеціальні правила.

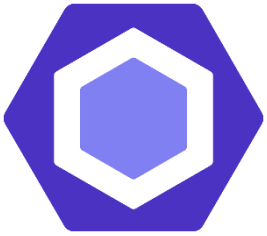

Рис. 3.11 – Лого ESLint

ESLint охоплює як якість коду, так і питання стилів кодування. Він підтримує поточні стандарти ECMAScript та включає експериментальний синтаксис, що випливає з чернеток для майбутніх стандартів. Крім того, ESLint може обробляти код на JSX або TypeScript, якщо використовується відповідний плагін або транспілятор.

В лістингу 3.3 представлені параметри лінту для даного проекту.

Лістинг 3.3 Параметри ESLint для АРМ КПП

```
{
  "env": {
    "browser": true,
    "es6": true,
    "node": true,
    "jest": true
  },
  "extends": ["airbnb", "prettier", "prettier/react"],
  "parser": "babel-eslint",
  "parserOptions": {
    "ecmaFeatures": {
      "jsx": true
    },
    "sourceType": "module"
  },
  "rules": {
    "react/prefer-stateless-function": 0,
    "react/no-array-index-key": 0,
    "no-underscore-
dangle": ["error", { "allow": ["_id", "_json"] }],
    "jsx-a11y/no-autofocus": 0,
    "jsx-a11y/anchor-is-valid": [
      2,
      {
        "components": ["Link"],
        "specialLink": ["to", "hrefLeft", "hrefRight"],
```

```
        "aspects": ["noHref", "invalidHref", "preferButton"]
      }
    ],
    "react/require-default-props": 0,
    "react/forbid-prop-types": 0,
    "import/prefer-default-export": 0
  }
}
```

На сьогоднішній день більшість веб-розробників не використовують при роботі зі стилями чистий CSS, а використовують метамови. Сама популярна з них – SASS. В даному проеткі використовується її діалект – SCSS. Він бере можливості SASS, але зберігає фігурні дужки оригінального CSS (SASS відрізняється відсутністю фігурних дужок, в ньому вкладені елементи реалізовані за допомогою відступів).

Рис. 3.12 – Лого SASS

Для перетворення SCSS в CSS використовують модулі для складачів. В даному проекті для роботи з цією мовою використовуються наступні модулі:

- sass-loader – основний модуль;
- node-sass – для роботи на NodeJS.

Код, представлений у лістингу 3.4 компілюється у код у лістингу 3.5.

Лістинг 3.4 Код стилів на SASS

```
.board {  display: inline-flex;  height: 100%; min-
width: 100%; }
.board-underlay { position: fixed; top: 0;
bottom: 0; left: 0; right: 0; z-index: -
1; transition: background 0.3s; }
.green .board-underlay { background: $green; }
.blue .board-underlay {  background: $blue; }
```

```
.red .board-underlay { background: $red; }
.pink .board-underlay {  background: $pink; }
.lists-wrapper { display: inline-flex;  box-sizing: border-
box; height: 100%;  padding: 85px 5px 8px 5px; }
.lists {  display: inline-flex;  align-items: flex-
start;  height: 100%;  user-select: none; }
```

Лістинг 3.5 Код стилів компільований з SASS в CSS

```
.boards { display: inline-flex;  flex-wrap: wrap }
.board-link { box-sizing: border-
box;  width: 260px;  height: 140px;
margin: 5px;    padding: 10px;  border-radius: 6px;    font-
size: 16px;    font-
weight: 700;    transition: background .1s;    cursor: point
er;    display: inline-flex;    flex-
direction: column;    color: #fff;    overflow-wrap: break-
word;    text-decoration: none }
.board-link.green { background: #032 }
.board-link.blue { background: #06a }
.board-link.red { background: #831}
.board-link.pink {  background: #b06 }
.board-link:focus.green,
.board-link:hover.green { background: #021 }
.board-link:focus.blue,
.board-link:hover.blue { background: #059 }
.board-link:focus.pink,
.board-link:hover.pink { background: #a05 }
.board-link:focus.red,
.board-link:hover.red { background: #720 }
.board-link-title { padding-bottom: 5px }
.mini-board { display: flex; height: 100% }
.mini-list {
    display: inline-block;
    box-sizing: border-box;
    width: 24px;
    height: 100%;
    margin-right: 6px;
    border-radius: 3px;
    background: hsla(0, 0%, 100%, .25)
}
```

Перед переглядом в браузері, проект повинен бути запущений на сервері. Для розробки використовують локальні сервери, бо доступ до віддалених дуже довгий і часто потребує додаткових маніпуляцій.

Для тестування в даному проекті використовувався модуль nodemon, який постійно перезавантажує локальний сервер після змін в проекті.

Рис. 3.13 – Знак nodemon в npm

Особливості:

– автоматичний перезапуск програми.

– виявляє розширення файлу за замовчуванням для моніторингу.

– підтримка за замовчуванням для вузла та coffeescript, але легко запускати будь-який виконуваний файл (наприклад, python, make тощо).

– ігнорування конкретних файлів чи каталогів.

– може переглядати конкретні каталоги.

– працює з серверними програмами або одноразовими утилітами та REPL.

Фреймворки завантажуються як модулі. Єдина відмінність – вони мають власну cli зі своїми командами та типи файлів. Для React характерні файлу типу jsx.

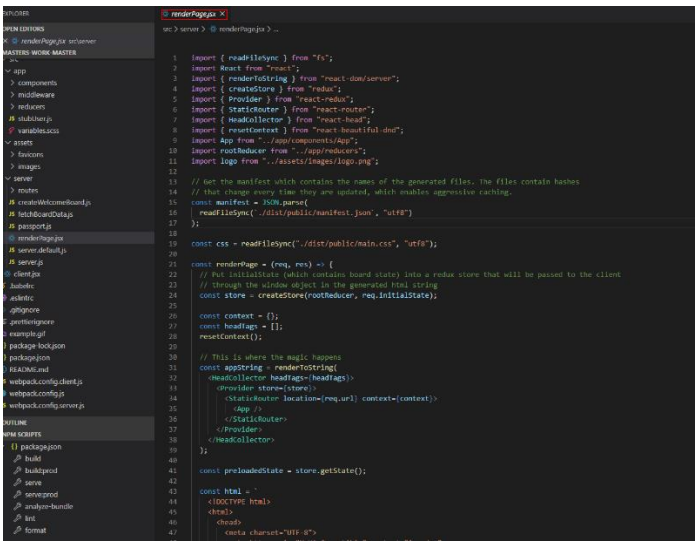

Рис. 3.14 – Файл типу jsx

React - JavaScript-бібліотека з відкритим вихідним кодом для розробки інтерфейсів, призначених для користувача.

У лістингу 3.6 наведено приклад використання React в HTML з JSX і JavaScript.

Лістинг 3.6 Приклад використання React в HTML

```
<div id = "myReactApp" > </ div>
<script  type = "text / babel">
  class  Greeter  extends  React . Component  {
    render ()  {   return  <h1> { this . props . greeting }
</h1>  }
  }
  ReactDOM . render ( <Greeter  greeting = "Hello World!"
/> ,  document . getElementById ( 'myReactApp' ));
</script>
```

Клас Greeter – це React компонент, який приймає властивість greeting. Метод ReactDOM.render відображає екземпляр класу (компонента) Greeter з властивістю greeting рівним 'Hello World' і вставляє відображати компонент в DOM-елемент з ідентифікатором myReactApp як вкладений елемент. В лістингу 3.7 відображено HTML елемент після рендеру.

Лістинг 3.7 HTML елемент після рендеру

```
<div  id = "myReactApp">
  <h1> Hello World! </ h1>
</ div>
```

Не тільки відображення HTML у браузері: React використовується не лише для рендерингу HTML у браузері. Наприклад, Facebook використовує динамічні графіки, які малюються на елементах <canvas>. Netflix і PayPal використовують ізоморфне рендеринг для створення ідентичного HTML як на сервері, так і на клієнті.

React Hooks: Хуки дозволяють використовувати стан та інші можливості React без написання класів. За допомогою спеціально призначених хуків можна включати логіку компонентів у повторно використовувані функції.

Програмування не стоїть на місці, а особливо веб-програмування. Управління станом в веб-додатках – це, по суті, розміщення всіх даних в одному місці, до якого можна однаково швидко достукатись з будь-якої частини коду. Також це забезпечує реактивність даних. Самою популярною бібліотекою для цього є Redux.

Redux – це відкрита бібліотека для JavaScript, яка призначена для управління станом додатка. Зазвичай вона використовується в поєднанні з React або Angular для розробки клієнтської частини. Вона надає набір інструментів, які спрощують передачу даних між різними частинами додатка через контекст. Redux був створений Данілом Абрамовим і Ендрю Кларком.

Рис. 3.15 – Лого Redux

Redux – бібліотека з простим API, передбачуване сховище стану додатків. Вона працює за тим же принципом, що і редьюсер, один з концептів функціонального програмування. Її творці надихалися функціональною мовою програмування Elm.

Весь стан програми зберігається в дереві об’єктів всередині одного магазину. Єдиний спосіб змінити стан дерева - це випромінювати дію, об’єкт, що описує те, що сталося. Приклад представлено в лістингу 3.8.

Лістинг 3.8 Код редюсеру

```
import { createStore } from 'redux'
function counter(state = 0, action) {
  switch (action.type) {
    case 'INCREMENT':
      return state + 1
    case 'DECREMENT':
      return state - 1
```

```
    default:
      return state  }}
let store = createStore(counter)
store.subscribe(() => console.log(store.getState()))
store.dispatch({ type: 'INCREMENT' }) // 1
store.dispatch({ type: 'INCREMENT' }) // 2
store.dispatch({ type: 'DECREMENT' }) // 1
```

Але можна додати новий функціонал. Для цього використовують інші модулі. В даному проекті це Focal.

Рис. 3.16 – Лого Focal

Бібліотека призначена для ефективного та зручного управляння станом додатку.

– представляє весь стан програми як непорушне та спостережуване єдине джерело істини;

– плавно вбудовує спостережувані елементи у компоновку компонентів react;

– використовує потужність rx.js (і спостережуваних даних загалом) для збагачення та комбінування частин стану програми, явно контролюючи потік даних;

– використовує лінзи, щоб розкласти стан програми на менші частини, щоб розробник зміг ізолювати компоненти інтерфейсу чистим способом та без особливих зусиль маніпулювати станом програми;

– дозволяє писати менше коду, який легше зрозуміти.

Приклад роботи бібліотеки зображено в лістингу 3.9.

Лістинг 3.9 Приклад роботи Focal

```
import { Atom } from '@grammarly/focal'
// create an Atom<number> with initial value of 0
const count = Atom.create(0)
// output the current value
console.log(count.get()) // => 0
count.set(5) // set 5 as the new value
console.log(count.get()) // => 5
count.modify(x => x + 1)
// modify the value: set a new value which is based on curre
nt value
console.log(count.get()) // => 6
```

Цей проект базується на MongoDB – системі управління базами даних з відкритим вихідним кодом, яка використовує документоорієнтований підхід і не вимагає опису схеми таблиць. MongoDB відноситься до категорії NoSQL і працює з JSON-подібними документами та схемою бази даних. Ця система реалізована на мові C++.

Рис. 3.17 – Лого MongoDB

СУБД має NoSQL модель, тому працює з JSON. JSON – простий формат обміну даними, зручний для читання і написання як людиною, так і комп'ютером.

3.2 Розробка ПЗ КПП

Основними файлами додатку АРМ КПП є:

- server.js, який відповідає за конфігурацію серверної частини додатку;
- client.jsx, який відповідає за клієнтську частину;
- App.scss, який відповідає за файл стилів.

Код для даних файлів представлений у додатках А-В.

3.2.1 Опис файлу server.js додатку APM керівника програмними проектами

Спочатку імпортуються необхідні npm пакети та конфігураційні файли:

```
import express from "express";
import { MongoClient } from "mongodb";
import passport from "passport";
import session from "express-session";
import connectMongo from "connect-mongo";
import compression from "compression";
import serveStatic from "express-static-gzip";
import helmet from "helmet";
import favicon from "serve-favicon";
import logger from "morgan";
import dotenv from "dotenv";
import renderPage from "./renderPage";
import configurePassport from "./passport";
import api from "./routes/api";
import auth from "./routes/auth";
import fetchBoardData from "./fetchBoardData";
```

– express – це фреймворк для веб-додатків Node.js, який використовується для налаштування та запуску додатку;

– MongoClient – сервіс для підключення клієнту до БД MongoDB;

– passport – сервіс для авторизації;

– session – сервіс для створення сесії;

– connectMongo – сервіс для підключення БД до додатку;

– compression – додаток для стиснення коду;

– serveStatic – додаток що дозволяє подавати файли заздалегідь стисненими;

– helmet – захищає додаток за допомогою різних HTTP заголовків;

– favicon – кешування іконки додатку;

– logger – для збереження інформації про HTTP запити;

– dotenv – завантажує змінні середовища з .env-файлу;

– renderPage – файл для рендеру клієнтської частини;

– configurePassport – конфігурація авторизації;

– api – список доступних запитів;

– fetchBoardData – допомагає отримати дані для роботи з дошками.

Далі завантажуються змінні середовища з .env-файлу: `dotenv.config();`

Ініціалізується фреймфорк express: `const app = express();`

Підключається зв'язок з БД: `const MongoStore = connectMongo(session);`

Створюється зашифроване посилання для зв'язку з БД:

```
const uri = `mongodb+srv://Andrey:${encodeURIComponent('f7v-h%wCVhj#nW[{')}@cluster0-pkgsd.mongodb.net/test?retryWrites=true&w=majority`;
```

Підключення бази даних:

```
MongoClient.connect(uri,
{ useNewUrlParser: true }).then(client => {
```

Якщо підключення не пройшло успішно, помилка буде оброблена та відображена у консолі:

```
.catch(console.error);
```

Якщо підключення пройшло успішно, зв'язуємо користувача з базою даних: `const db = client.db(process.env.MONGODB_NAME);`

Запускаємо конфігурацію авторизації: `configurePassport(db);`

Підключаємо імпортовані сервіси до express фреймворку:

```
 app.use(helmet());
  app.use(logger("tiny"));
  app.use(compression());
  app.use(favicon("dist/public/favicons/favicon.ico"));
  app.use(express.json());
  app.use(express.urlencoded({ extended: true }));
  app.use(
    "/static",
    serveStatic("dist/public", { enableBrotli: true, maxAge: "1y" })
  );
  app.use(
    session({
      store: new MongoStore({ db }),
```

```
        secret: process.env.SESSION_SECRET,
        resave: false,
        saveUninitialized: false
      })
    );
    app.use(passport.initialize());
    app.use(passport.session());
    app.use("/auth", auth);
    app.use("/api", api(db));
    app.use(fetchBoardData(db));
```

Отримуємо фронт сторінку:

```
app.get("*", renderPage);
```

Ініціалізуємо порт із змінних середовища або використовуємо стандартний: `const port = process.env.PORT || "1337";`

«Слухаємо» додаток за заданим портом:

```
app.listen(port, () => console.log(Server listening on port
${port}));
```

3.2.2 Опис файлу client.jsx додатку APM керівника програмними проектами

Імпортуються необхідні npm пакети та конфігураційні файли:

```
import React from "react";
import ReactDOM from "react-dom";
import { createStore, applyMiddleware } from "redux";
import { Provider } from "react-redux";
import { composeWithDevTools } from "redux-devtools-
extension";
import { BrowserRouter } from "react-router-dom";
import rootReducer from "./app/reducers";
import persistMiddleware from
"./app/middleware/persistMiddleware";
import App from "./app/components/App";
```

– React та ReactDOM – потрібні для рендеру React додатку;

– createStore, applyMiddleware та rootReducer – потрібні для створення та логування сховища даних;

– Provider – дозволяє передати дані зі сховища в React додаток;

– composeWithDevTools – додаток, за допомогою якого можна «дебажити» додаток в браузері;

– BrowserRouter – дає можливість React додатку переходити по сторінкам з різними URL;

– persistMiddleware – зберігає дані сховища додатку в локальне сховище браузеру;

– App – імпорт React компоненту.

Далі витягується початковий стан сховища:

```
const preloadedState = window.PRELOADED_STATE;
```

Видаляється початковий стан сховища з об’єкту window тому, що далі він не знадобиться:

```
delete window.PRELOADED_STATE;
```

Створюється та конфігурується сховище:

```
const store = createStore(
  rootReducer,
  preloadedState,
  composeWithDevTools(applyMiddleware(persistMiddleware))
);
```

Далі рендериться React додаток та вставляється в сторінку за допомогою html елементу з id "app":

```
ReactDOM.hydrate(
  <Provider store={store}>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </Provider>,
  document.getElementById("app")
);
```

3.2.3 Опис файлу App.scss додатку APM керівника програмними проектами

У файлі App.scss задаються базові стилі. Для html елементу задається висота та ширина:

```
html {
  height: 100%;
  width: 100%;
}
```

В ньому визначаємо стилі полоси прокрутки:

```
  &::-webkit-scrollbar {
    width: 10px;
    height: 10px;
    background: #ddd;
  }
  &::-webkit-scrollbar-thumb {
    border-radius: 3px;
    background: #aaa;
}
```

Далі тілу html документу задаємо висоту, ширину, зовнішній відступ та шрифт:

```
body {
  height: 100%;
  width: 100%;
  margin: 0;
  font-family: "Helvetica Neue", "Segoe UI", "Trebuchet MS",
Geneva, Tahoma,
    sans-serif;
  line-height: 1;
}
```

Для елементу з id «app» визначаємо тип відображення на сторінці, висоту та мінімальну ширину:

```
#app {
  display: inline-flex;
  height: 100%;
  min-width: 100%;
}
```

Визначаємо стандартні стилі для кнопок, посилань та строкового елементу span:

```
button,
span,
a {
  vertical-align: baseline;
  margin: 0;
  padding: 0;
```

```
  border: 0;
  font-size: 100%;
  font: inherit;
}
```

3.3 Розробка інструкції по впровадженню та експлуатації

Для роботи програми необхідно використовувати такі операційні системи як Windows або Ubuntu.

По перше треба склонувати проект КПП з гітхаб. Це можна зробити за використанням терміналу та команди: `git clone https://github.com/AndreyBorenko/Masters-work.git`.

Після успішного клонування проекту можна побачити файлову структуру додатку. Наступні кроки описані у файлі README у корні проекту (рис 3.18).

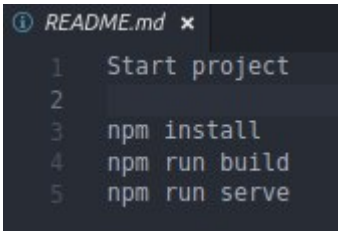

Рисунок 3.18 – Інструкція по запуску додатку

Команда npm install встановлює необхідні залежності проекту. Успішна робота цієї команди зображена на рисунку 3.19.

Команда npm run build збирає проект у папку dist. Успішна робота цієї команди зображена на рисунку 3.20.

Команда npm run serve запускає проект та відкриває додаток у новому вікні.

Стартовою сторінкою служить список проектів, який зображено на рисунку 3.21, тобто кожен користувач одразу переходить до роботи.

```
andrey@andrey-System-Product-Name:~/Projects/Masters-work$ npm install
info No lockfile found.
[1/4] Resolving packages...
warning react-beautiful-dnd > babel-runtime > core-js@2.6.11: core-js@<3 is no longer maintained and not recommended for usage
warning react-head > @babel/runtime > core-js@2.6.11: core-js@<3 is no longer maintained and not recommended for usage due to t
warning css-loader > cssnano > autoprefixer > browserslist@1.7.7: Browserslist 2 could fail on reading Browserslist >3.0 config
warning css-loader > cssnano > postcss-merge-rules > browserslist@1.7.7: Browserslist 2 could fail on reading Browserslist >3.0
warning css-loader > cssnano > postcss-merge-rules > caniuse-api > browserslist@1.7.7: Browserslist 2 could fail on reading Bro
warning eslint > file-entry-cache > flat-cache > circular-json@0.3.3: CircularJSON is in maintenance only, flatted is its succe
warning webpack-bundle-analyzer > bfj-node4@5.3.1: Switch to the `bfj` package for fixes and new features!
warning webpack-cli > jscodeshift > babel-register > core-js@2.6.11: core-js@<3 is no longer maintained and not recommended for
warning webpack-cli > jscodeshift > nomnom@1.8.1: Package no longer supported. Contact support@npmjs.com for more info.
warning webpack-cli > webpack-addons > jscodeshift > nomnom@1.8.1: Package no longer supported. Contact support@npmjs.com for m
warning webpack-cli > webpack-addons > jscodeshift > recast > core-js@2.6.11: core-js@<3 is no longer maintained and not recomm
warning webpack-cli > jscodeshift > babel-preset-es2015@6.24.1: 🙌  Thanks for using Babel: we recommend using babel-preset-env
warning webpack-cli > webpack-addons > jscodeshift > babel-preset-es2015@6.24.1: 🙌  Thanks for using Babel: we recommend using
[2/4] Fetching packages...
info fsevents@1.2.11: The platform "linux" is incompatible with this module.
info "fsevents@1.2.11" is an optional dependency and failed compatibility check. Excluding it from installation.
[3/4] Linking dependencies...
warning "eslint-plugin-jsx-a11y > axobject-query@2.1.1" has incorrect peer dependency "eslint@^5 || ^6".
warning "eslint-plugin-react > eslint-plugin-eslint-plugin@2.1.0" has incorrect peer dependency "eslint@>=5.0.0".
[4/4] Building fresh packages...
success Saved lockfile.
Done in 469.07s.
andrey@andrey-System-Product-Name:~/Projects/Masters-work$
```

Рисунок 3.19 – Успішне встановлення залежностей

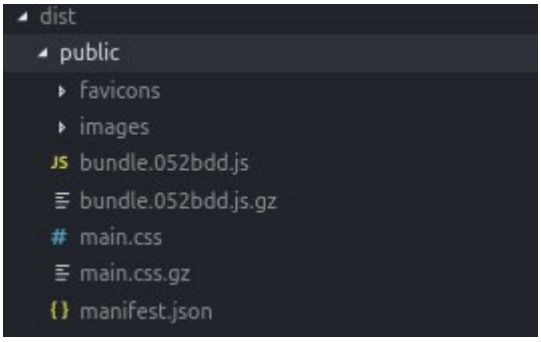

Рисунок 3.20 – Успішний збір проекту

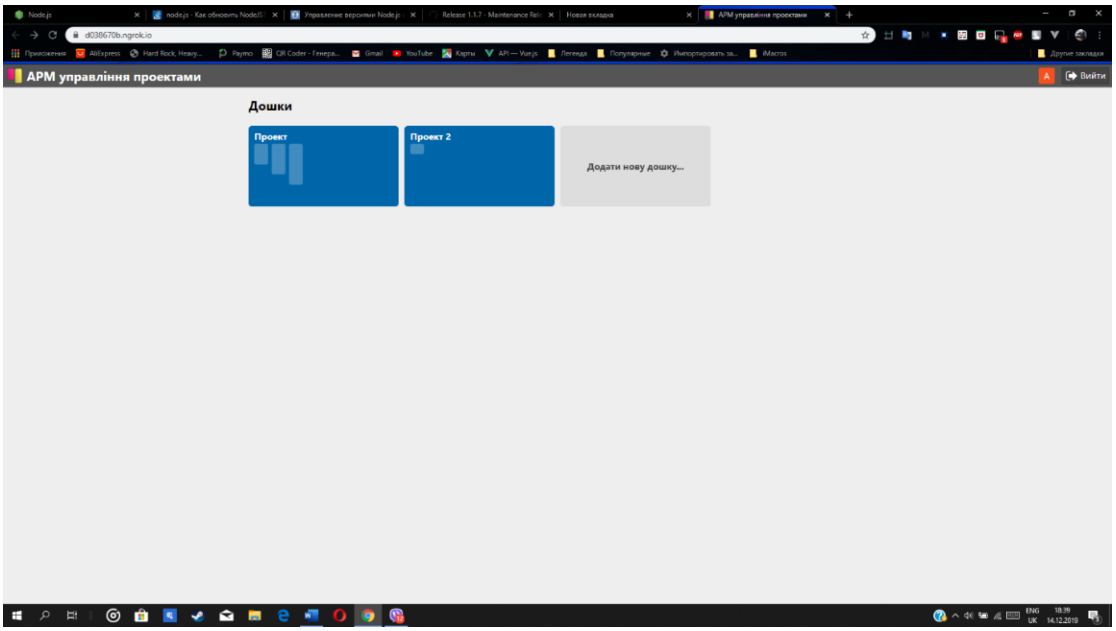

Рис. 3.21 – Головна/список проектів

Кожен проект поділений на певні задачі. Задачі сортуються по певним параметрам, на рисунку 3.22 показано, які саме задає менеджер.

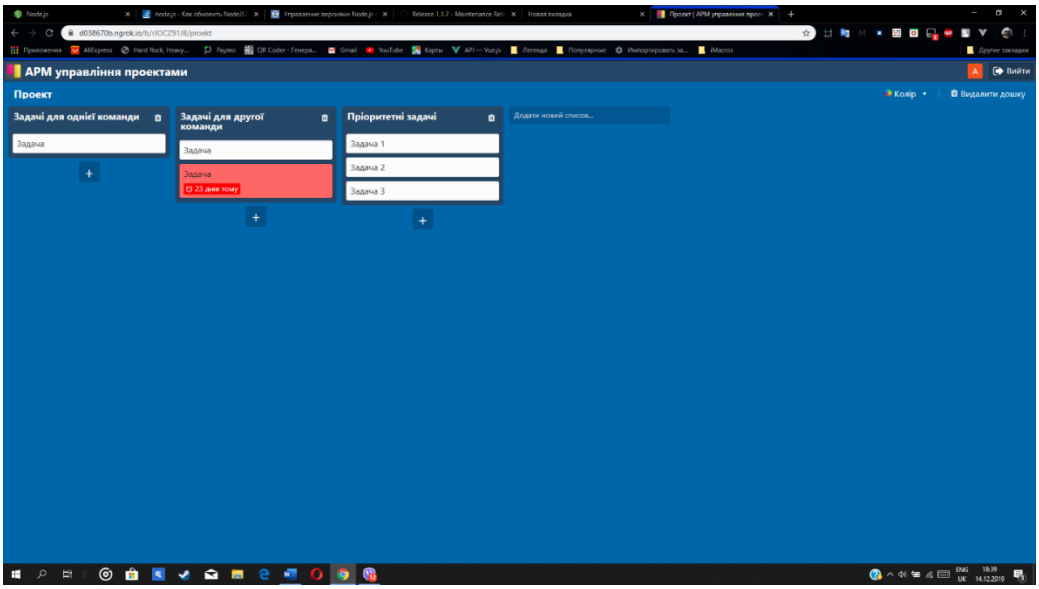

Рис. 3.22 – Задачі проекту

Можна додавати додаткові задачі до проекту, на рисунку 3.23, представлена ця дія.

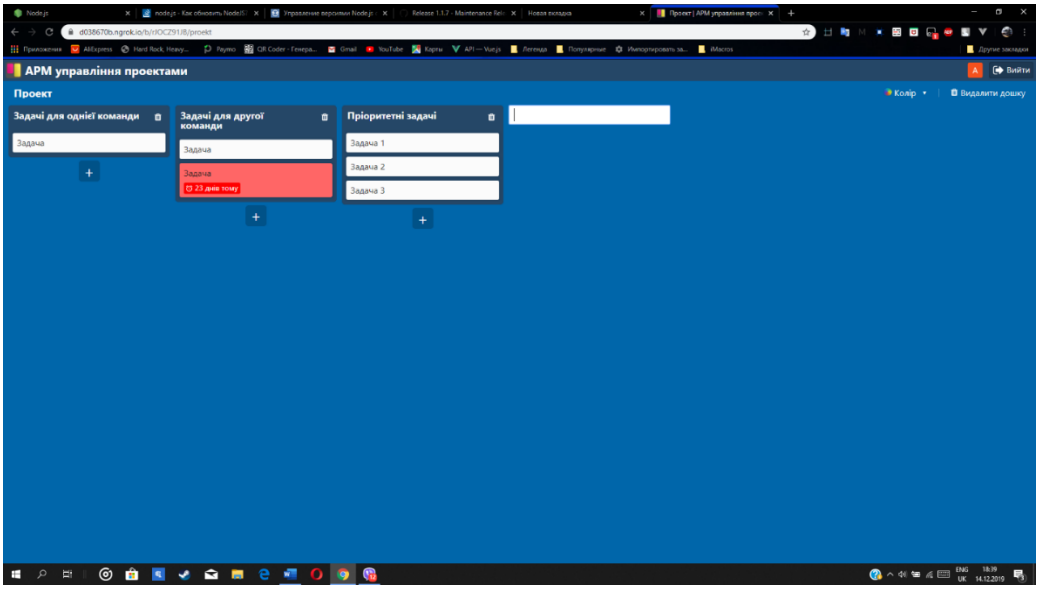

Рис. 3.23 – Додавання задачі до проекту

Задачі можна редагувати (рис. 3.24).

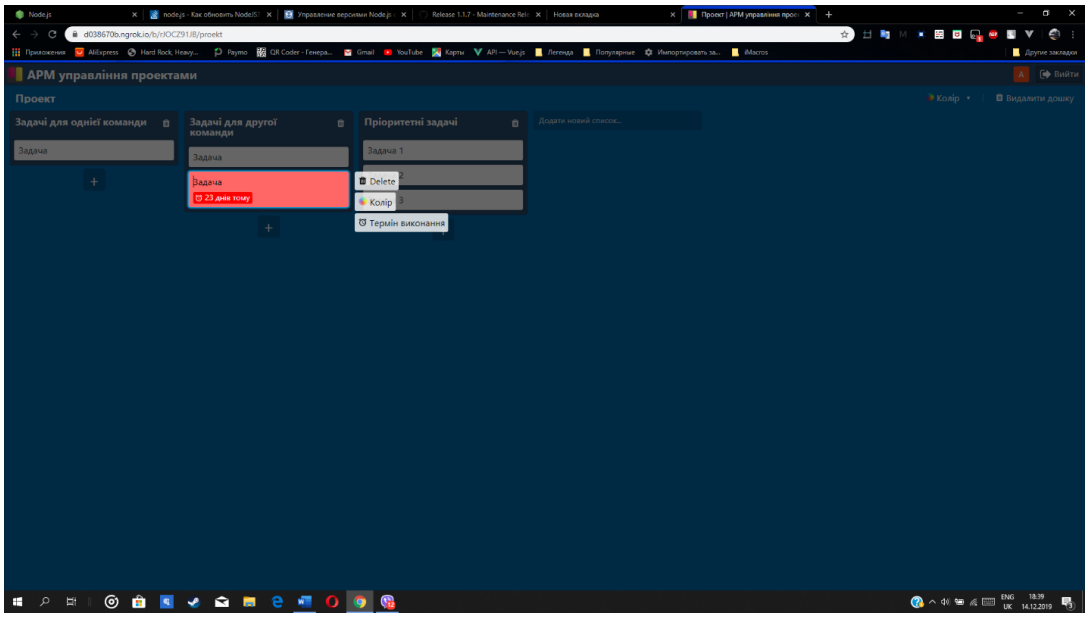

Рис. 3.24 – Параметри задачі

Робоче місце можна підлаштовувати під певного користувача, персонал, – змінювати фон WEB-сайту (рис. 3.25)

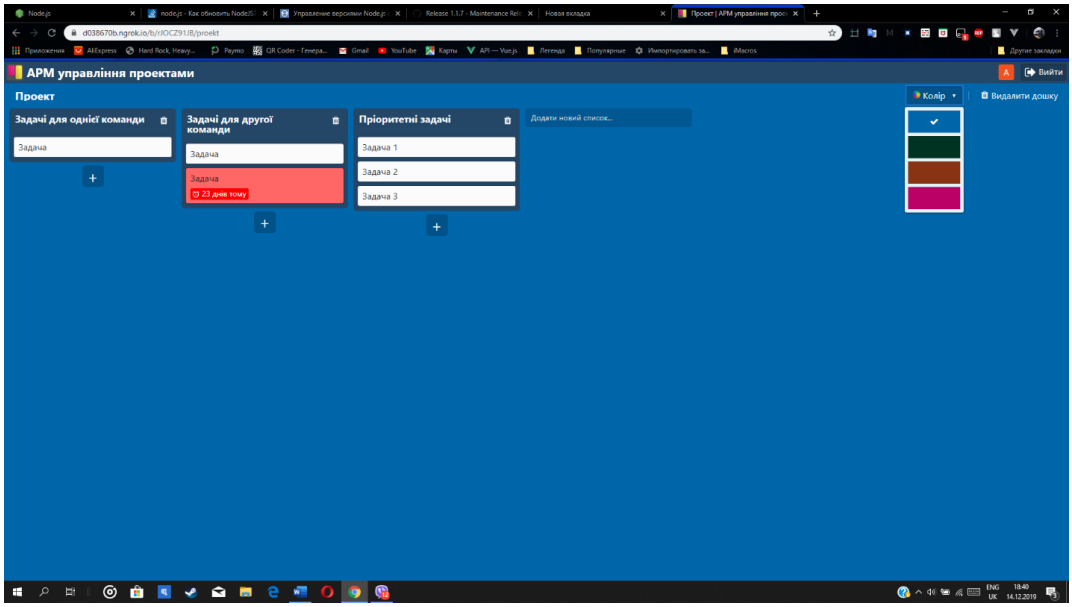

Рис. 3.25 – Вибір фону сайту

По завершенню або за інших обставин, задачі можна видаляти (рис. 3.26).

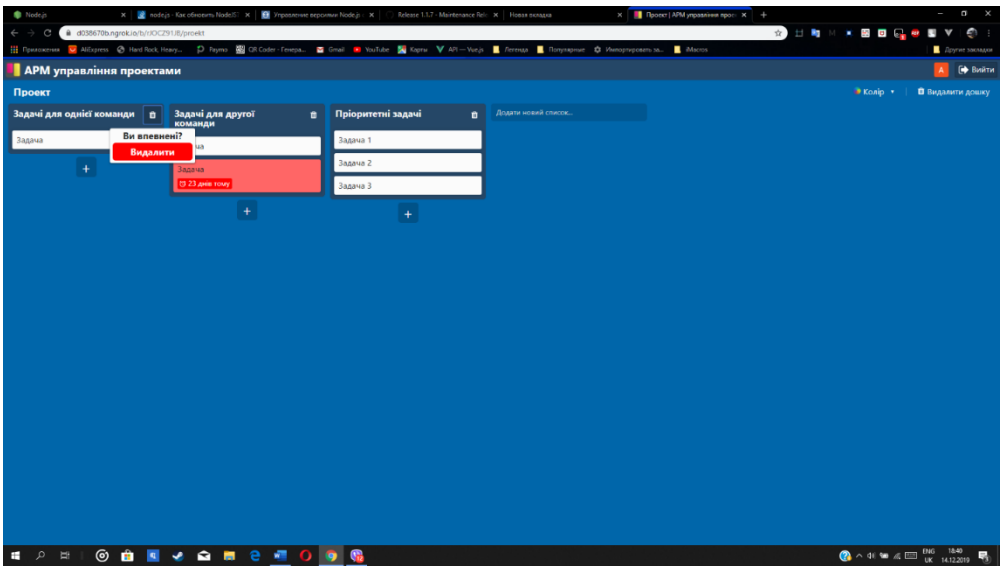

Рис. 3.26 – Видалення задачі

Можна створювати проекти (дошки) (рис. 3.27).

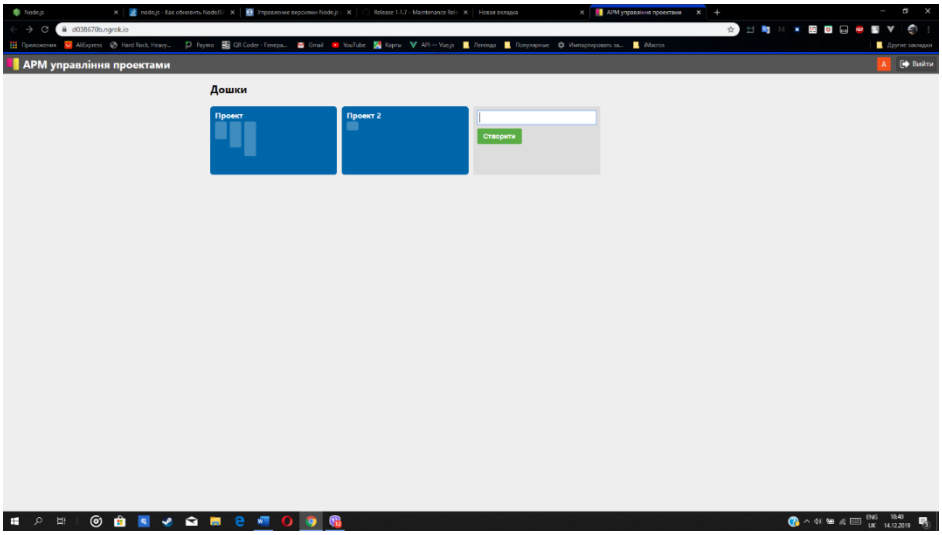

Рис. 3.27 – Додавання дошки

Таким чином, у цьому розділі були розглянуті технології та засоби створення веб додатку. Описана розробка основних файлів системи. Серед них були:

– server.js – який відповідає за конфігурацію серверу та з’єднанню з БД;

– client.jsx – який відповідає за конфігурацію клієнту та виводу інтерфейсу;

– App.scss – який відповідає за стилі додатку.

ВИСНОВКИ

Сутність проектного управління полягає в систематичному підході до планування, організації, виконання та контролю проектів з метою досягнення конкретних цілей. Основна мета проектного управління – забезпечити успішне виконання проекту в межах обмежень, таких як обсяг ресурсів, бюджет, терміни та якість.

Веб-додаток може бути потужним засобом управління проектами, що надає командам зручність, доступність та спільну платформу для спілкування та керування проектами.

У дипломній роботі виконано його основну мету – створення ARM менеджера проектів. Система виконує свої основні функції.

В роботі проведено аналіз та виконано наступні задачі:

– створено базу даних для проекту;

– написано код програми;

– винесені висновки та складено керівництво користувача;

– оформлена проектна документація, яка включає «Керівництво користувача».

Виконана робота має конкретний практичний характер використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Фергус О’Коннел (2005 р.). «Как успешно руководить проектами. Серебряная пуля».
2. https://habr.com/ru/company/hygger/blog/460985/
3. "React - бібліотека JavaScript для побудови інтерфейсів користувача" .
4. Крилл, Павло " React. Створюйте швидші, плавніші інтерфейси для веб-додатків, керованих даними" . InfoWorld .
5. Доусон, Кріс (25 липня 2014 р.). "Історія JavaScript і як це призвело до React на JS" . Новий стек .
6. Дере, Мохан (2018-02-19). "Як інтегрувати додаток create-react з усіма бібліотеками, які вам потрібні, щоб зробити чудовий додаток" . freeCodeCamp .
7. Самп, Йон (2018-01-13). " React маршрутизатор на перший маршрутизатор Redux" . Про Codecademy .
8. "Redux · Передбачуваний державний контейнер для JS Apps" . redux.js.org.
9. "Маршрутизатор React: Декларативна маршрутизація для React" . ReactRouterWebsite .
10. Обіцяйте HTTP-клієнт для браузера та node.js: axios / axios , axios, 2019-10-23
11. "Компоненти та реквізити" . React . Facebook .
12. "Відгуки та DOM" . Блог React.
13. "Чернетка: Специфікація JSX" . JSX . Facebook .
14. Кларк, Ендрю (26 вересня 2017 р.). "React v16.0§Нові типи повернення візуалізації: фрагменти та рядки" . Блог React.
15. Кларк, Ендрю (26 вересня 2017 р.). " React v16.0§підтримка спеціальних атрибутів DOM" . Блог React.
16. Фішер, Людовико (2017-09-06). React на реальні: Front-End Code, Untangled . Прагматична книжкова полиця. ISBN 9781680504484.

17. "Чому ми побудували React? - React Blog" .
18. "reactjs / react-future - майбутнє реагувати" . GitHub.
19. "facebook / react - проблеми із запитом на функції" . GitHub.
20. Уолк, Йордан. "FaxJS".
21. " " React v16.0 " . React.js. 2017-09-26 .
22. " " React v16.8 " . React.js. 2019-02-16 .
23. "React CHANGELOG.md" . GitHub .
24. Лю, Остін. "Переконлива причина не використовувати ReactJS" . Середній .
25. Кларк, Ендрю (26 вересня 2017 р.). " React v16.0§MIT ліцензовано" . Блог React.
26. Хунзакер, Натан (25 вересня 2017 р.). " React v15.6.2" . Блог React.
27. Кайл Бэнкер (2012 р.). " MongoDB в действии".
28. Система управління програмними проектами Trello [Електронний ресурс] - Режим доступу: https://www.trello.com.
29. Система управління програмними проектами Paymoapp [Електронний ресурс] - Режим доступу: https://www.paymoapp.com.
30.

ДОДАТОК А

Лістинг конфігураційного файлу для серверу – server.js

```
import express from "express";
import { MongoClient } from "mongodb";
import passport from "passport";
import session from "express-session";
import connectMongo from "connect-mongo";
import compression from "compression";
import serveStatic from "express-static-gzip";
import helmet from "helmet";
import favicon from "serve-favicon";
import logger from "morgan";
import dotenv from "dotenv";
import renderPage from "./renderPage";
import configurePassport from "./passport";
import api from "./routes/api";
import fetchBoardData from "./fetchBoardData";

 .env file
dotenv.config();

const app = express();

const MongoStore = connectMongo(session);

const uri = `mongodb+srv://Andrey:${encodeURIComponent('f7v-
h%wCVhj#nW[{')}@cluster0-
pkgsd.mongodb.net/test?retryWrites=true&w=majority`;

MongoClient.connect(uri, { useNewUrlParser: true }).then(client
=> {
  const db = client.db(process.env.MONGODB_NAME);

  configurePassport(db);

  app.use(helmet());
  app.use(logger("tiny"));
  app.use(compression());
  app.use(favicon("dist/public/favicons/favicon.ico"));
  app.use(express.json());
  app.use(express.urlencoded({ extended: true }));
  app.use(
    "/static",
    serveStatic("dist/public", { enableBrotli: true, maxAge:
"1y" })
  );
```

```
  // Persist session in mongoDB
  app.use(
    session({
      store: new MongoStore({ db }),
      secret: process.env.SESSION_SECRET,
      resave: false,
      saveUninitialized: false
    })
  );
  app.use(passport.initialize());
  app.use(passport.session());
  app.use("/api", api(db));
  app.use(fetchBoardData(db));
  app.get("*", renderPage);

  const port = process.env.PORT || "1337";
  /* eslint-disable no-console */
  app.listen(port, () => console.log(Server listening on port
${port}));
}).catch(console.error);
```

ДОДАТОК Б

Лістинг файлу для клієнту – client.jsx

```
import React from "react";
import ReactDOM from "react-dom";
import { createStore, applyMiddleware } from "redux";
import { Provider } from "react-redux";
import { composeWithDevTools } from "redux-devtools-extension";
import { BrowserRouter } from "react-router-dom";
import rootReducer from "./app/reducers";
import persistMiddleware from
"./app/middleware/persistMiddleware";
import App from "./app/components/App";

// Extract initial redux state received from the server
const preloadedState = window.PRELOADED_STATE;
delete window.PRELOADED_STATE;

const store = createStore(
  rootReducer,
  preloadedState,
  composeWithDevTools(applyMiddleware(persistMiddleware))
);

ReactDOM.hydrate(
  <Provider store={store}>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </Provider>,
  document.getElementById("app")
);
```

ДОДАТОК В

Лістинг файлу стилів – App.scss

```
html {
  height: 100%;
  width: 100%;
  &::-webkit-scrollbar {
    width: 10px;
    height: 10px;
    background: #ddd;
  }

  &::-webkit-scrollbar-thumb {
    border-radius: 3px;
    background: #aaa;
  }
}

body {
  height: 100%;
  width: 100%;
  margin: 0;
  font-family: "Helvetica Neue", "Segoe UI", "Trebuchet MS",
Geneva, Tahoma,
    sans-serif;
  line-height: 1;
}

#app {
  display: inline-flex;
  height: 100%;
  min-width: 100%;
}

button,
span,
a {
  vertical-align: baseline;
  margin: 0;
  padding: 0;
  border: 0;
  font-size: 100%;
  font: inherit;
}
```

ДОДАТОК Г

Список npm команд

```
> access Set access level on published packages
> adduser Add a registry user account
> audit Run a security audit
> bin Display npm bin folder
> bugs Bugs for a package in a web browser maybe
> build Build a package
> bundle REMOVED
> cache Manipulates packages cache
> ci Install a project with a clean slate
> completion Tab Completion for npm
> config Manage the npm configuration files
> dedupe Reduce duplication
> deprecate Deprecate a version of a package
> dist-tag Modify package distribution tags
> docs Docs for a package in a web browser maybe
> doctor Check your environments
> edit Edit an installed package
> explore Browse an installed package
> help-search Search npm help documentation
> help Get help on npm
> hook Manage registry hooks
> init create a package.json file
> install-ci-test Install a project with a clean slate and run
tests
> install-test Install package(s) and run tests
> install Install a package
> link Symlink a package folder
> logout Log out of the registry
> ls List installed packages
> npm javascript package manager
> org Manage orgs
> outdated Check for outdated packages
> owner Manage package owners
> pack Create a tarball from a package
> ping Ping npm registry
> prefix Display prefix
> profile Change settings on your registry profile
> prune Remove extraneous packages
> publish Publish a package
> rebuild Rebuild a package
> repo Open package repository page in the browser
> restart Restart a package
> root Display npm root
```

```
> run-script Run arbitrary package scripts
> search Search for packages
> shrinkwrap Lock down dependency versions for publication
> star Mark your favorite packages
> stars View packages marked as favorites
> start Start a package
> stop Stop a package
> team Manage organization teams and team memberships
> test Test a package
> token Manage your authentication tokens
> uninstall Remove a package
> unpublish Remove a package from the registry
> update Update a package
> version Bump a package version
> view View registry info
> whoami Display npm username
```