МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____Аліна САВЧЕНКО

"____"_____2021 р.

# ДИПЛОМНА РОБОТА
**(ПОЯСНЮВАЛЬНА ЗАПИСКА)**

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
**"МАГІСТРА"**
ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ "ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ"

**Тема: " Алгоритм проектування та розробки веб-додатку на базі JavaScript використовуючи бібліотеку React"**

**Виконавець:** Раков Олег Валентинович
**Керівник:** професор Воронін Альберт Миколайович

**Нормоконтролер:** _____ Ігор РАЙЧЕВ

**Київ - 2021**

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра Комп'ютерних інформаційних технологій

Галузь знань, спеціальність, освітньо-професійна програма: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології"_____.

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Аліна САВЧЕНКО
«____»_____2021р.

# ЗАВДАННЯ

**на виконання дипломної роботи студента**

Ракова Олега Валентиновича
(прізвище, ім'я, по батькові)

**1. Тема роботи:** «Алгоритм проектування та розробки веб-додатку на базі JavaScript використовуючи бібліотеку React» затверджена наказом ректора від 12.10.2021 за № 2228/ст.

**2. Термін виконання роботи:** з 12.10.2021 по 31.12.2021.

**3. Вихідні дані до роботи:** розробка веб-додатку для спрощення процесу управління дипломними проектами.

**4. Зміст пояснювальної записки:** вступ, постановка задачі реалізації модуля студент системи управління дипломними проектами та огляд існуючих програмних рішень поставленної задачі, засоби реалізації веб-додатку на базі JavaScript використовуючи бібліотеку React, опис реалізації веб-додатку на базі JavaScript використовуючи бібліотеку React, реалізація веб-додатка на базі JavaScript використовуючи бібліотеку React, висновок.

**5. Перелік обов'язкового ілюстративного матеріалу:** використання користувачем веб-додатку.

**6. Календарний план-графік**

| № п/п | Завдання | Термін виконання | Підпис керівника |
|---|---|---|---|
| 1. | Отримання завдання на дипломну роботу та побудова плану-графіку виконання робіт. | 12.10.2021 – 15.10.2021 | |
| 2. | Аналіз літератури та джерел за темою дипломного проекту. | 16.10.2021 – 19.10.2021 | |
| 3. | Розробка та затвердження плану дипломного проекту. | 20.10.2021 – 24.10.2021 | |
| 4. | Проведення консультації з науковим керівником щодо створення першого розділу. | 25.10.2021 – 31.10.2021 | |
| 5. | Розробка розділу 1. | 01.11.2021 – 07.11.2021 | |
| 6. | Розробка розділу 2. | 08.11.2021 – 17.11.2021 | |
| 7. | Розробка розділу 3. | 18.11.2021 – 01.12.2021 | |
| 8. | Розробка розділу 4. | 02.12.2021 – 11.12.2021 | |
| 9. | Висновки та оформлення пояснювальної записки дипломного проекту. | 12.12.2021 – 20.12.2021 | |

**7. Дата видачі завдання:** 12.10.2021р.

**Керівник дипломної роботи** _____ Альберт ВОРОНІН
(підпис керівника)

**Завдання прийняв до виконання** _____ Олег РАКОВ
(підпис випускника)

# РЕФЕРАТ

Пояснювальна записка до дипломної роботи “Алгоритм проектування та розробки веб-додатку на базі JavaScript використовуючи бібліотеку React” складається із вступу, чотирьох розділів, загальних висновків, списку використаних джерел і містить 77 сторінок тексту, 30 рисунків. Список використаних джерел містить 22 найменування.

**Метою** дипломної роботи є створення веб-додатку для спрощення процесу управління дипломними проектами.

**Предметом дослідження** є структура, категорії та розробка веб-додатків.

**Об’єктом дослідження** є веб-додаток на базі JavaScript використовуючи бібліотеку React.

**Ключові слова:** СКБД, API, БД, CRUD, SQL, React.

**ЗМІСТ**

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

СКБД — система керування базами даних;

API (англ. Application Programming Interface) — прикладний програмний інтерфейс;

БД — база даних;

CRUD (англ. create read update delete) — 4 базові функції управління даними «створення, зчитування, зміна і видалення»;

SQL (англ. Select query language) — декларативна мова програмування для взаємодії користувача з базами даних.

# ВСТУП

Управління проектом — це методологія організації, планування, керівництва, координації трудових, фінансових та матеріально-технічних ресурсів протягом проектного циклу, спрямована на ефективне досягнення його цілей шляхом застосування сучасних методів, техніки та технології управління для досягнення певних результатів, щодо складу та обсягу робіт, вартості, часу, якості та задоволення учасників проекту. Однією з актуальних проблем навчального процесу в університеті є управління випускними проектами. Наприклад, студенту необхідно знайти інформацію про розклад випускного проекту, доступні теми, список керівників.

Як наслідок, певні проблеми виникають у ситуації, коли учень не знає, до кого звернутися за інформацією, як знайти вчителя, чи робота, яка не виконується або виконується погано. У цифрову епоху, коли електронні пристрої майже завжди під рукою, контролювати і виправляти тези за допомогою спеціальних програм набагато легше. Особливе значення має Система управління дипломними проектами, метою якої є створення інструменту, здатного підвищити ефективність людських ресурсів.

Перший розділ містить постановку проблеми даної дипломної роботи та опис існуючих програмних рішень та переваг розроблених систем перед ними. Другий розділ описує інструменти та методи, які були використані для виконання дипломної роботи. Третій розділ містить опис програмної реалізації всієї системи. У четвертому розділі описано, як користувач працює з розробленим програмним продуктом. Далі є висновки, які були зроблені під час виконання.

# РОЗДІЛ 1.
# ПОСТАНОВКА ЗАДАЧІ РЕАЛІЗАЦІЇ МОДУЛЯ СТУДЕНТ СИСТЕМИ УПРАВЛІННЯ ДИПЛОМНИМИ ПРОЕКТАМИ ТА ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ ПОСТАВЛЕНОЇ ЗАДАЧІ

## 1.1. Постановка задачі реалізації модуля студент системи управління дипломними проектами

Метою роботи є розробка модуля “Студент” системи управління дипломними проектами, що надає можливість підвищити ефективність людських ресурсів і покращити комунікацію між викладачами та студентами.

Розроблена система повинна бути представленна як веб-застосунок.

Задачами, які мають виконуватись даним продуктом є:

— створення серверної частини, що має змогу взаємодіяти з базою даних;

— база даних, що задовольняє тему дипломної роботи;

— розробка веб-застосунку.

Система має забезпечувати наступні можливості:

— шукати та переглядати теми дипломних робіт;

— подавати і відміняти заявки на теми дипломних робіт;

— переглядати інформацію по своїй дипломній роботі після прийняття заявки викладачем;

— переглядати графік виконання дипломної роботи;

— переглядати навантаження на викладачів;

— переглядати інформацію про склад та напрямки лабораторії;

— переглядати та редагувати власний профіль;

— вхід в систему;

— відновлення паролю.

У цій роботі було проведено аналіз проблем управління дипломними проектами та знайдено рішення для полегшення роботи над дипломним проектом.

Дипломний проект – це робота, яка має об’єктивно оцінювати вміння. розв'язувати типові завдання, які в освітньо-кваліфікаційних характеристиках спрямовані на проектування та виконання робочих функцій.

Дипломні проекти (Технічне обслуговування) здійснюються на кінцевому етапі навчання студентів і передбачають: закрирення, систематизацію та отримання знань як практичних, таких і теоретичних з його спеціальністю та здібності, застосовують застосуванні економічні технічні індикації, т. п. розвиток навичок самостійної роботи та оволодіння методами дослідження та експериментів, пов’язаних із темою проекту.

## 1.2. Застосунок Trello

Застосунок Trello — це одна з найпростіших і зручних систем управління проектами. Вона універсальна, легка, гнучка, з відкритим API, а головне, безкоштовна. Її найсильніші сторони: можливість паралельно відстежувати статус різних завдань на одному екрані і зручна інтеграція з іншими популярними інструментами. Це максимально простий інструмент, який легко впровадити в робочий процес без довгої адаптації з боку персоналу.

Весь інтерфейс збудований на основі канбан-дощок. Для організації завдань використовується дошка з картками, які розподіляються за типами. Як правило, завдання розбиваються на заплановані, поточні і виконані. Якщо ви берете участь у великій кількості робочих проектів або якщо ви самозайняті, то ефективне управління проектами багато. Якщо ви не плануєте свій час і зусилля, то в кінцевому підсумку ви знехтуйте його на Facebook і в результаті відсутні терміни. Індустрія інструментів онлайн-управління проектами є дуже

конкурентоспроможною, але, як здається, вона перемагає над усіма іншими, це Trello (рисунок 1.1).

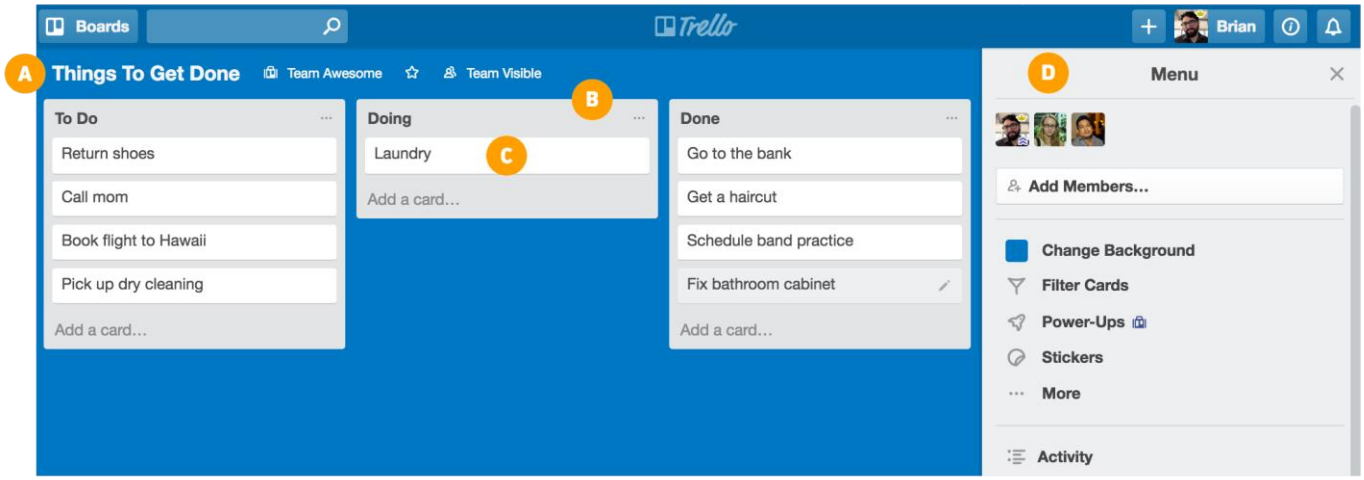


Рис. 1.1. Зразок інтерфейсу сайту Trello

Trello улюблений тими, хто одержимий управлінням своїм часом і проектами, і легко зрозуміти, чому. Trello не тільки дешевий і доступний, але й надзвичайно простий у використанні. Використовуючи дуже візуальний підхід до управління часом і проектами, ви можете переміщати карти навколо, щоб ви могли побачити, що потрібно зробити і коли.

Trello — це інструмент спільної роботи, який організовує ваші проекти в дошки. Одним поглядом, Трелло розповідає вам про те, на чому працює, хто працює над чим і де щось знаходиться в процесі.

Уявіть собі білу дошку, заповнену списками наліпок, кожна нота як завдання для вас і вашої команди. Тепер уявіть, що кожна з цих наліпок містить фотографії, вкладення з інших джерел даних, таких як BitBucket або Salesforce, документи, і місце для коментарів і співпраці з вашими товаришами по команді. А тепер уявіть, що ви можете взяти цю дошку в будь-якому місці вашого смартфона і отримати доступ до неї з будь-якого комп’ютера через Інтернет.

З мінусів я можу виділити те, що:

— функціоналу Trello недостатньо для дійсно великих компаній;

— на маленьких екранах Trello стає не такою зручною; автоматизацію, повторення і швидке додавання завдань важкувато реалізувати.

Незважаючи на деякі недоліки, Trello нарощує популярність і активно впроваджується компаніями в самих різних сферах. Це прекрасне онлайн-рішення для швидкої і легкої організації роботи.

Стан проектів в поточний момент часу і можливість запускати кілька проектів одночасно є головною перевагою Trello. Ця система надає можливість контролювати виконання проектів в будь-який момент часу і бачити зміни в режимі реального часу. Ця система підходить для домашніх проектів, просто наборів ідей і стартапів, але не для великих проектів.

## 1.3. Застосунок Atlassian JIRA

Застосунок JIRA — це система, яка підходить для стеження за вадами і управління проектом в компанії будь-якого розміру. Це інструмент для всіх співробітників в команді і керівників проектів.

JIRA допомагає команді обмінюватися інформацією і легко залучати різних співробітників в проекти і завдання, відстежувати і фіксувати помилки користувачів в роботі з програмними продуктами, забезпечувати дотримання роботи точно в строк і в рамках регламенту робочого процесу, перевіряти і планувати ефективність працівників і призначати певні завдання, працювати разом з колегами за допомогою інструментів спільного редагування файлів, а також відстежувати прогрес і оновлення кожного завдання команди.

Динамічні інструменти системи для управління проектами JIRA дають можливість керівникам виявити перешкоди, які не дають команді працювати ефективніше, приймати цілеспрямовані дії по їх усуненню і визначати області поліпшення робочого процесу. Необхідно багато часу і зусиль, щоб з’ясувати, як ефективно використовувати цей інтерфейс.

Основні можливості JIRA:

-функціоналу Trello недостатньо для дійсно великих компаній;

-перегляд процесу роботи розробників над проектом, які виникають помилки в його -використанні клієнтами, швидке їх усунення;

-документи, пошта, спілкування знаходяться в одному місці, рішення кожного питання відбувається швидко і легко;

-планування робочого процесу;

-постановка завдань, пріоритетів, допомога команді у виділенні важливого в роботі і відстеження виконання завдань;

-обмін інформацією по проекту, спільне вирішення питань і звернення за допомогою до колег;

-новини, пошта, перегляд роботи в режимі реального часу в одному місці;

-інтеграція з різними розробкамиі доповненнями Altassiani іншими розробниками.

Завдяки можливості настроювати JIRA її можна застосовувати і для задач поза IT, зокрема для управління HR, для ризик-менеджменту і управління вимогами.

Для виклику віддалених процедур використовується REST, раніше також була підтримка SOAP і XML-RPC, але від них відмовились у версії 7.0. В JIRA є підтримка англійської, французької, японської, іспанської і німецької мов. За допомогою додатків також можна добавити китайську, чеську, данську, італійську, норвезьку, польську, португальську, російську і словацьку мови.

JIRA відмінно підходить для швидких команд розробників і користувачів, які люблять технічне, складне програмне забезпечення.

Питання JIRA починаються з застарілого інтерфейсу. Програма має багато неефективності і часто виявляється надмірно складним. Команди управління проектами, які спробували параметри налаштування, вважають цей процес громіздким. Ті, хто шукає спосіб підвищити продуктивність, знайдуть дизайн дизайну JIRA користувачем невиправдано складним.

Зрештою, ніхто не буде сперечатися з тим, що JIRA була побудована на увазі команд розвитку. Організації, які не покладаються на розробників, знайдуть JIRA неефективною.

JIRA - це інструмент для відстеження випуску та проекту Atlassian. Простіше кажучи: JIRA дозволяє відстежувати будь-яку одиницю роботи (будь то проблема,

помилка, історія, завдання проекту тощо) через заздалегідь визначений робочий процес (рисунок 1.2).

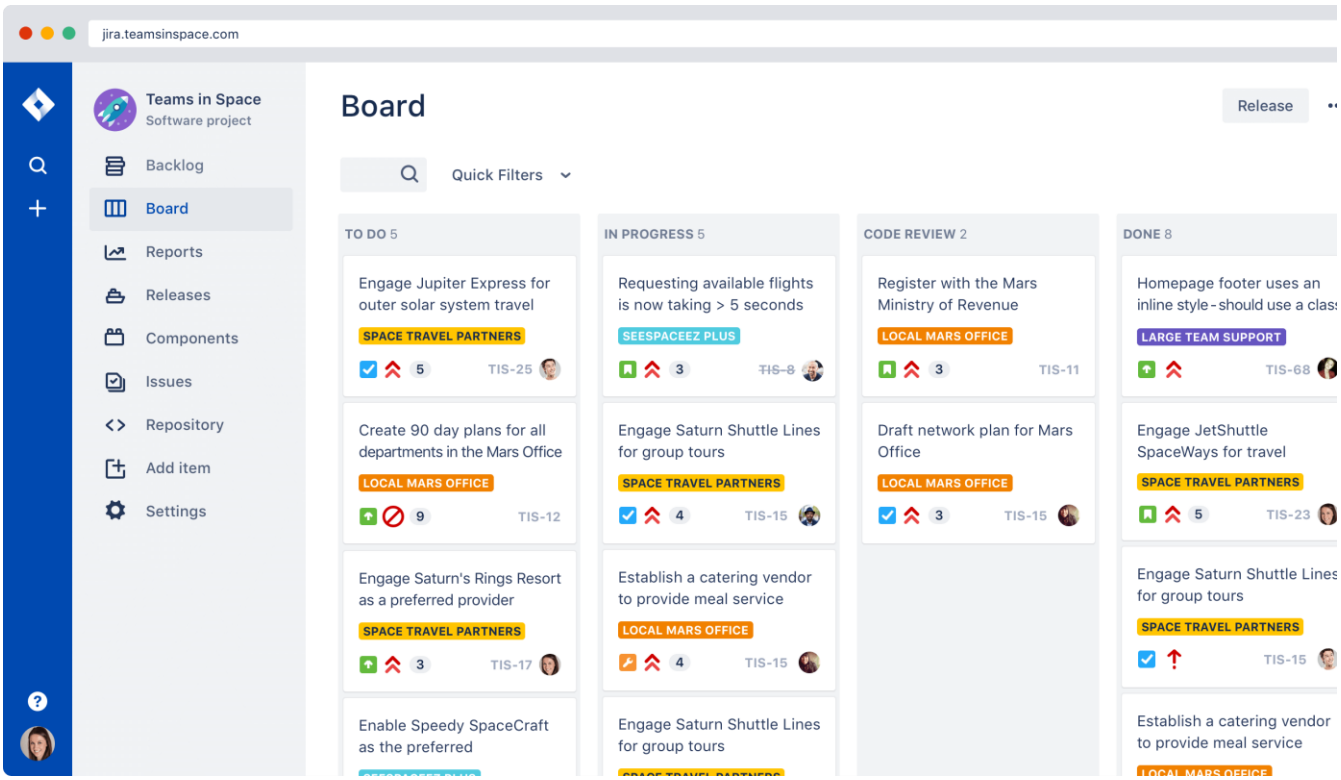

Рис. 1.2. Зразок інтерфейсу сайту Jira

Обидва пристрої (те, що ви називаєте робочим елементом) і робочий процес (кроки, які виконується елементом з відкритих до закритих), можуть бути налаштовані спеціально для конкретних вимог вашої команди, будь то прості або складніші.

Крім того, JIRA дійсно добре відстежує (через детальні, спеціальні звіти та інформаційні панелі), де всі ваші проекти / питання лежать на рівні команди, компанії або особистості - наприклад, які питання стосуються мене , створених за останні 7 днів?

Деякі поширені випадки використання включають розробку програмного забезпечення, реалізацію функцій, відстеження помилок, гнучке управління проектами (з JIRA Agile) та відстеження квитка на службу (з JIRA Service Desk).

Jira має високу конфігурацію та гнучкість, що дозволяє використовувати їх у різних середовищах та процесах. Робочі процеси Jira, типи випусків і екрани

дозволяють пристосовувати практично будь-який сценарій і легко можуть бути змінені за допомогою графічного інтерфейсу адміністратора. Atlassian надає відмінні ресурси онлайн-підтримки та особисто групи користувачів Atlassian, щоб допомогти спільноті.

# РОЗДІЛ 2.
# ЗАСОБИ РЕАЛІЗАЦІЇ ВЕБ-ДОДАТКА НА БАЗІ JAVASCRIPT ВИКОРИСТОВУЮЧИ БІБЛІОТЕКУ REACT

Ця система була написана на Javascript в операційній системі windows. Середовищем розробки були WebStorm, бібліотека React.js і фреймворк Express.js.

Інтерфейс користувача реалізовано за допомогою бібліотеки React.js в JavaScript, яка стилізована за допомогою бібліотеки styled-components.

Вся частина була реалізована за допомогою фреймворку Express.js Мова JavaScript.

Дані про дипломні роботи зберігаються в базі даних MySQL.

## 2.1. Вибір архітектури системи

Розглянута система повинна мати архітектуру клієнт-сервер. Це гранична або обчислювальна архітектура, в середині якої завдання розподіляються між постачальником послуг (функцією), який називається сервером, і клієнтами, які називаються клієнтами. Загалом клієнт і сервер є програмним забезпеченням. Ці програми зазвичай розташовані на різних комп’ютерах і взаємодіють один з одним через комп’ютерну мережу за допомогою мережевих протоколів, але вони також можуть бути розташовані на різних комп’ютерах.

Архітектура клієнт-сервер — це обчислювальна модель, в якій сервер розміщує, доставляє та керує більшістю ресурсів і послуг, які споживає клієнт. Цей тип архітектури має один або кілька клієнтських комп'ютерів, підключених до центральної мережі через мережу або підключених до Інтернету.

Архітектура клієнт-сервер також відома як модель мережевих обчислень або мережа клієнт-сервер, оскільки всі запити та послуги доставляються по мережі.

Архітектура клієнт-сервер — це архітектура виробників і споживачів, де сервер виступає як виробник, а клієнт — як споживач (рис. 2.1). Сервер розгортає та надає висококласні послуги клієнтам на вимогу. Ці послуги можуть включати доступ, зберігання, спільний доступ до файлів, доступ до принтера або прямий доступ до необробленої обчислювальної потужності сервера.

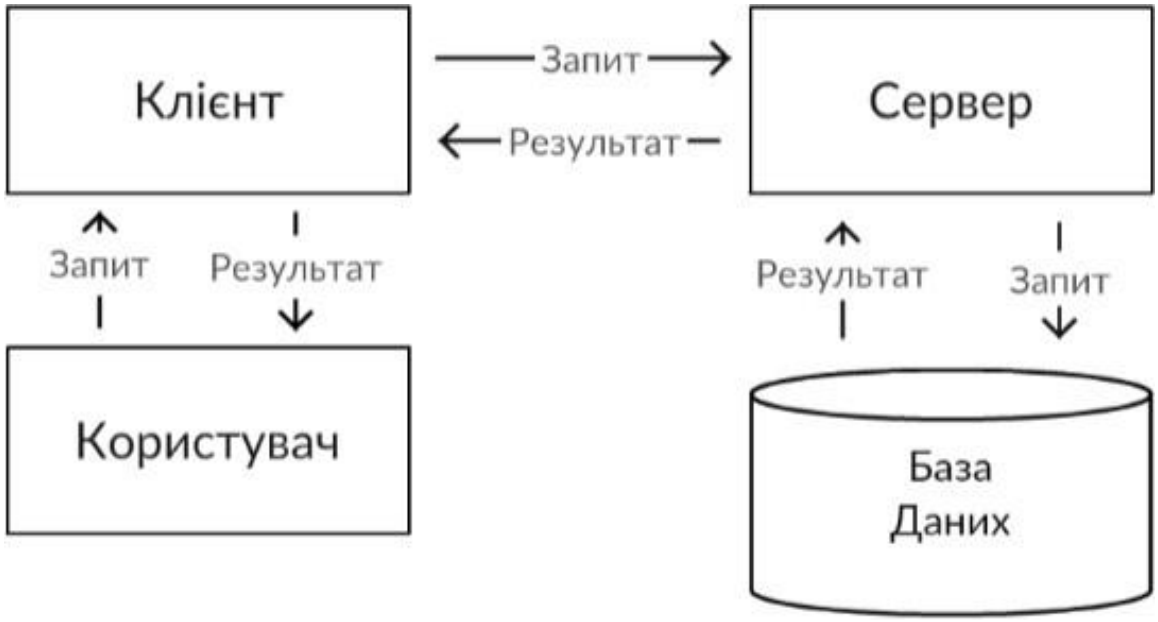


Рис. 2.1. Схематичне зображення клієнт – серверної архітектури

Архітектура клієнт-сервер працює, коли клієнтський комп’ютер надсилає запит процес або процес на сервері через мережеве з’єднання, який потім обробляється та доставляється клієнту. Кілька комп’ютерів можуть одночасно керувати кількома клієнтами, тоді як один клієнт може бути підключений до кількох серверів одночасно, кожен з яких надає різний набір послуг. У найпростішому вигляді Інтернет також заснований на архітектурі клієнт-сервер, де веб-сервери обслуговують багато одночасних користувачів з даними веб-сайту.

Посилення впливу моделі клієнт-сервер на вдосконалення онлайн-індустрії підвищило вимоги до програм клієнт-сервер. Додатки клієнтів відіграють значну

роль у спілкуванні з бізнес-організаціями в Інтернеті, які поширюються через Інтернет. Тут важливою стає важливість архітектури клієнт-архітектор.

Архітектура клієнт-сервер – це поширена система архітектури, де завантажується клієнт-сервер. Архітектура клієнт-сервер - це централізована система ресурсів, де сервер зберігає всі ресурси. Сервер отримує численні виступи на своєму сайті для спільного використання своїм клієнтам за запитом. Клієнт і сервер можуть бути в одній або декілька мереж. Ланцюжок глибоко стабільний і масштабований, щоб давати клієнтам відповіді. Ця архітектура є сервіс-орієнтованою, а це означає, що обслуговування клієнтів не буде передано. Архітектура клієнт-сервер підпорядковує мережевий трафік, відповідаючи на запити клієнта, а не на повну передачу файлів. Кошик відновлює файловий сервер за допомогою сервера бази даних.

Клієнтські комп’ютери спілкуються, щоб дозволити користувачеві комп'ютера, щоб запитати послуги сервера та представити результати, про які повідомляє сервер. Мережі чекають запитів від клієнтів і повертають їх. Сервер, як правило, надає клієнтам стандартизований простий інтерфейс, щоб уникнути плутанини апаратного і програмного забезпечення. Клієнти розташовуються на робочих місцях або на особистих автомобілях, при цьому сервери будуть розташовуватися десь потужне в мережі. Ця архітектура в основному корисна, коли клієнти мають виконувати конкретні завдання. Багато клієнтів можуть одночасно надсилати інформацію про сервер, а також клієнтський комп'ютер може виконувати інші завдання, наприклад, надсилати електронні листи.

## 2.2. Пакетний менеджер npm

Npm Package Manager — це менеджер пакетів для Node.js із сотнями тисяч пакетів. Незважаючи на те, що це створює деякі ваші структури/організації, це не є основною метою. Основна мета, як ви сказали, - це автоматизоване управління залежностями і пакетами. Це означає, що ви можете вказати всі залежності вашого проекту у файлі package.json, тоді щоразу, коли ви (або хтось інший) повинні почати роботу над проектом, вони можуть просто почати встановити та негайно встановити всі встановлені залежності. Крім того, ви також можете вказати, від яких версій залежить ваш проект, щоб запобігти розвитку вашого проекту.

Це, звичайно, можна вручну завантажити в бібліотеки, скопіювати правильні каталоги та використовуйте їх наступним чином. Однак у міру зростання вашого проекту (і списку залежностей) він швидко стає трудомістким і виснажливим. Це також ускладнює співпрацю та обмін вашим проектом.

Сподіваюся, це стане ясніше, яка мета npm. Як користувач Javascript (як на стороні клієнта, так і на стороні сервера), npm є незамінним інструментом у моєму робочому процесі.

## 2.3. Середовище розробки WebStorm

Середовище розробки WebStorm (Малюнок 2.2) — це інтегроване середовище розробки JavaScript, HTML і CSS від JetBrains, розроблене на основі платформи IntelliJ IDEA як високоякісна IDE.

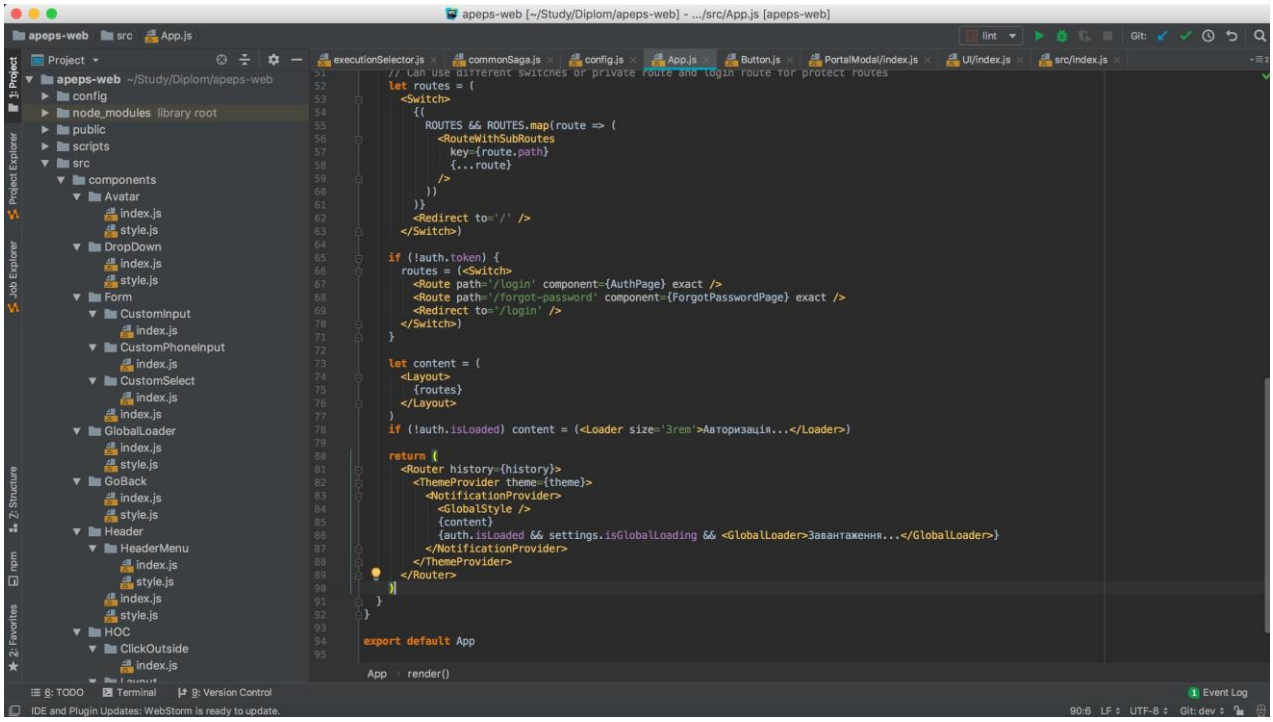

Рис. 2.2. Зразок інтерфейсу сайту середовища розробки WebStorm

WebStorm забезпечує інтуїтивно зрозумілу підтримку кодування для JavaScript і скомпільованих мов JavaScript, Node.js, HTML і CSS. Роздаткові матеріали можуть насолоджуватися доповненням коду, розширеними функціями навігації, розпізнаванням помилок «на льоту» та факторами для всіх цих мов. Програма надає велику допомогу для кодування Angular, React, Vue.js і Meteor. WebStorm також підтримує React Native, PhoneGap, Cordova і Ionic для підтримки мобільних пристроїв і включає Node.js для північної частини.

Оскільки всі ці функції доступні лише в одній системі, споживачам не потрібно купувати певні програми.

Інтуїтивно зрозумілий педагог WebStorm може ретельно оцінити проект отримати найкращі результати заповнення коду для всіх підтримуваних мов. Сотні інтегрованих розшифровок визначають будь-які можливі запитання щодо типів користувачів, а потім пропонують варіанти швидкого вирішення. Це гарантує якість проекту.

WebStorm має функцію навігації та пошуку, яка допомагає користувачам використовувати код ефективніше та швидше. Ви можете перейти до визначення

методу, функції або змінної. Це особливо важливо для помічників, які працюють над великими проектами.

Середовище розробки WebStorm може похвалитися вбудованими функціями для налаштування, тестування та відстеження ваших програм на стороні клієнта та Node.js. Завдання набагато простіше, так як вони вимагають лише мінімальної конфігурації. Копіювальні пристрої можуть вказувати точки зупинки, код і оцінювати вирази, не виходячи з IDE. WebStorm також має вбудований інструмент, який допомагає відстежувати коди JavaScript. Файли пов'язуються з викликами функцій і визначаються потенційні проблемні області для підвищення ефективності.

Інтерфейс усунення несправностей WebStorm легко інтегрується з інструменти командної панелі для веб-навчання. Це забезпечує оптимізований досвід, який підвищує продуктивність без використання командної підкови. Інструменти для збивання відображаються в простому уніфікованому інтерфейсі для виконання завдань Grunt, Gulp або npm.

На додаток до сотень власних версій WebStorm, технологія може запускати ESLint, JSCS, TSLint, Stylelint, JSHint або JSLint через ваш код. WebStorm має функцію локальної історії, яка може переписати всі дії та повернутися до попередніх версій.

Розробники можуть працювати над більшою кількістю програм, використовуючи вбудовані шаблони WebStorm. На додаток до популярних проектів, таких як Express або Web Starter Kit, користувачі можуть отримати доступ до більшої кількості генераторів проектів через інтеграцію Yeoman WebStorm.

Середовище розробки WebStorm легко налаштовується, оскільки його можна налаштувати для будь-якого стилю ручного кодування. Користувачі можуть налаштовувати мітки, шрифти та візуальні теми на панелі інструментів і макеті редактора.

Основними перевагами цієї інтегрованої системи розробки є: факторинг, навігація по коду, автозаповнення, аналіз коду на льоту, миттєва зміна системи через вплив системи.

Завдяки вбудованим інструментам тестування інтегрованої системи розробки WebStorm може виконувати тестування з найменшими зусиллями та часом. Програмісти можуть запускати та переглядати результати тестування безпосередньо в середовищі розробки, переписувати код за допомогою програм Jest і тестувати React.js за допомогою Enzyme.

### 2.4. Мова програмування JavaScript

JavaScript є динамічною мовою комп'ютерного програмування. Він легкий і найчастіше використовується як частина веб-сторінок, реалізація якого дозволяє клієнтському сценарію взаємодіяти з користувачем і створювати динамічні сторінки. Це інтегрована мова програмування з об'єктно-орієнтованими можливостями.

Спочатку JavaScript був відомий як LiveScript, але Netscape змінив свою назву на JavaScript, можливо, через збудження Java. JavaScript вперше з’явився в Netscape 2.0 у 1995 році під назвою LiveScript. Ядро мови загального призначення вбудовано в Netscape, Internet Explorer та інші веб-браузери.Специфікація ECMA-262 визначила стандартну версію основної мови JavaScript:

— функціоналу Trello недостатньо для дійсно великих компаній;
— JavaScript є легкою, інтерпретованою мовою програмування;
— призначений для створення мережевих орієнтованих додатків;
— доповнює та інтегрується з Java;
— доповнює і інтегрується з HTML;
— відкрита і крос-платформна.

### 2.4.1. JavaScript на стороні клієнта

На стороні клієнта JavaScript є найпоширенішою формою мови. Сценарій повинен бути включений в HTML-документ або посилання на нього, щоб код інтерпретував браузер.

Це означає, що веб-сторінка не обов'язково має бути статичним HTML, але може містити програми, які взаємодіють з користувачем, керують браузером і динамічно створюють вміст HTML.

Механізм на стороні клієнта JavaScript надає багато переваг перед традиційними сценаріями на стороні сервера CGI. Наприклад, ви можете використовувати JavaScript для перевірки того, що користувач вводить дійсну адресу електронної пошти в поле форми.

Код JavaScript виконується, коли користувач надсилає форму, і лише якщо всі записи дійсні, вони будуть відправлені на веб-сервер.

JavaScript можна використовувати для запису ініційованих користувачем подій, таких як натискання кнопок, навігація за посиланнями та інші дії, які явно чи неявно ініціюються користувачем.

### 2.4.2. Переваги JavaScript

Переваги використання JavaScript:

— менша взаємодія з сервером — ви можете перевірити вхід користувача перед відправкою сторінки на сервер, що економить навантаження на сервер;

— зворотний зв'язок з користувачами, яким не потрібно чекати, поки сторінка перезавантажиться, щоб побачити, що вони мають змогу щось ввести;

— підвищена інтерактивність — ви можете створювати інтерфейси, які реагують, коли користувач зависає над ними за допомогою миші або активує їх за допомогою клавіатури;

— більш чудові інтерфейси — ви можете використовувати JavaScript, щоб включити такі елементи, як компоненти перетягування і повзунки, щоб надати відвідувачам свого сайту багатий інтерфейс.

### 2.4.3. Обмеження JavaScript

Ми не можемо розглядати JavaScript як повноцінну мову програмування.

Вона не має таких важливих функцій:

— на стороні клієнта JavaScript не дозволяє читати або писати файли. Це зберігається з міркувань безпеки;

— JavaScript не можна використовувати для мережевих програм, оскільки такої підтримки немає;

— у JavaScript відсутні багатопоточні або багатопроцесорні можливості;

— знову ж таки, JavaScript є легкою, інтерпретованою мовою програмування, яка дозволяє створювати інтерактивність у статичних HTML-сторінках.

## 2.5. Бібліотека реалізації веб-інтерфейсу React.js

Бібліотека React.js — це бібліотека JavaScript з відкритим кодом, яка використовується для створення інтерфейсів користувача спеціально для однорівневих додатків. Він використовується для тестування презентації для веб-додатків і мобільних додатків. React також дозволяє нам створювати повторно використані компоненти інтерфейсу копіювання. React вперше створив Джордан Волке, інженер-програміст, який працює у Facebook. React вперше був опублікований у стрічці новин Facebook у 2011 році та на Instagram.com у 2012 році.

React дозволяє користувачам створювати великі веб-додатки, які можуть змінювати дані без перезавантаження сторінки. Основна мета React — бути швидким, масштабованим і простим. Працює лише на інтерфейсах, призначених для

користувача в додатку. Це відповідає представленню в шаблоні MVC. Його можна використовувати з комбінацією інших бібліотек JavaScript або середовищ, таких як Angular JS у MVC.

### 2.5.1. Чому React.js?

Тепер головне питання, яке постає перед нами, — чому слід використовувати ReactJS. Існує багато платформ з відкритим кодом для полегшення розробки веб-додатків, таких як Angular. Давайте швидко розглянемо переваги React перед іншими конкуруючими технологіями чи структурами. Оскільки інтерфейс змінюється щодня, важко приділяти час вивченню нових фреймів, особливо коли ці кадри можуть згодом зайти в тупик. Отже, якщо ви шукаєте наступний найкращий фреймворк, але відчуваєте, що застрягли в джунглях, я пропоную вам спробувати React.

Компонентно-орієнтований підхід, можливість легко змінювати наявні компоненти і перевикористовувати код перетворюють React розробку в безперервний процес поліпшення. Компоненти, створені під час роботи над тим чи іншим проектом, немає додаткових залежностей. Таким чином, ніщо не заважає використовувати їх знов і знов у проектах різного типу. Весь попередній досвід може бути легко застосований при роботі над новим сайтом або навіть при створенні мобільного додатка. Використовуючи передові можливості, такі як Virtual DOM або ізоморфний JavaScript, розробники React можуть з високою швидкістю створювати високопродуктивні програми, незважаючи на рівень їх складності. Можливість з легкістю наново використовувати вже наявний код підвищує швидкість розробки, спрощує процес тестування, і, як наслідок, знижує витрати. Той факт, що ця бібліотека розробляється і підтримується висококваліфікованими розробниками та набирає все більшої популярності з кожним роком, дає підстави сподіватися, що тенденція до подальших покращень продовжиться.

### 2.5.2. Простота

React.js просто легше зрозуміти. Компонентний підхід, чітко визначений життєвий цикл і використання простого JavaScript дозволяють дуже просто реагувати, вчитися, створювати професійні веб-сайти (і його мобільні програми). React використовує спеціальний синтаксис під назвою JSX, який дозволяє змішувати HTML з JavaScript. Це необов'язково; Розробник все ще може писати на звичайному JavaScript, але JSX набагато простіше використовувати.

### 2.5.3. Легко навчатись

Будь-хто з базовими знаннями програмування може легко зрозуміти React, тоді як Angular і Ember називають «предметно-орієнтованою мовою», пам'ятаючи, що вони важливі. Для React.js вам просто потрібні базові знання CSS і HTML.Нативний підхід.

React можна використовувати для створення мобільних додатків (React Native). І React є переконаним прихильником повторного використання, що означає, що підтримується багато повторного використання коду. Тож одночасно ми можемо створювати iOS, Android та веб-додаток.

### 2.5.4 Прив'язка даних

React використовує одностороннє прив'язування даних, а архітектура програми, яка називається Flux, контролює потік даних до компонентів через єдину контрольну точку, диспетчер. Простіше налаштовувати окремі компоненти великих доповнень ReactJS.

### 2.5.5. Продуктивність

React не пропонує жодної концепції вбудованого контейнера залежностей. Ви можете використовувати модулі Browserify, Require JS, EcmaScript 6, які ми можемо використовувати через Babel, ReactJS-di для автоматичного введення залежностей.

### 2.5.6. Тестуваність

Програмне забезпечення React.js дуже легко переглянути. Реактивні види можна розглядати як функції стану, тому ми можемо маніпулювати станом, переходимо до перегляду React.js і дивимося на вихід і роботу дій, подій, функцій.

## 2.6. Віртуальний дом

Двостороння DOM – це концепція програмування, де ідеальне або «віртуальне» представлення інтерфейсу користувача зберігається в пам’яті та синхронізується з «справжньою» бібліотекою DOM як вона є. Цей процес називається процесом.

Virtual DOM є відображенням у реальній пам'яті DOM. Це легкий об’єкт JavaScript, який є копією Real DOM.ReactJS не оновлює Real DOM безпосередньо, а оновлює Virtual DOM. Це спричиняє велику користь для ReactJS.

### 2.6.1. Проблема

Маніпуляція DOM — це набір сучасних інтерактивних мереж. На жаль, це також набагато повільніше, ніж більшість операцій JavaScript.

Ця повільність посилюється тим, що більшість фреймворків JavaScript оновлює DOM набагато більше, ніж слід. Наприклад, припустимо, що у вас є список з десяти пунктів. Зверніть увагу на перший елемент. Більшість фреймворків JavaScript відновлять весь список. Це в десять разів більше, ніж вам потрібно! Змінився лише один елемент, а інші дев’ять перебудовуються, як і раніше.

Відновлення списку не дуже важливо для веб-браузера, але сучасні веб-сайти можуть використовувати багато маніпуляцій DOM. Неефективні оновлення стали серйозною проблемою. Щоб вирішити цю проблему, люди в React просунули так званий віртуальний DOM.

Що робить маніпуляції з DOM повільними?

DOM є деревоподібною структурою даних. Тому зміни та оновлення самого DOM є досить швидкими. Але після зміни оновлений елемент та всі його нащадки (дочірні елементи) повинні бути повторно відмальовані (відрендерені) для оновлення програми UI. Повторний рендеринг – дуже повільний процес. Таким чином, чим більше компонентів UI, тим більш дорогими з точки зору продуктивності є оновлення DOM.

Маніпуляція з DOM є серцем сучасного інтерактивного Інтернету. На жаль, вони набагато повільніші за більшість JavaScript-операцій. Ситуація посилюється тим, що багато JavaScript-фреймворків оновлюють DOM частіше, ніж необхідно.

Допустимо, у нас є перелік із 10 елементів. Ми змінюємо перший елемент. Більшість фреймворків перебудують весь список. Це у 10 разів більше роботи, ніж потрібно! Тільки 1 елемент змінився, решта 9 залишилися колишніми.

Перебудова списку – це легке завдання для браузера, але сучасні веб-сайти можуть здійснювати безліч маніпуляцій з DOM. Тому неефективне оновлення часто стає серйозною проблемою. Для вирішення цієї проблеми команда React популяризувала щось під назвою віртуальний (virtual) DOM (VDOM).

### 2.6.2. Вирішення проблеми

У React для кожного об'єкта DOM є відповідний "віртуальний об'єкт" ДОМА». Віртуальний об’єкт DOM – це представлення об’єкта DOM у вигляді легкої копії.

Віртуальний об’єкт DOM має ті самі властивості, що й реальний об’єкт DOM, але він не має реальної можливості безпосередньо змінювати те, що відображається на екрані.

Маніпулювання DOM повільно. Віртуальна маніпуляція DOM набагато швидше, тому що на екрані нічого не відбувається. Уявіть маніпулювання віртуальним DOM як план плану, а не переміщення кімнат у реальному будинку. Як це допомагає? Коли ви представляєте елемент JSX, кожен окремий об’єкт є віртуальним DOM оновлено. Це звучить неймовірно неефективно, але вартість є незначною, оскільки віртуальний DOM можна оновлювати так швидко. Після

оновлення віртуальної DOM React завершує створення віртуальної DOM за допомогою знімок віртуальної DOM, зібраної безпосередньо перед оновленням.
Порівнюючи нову віртуальну DOM з попередньою версією, React з'ясовує, які віртуальні об'єкти DOM змінилися. Цей процес називається «дивергентним».
Як тільки React дізнається, які віртуальні об'єкти DOM змінилися, React оновлює ці об'єкти, і тільки ці об'єкти, на справжньому DOM. У нашому прикладі з React було б достатньо розумно відновити ваш список з одним елементом і залишити список у спокої.

Це дуже важливо! Реакція може оновлювати лише необхідні частини DOM. Репутація React щодо ефективності багато в чому пов'язана з цією інновацією.Загалом, ось що відбувається, коли ви намагаєтеся оновити DOM у React:

— весь віртуальний DOM оновлюється;

— віртуальний DOM порівнюється з тим, як він виглядав, перш ніж оновити його. Реагуйте, які об'єкти змінилися;

— змінені об'єкти і тільки змінені об'єкти оновлюються на реальному DOM;

— зміни на реальному DOM призводять до зміни екрана.

## 2.7. Single Page Application

У минулому, коли браузери були набагато менш функціональними, ніж сьогодні, а продуктивність JavaScript була слабкою, кожна сторінка приходила з серверу. Кожен раз, коли ви натискали щось, новий запит надсилався на сервер, і браузер пізніше завантажував нову сторінку.Тільки дуже інноваційні продукти працювали по-іншому та експериментували з новими підходами.

Сьогодні, популяризуючи сучасні фреймворки JavaScript, такі як React, програма зазвичай створюється як застосування однієї зупинки: ви завантажуєте програмий код (HTML, CSS, JavaScript), що закликає на запит будь-який JSON або

виконує дію на сервері, але зупинка, яку бачить користувач, ніколи не буде повністю знищена, а диск порожній.

Програми для однієї сторінки побудовані на JavaScript (або, принаймні, зібрані в JavaScript) і працювати у браузері. Технологія завжди однакова, але філософія та деякі ключові компоненти роботи програми запізнюються.

### 2.7.1. Переваги і недоліки SPA

Користувач SPA відчуває себе набагато швидше, тому що замість того, щоб чекати з'єднання клієнт-сервер і чекати, поки веб-браузер знову завантажить сторінку. Це відповідальність розробника програми, але ви можете мати переходи, напрямки та будь-які покращення UX, що, безумовно, краще, ніж традиційний робочий процес.

На додаток до швидшого використання досвіду користувача, споживати менше ресурсів, тому що ви можете зосередитися на наданні ефективного API, замість створення окремих частин макетів.

Це зробить ідеальним, якщо ви також створюєте мобільну програму поверх API, оскільки ви можете повністю повторно використовувати існуючий код.

Програми для однієї сторінки легко конвертуються в прогресивні веб-програми, що, у свою чергу, дозволяє вам локально кешувати і підтримувати автономний режим для ваших послуг, якщо ви не вмієте працювати. SPA найкраще використовувати, коли немає потреби в SEO (пошуковій оптимізації). Наприклад, для програм входу.

Пошукові системи, щодня вдосконалюючи, все ще мають проблеми індексація сайтів, створених за допомогою підходу SPA, а не традиційних сторінок, наданих сервером. Це стосується блогів. Якщо ви збираєтеся покладатися на пошукові системи, навіть не створюйте додаток для однієї сторінки, не маючи також частини, яка забезпечує сервер.

Під час кодування SPA ви пишете багато JavaScript. Поки програма може працювати довго, вам потрібно буде приділяти більше уваги можливим витокам

пам’яті – якщо раніше тривалість життя вашої сторінки зберігалася в хвилинах, то тепер SPA Пам’ять, яка збільшить використання пам'яті браузера набагато більше, і це призведе до неприємного повільного досвіду, якщо ви не подбаєте про це.

SPA-центри чудово підходять для роботи в команді. Розробники серверної частини можуть зосередитися лише на API, а розробники інтерфейсу можуть зосередитися на створенні найкращої взаємодії з користувачем за допомогою вбудованого серверного API.

Як односторінкові програми, односторінкові програми значною мірою покладаються на JavaScript. Це може призвести до поганого досвіду використання програм, які працюють на пристроях з низьким споживанням енергії. Крім того, деякі відвідувачі можуть вимкнути JavaScript, і вам також слід перевірити доступність усього, що ви завантажуєте.

### 2.7.2. Приклади застосування Single Page Applications

Прикладом технології SPA є реалізований сервіс Gmail від Google. З точки зору користувача, ця технологія забезпечує високу швидкість реагування на дії в інтерфейсі. Це не потрібно повне або навіть часткове перезавантаження WEB-сторінки з сервера – все візуальні елементи розробляються безпосередньо в браузері за допомогою технології JavaScript, маніпулюючи структурою DOM документа.

Таким чином, WEB-додатки стають дуже схожими на звичайні програми робочих станцій, які завантажують інформацію з Інтернету. Середовищем виконання для них є не операційна система, а браузер, який в результаті змушений нести все навантаження, пов’язане з виконанням стороннього коду.

Деякі визначні приклади:

-Gmail;

-Google Maps;

-Facebook;

-Twitter;

-Google Drive;

### 2.7.3. Перевизначення навігації

Коливання від вас можуть з'являтися в навігації браузера для користувачів, URL-адресу потрібно зберігати вручну. Серед фреймворків вже граються в них (як Ембер), інші вимагають бібліотеку, яка переможе (наприклад, React Router).

В чому проблема? Кілька речей для відвідувачів, які були встановлені програмами з однієї зупинки. Ціна піднялася до помилки таємної проблеми з «кнопкою обертання»: при навігації по URL-програм вона не змінилася. Ви можете перейти на сайт, який ви бачили давно.

Тепер я можу переглянути проблему за допомогою додаткової історії інтерфейсу API, як зрозуміти веб-переглядач, або більше години ви будете грати в бібліотеку, як внутрішній віце-король, використовуючи API, наприклад React Router.

## 2.8. Бібліотека реалізації керування станом системи Redux

Redux – це не що інше, як репозиторій, що містить стан програм. Це стає болючим завданням, коли розмір програми стає більшим, щоб контролювати стан кожного компонента вашої програми. Таким чином, продукція приходить на допомогу, підтримуючи та оновлюючи стан кожного компонента нашої програми.

Redux часто буває незрозумілим, коли ми вперше пробуємо свої сили. Отже, я наведу приклад, щоб ви зрозуміли, що таке редукція, і навіщо вона взагалі потрібна. В активному додатку все є компонентом. Уявляєте, як складно стає, якщо у вашому додатку багато компонентів, схожих на ті, що показані на малюнку 2.3, тому стає важко контролювати потік даних від батьківських компонентів до дочірніх.

Це перша причина, через яку ми використовуємо redux, тому що він контролює стан всіх компонентів за нас.

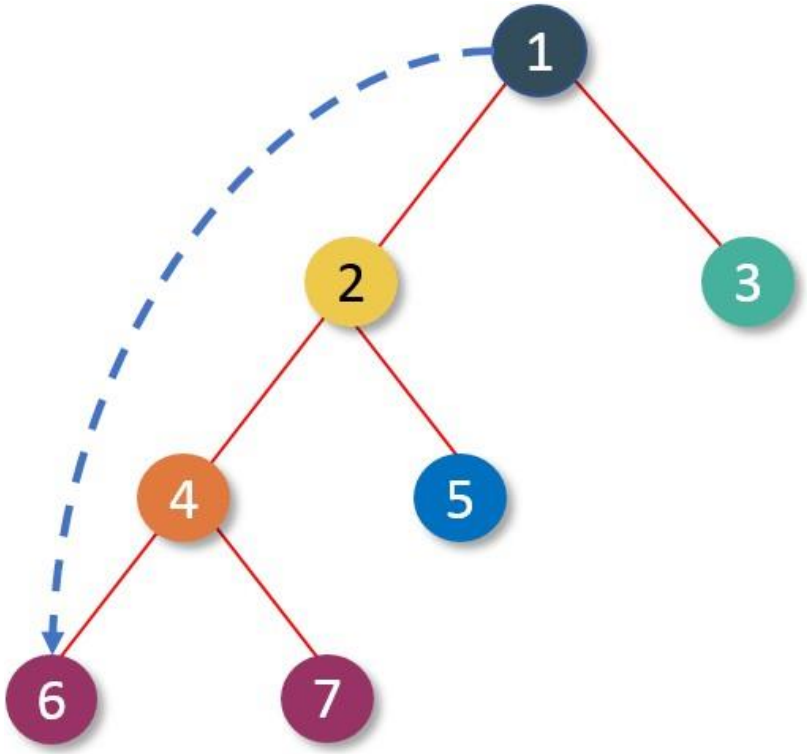


Рис. 2.3. Зразок дерева компонентів React

У реактивному застосуванні існує односпрямований потік даних за односпрямованим я маю на увазі потоки даних від батьківських компонентів до дочірніх компонентів не навпаки, так що ви посилаєте дані з батьківських компонентів до дочірніх компонентів у вигляді того, що ми називаємо реквізитами, тоді цей дочірній компонент використання цієї підставки.

Redux — це бібліотека відкритого коду JavaScript для керування станом програми. Після виконання дії в цьому випадку замовлення взуття є акцією, яку ви чекаєте на доставку, але чи відбувається це так, як тільки ви замовляєте щось із фліпкарту, ви отримаєте доставку відразу. Ні, насправді це вимагає часу, і є процес, який слідує кожен раз, коли ви замовляєте щось з вашого улюбленого веб-сайту.

Таким чином, подібно до редуксу після виконання дії, існує термін, що називається диспетчером, який посилає вашу дію на редуктор. Так само, як і після розміщення замовлення, ваш пакет буде відправлений у найближчий склад на вашу

адресу. Така ж робота в редукції здійснюється шляхом відправки. Існує кілька понять, які необхідно зрозуміти, щоб зрозуміти redux.

Я намагаюся пояснити їх за допомогою прикладу (рис. 2.4). Скажімо, ви замовили туфлі-шльопанці після замовлення взуття, які ви отримуєте від агента доставки в певний час. Отже, ваше замовлення взуття — це дія, яка є одним із понять redux.

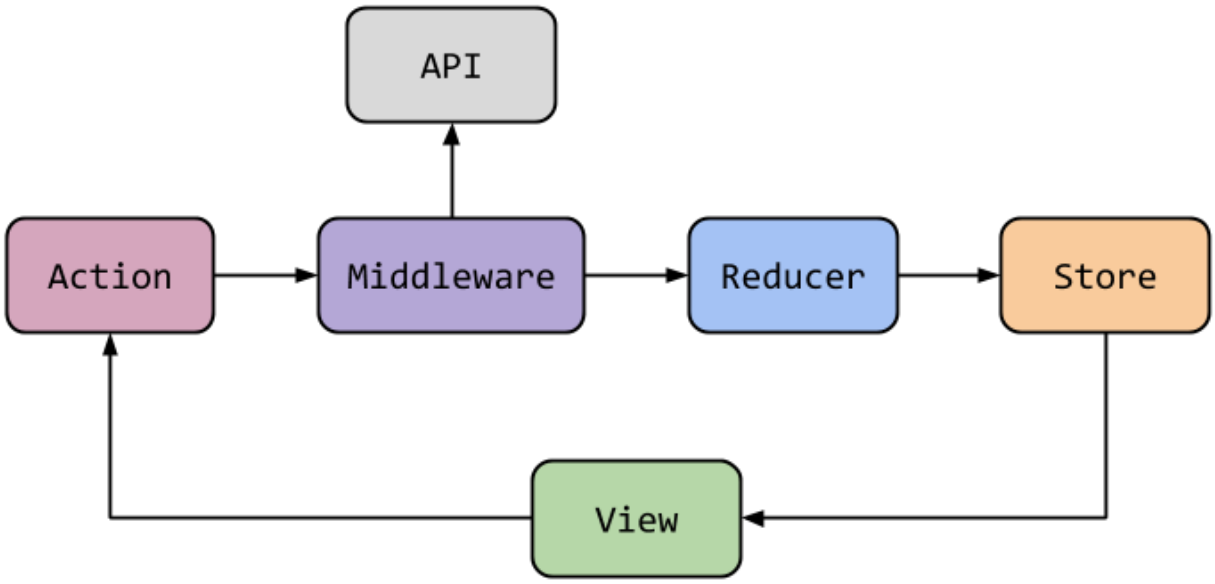


Рис. 2.4. Архітектура Redux

Тепер Reducer дивиться на дію і робить те, що необхідно, щоб відповідно зберегти дані. Редуктор — це не що інше, як файл, який складається з оператора case і використовується для зберігання даних у сховищі та отримання оновленого значення стану зі сховища. Тому, коли статус оновлюється, значення в магазині також оновлюється.

# РОЗДІЛ 3.
# ОПИС РЕАЛІЗАЦІЇ ВЕБ-ДОДАТКА НА БАЗІ JAVASCRIPT ВИКОРИСТОВУЮЧИ БІБЛІОТЕКУ REACT

Цей програмний продукт розроблено та протестовано в середовищі розробки програмного забезпечення WebStorm. Ця система також має архітектуру клієнт-сервер (рисунок 3.1), яка складається з сервера, веб-клієнта та бази даних.

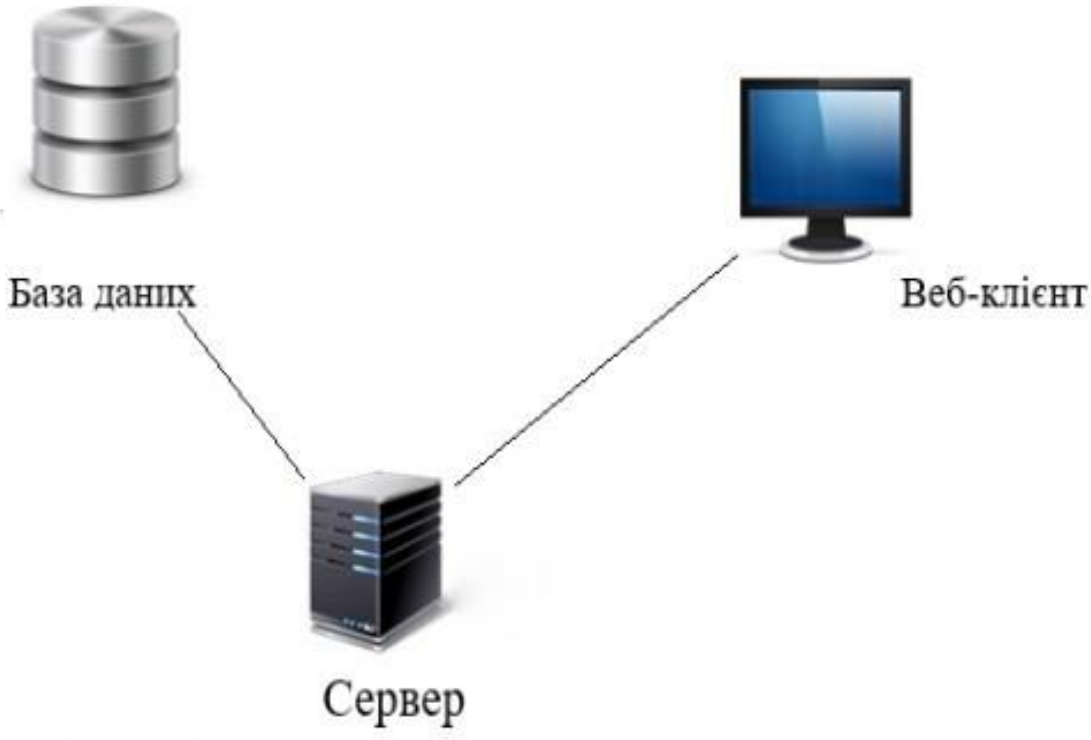


Рис. 3.1. Архітектура системи

Рисунок 3.1 — Дана система складається з таких частин:

— веб-клієнт React.js;

— сервер Express.js;

— база даних MySQL.

### 3.1. Опис функціональності системи

Модель «Студент» веб-додаток системи управління проектами має одного актора, яким є студент. На малюнку 3.2 представлена прецедентна діаграма, яка описує дії та функції учня в системі.

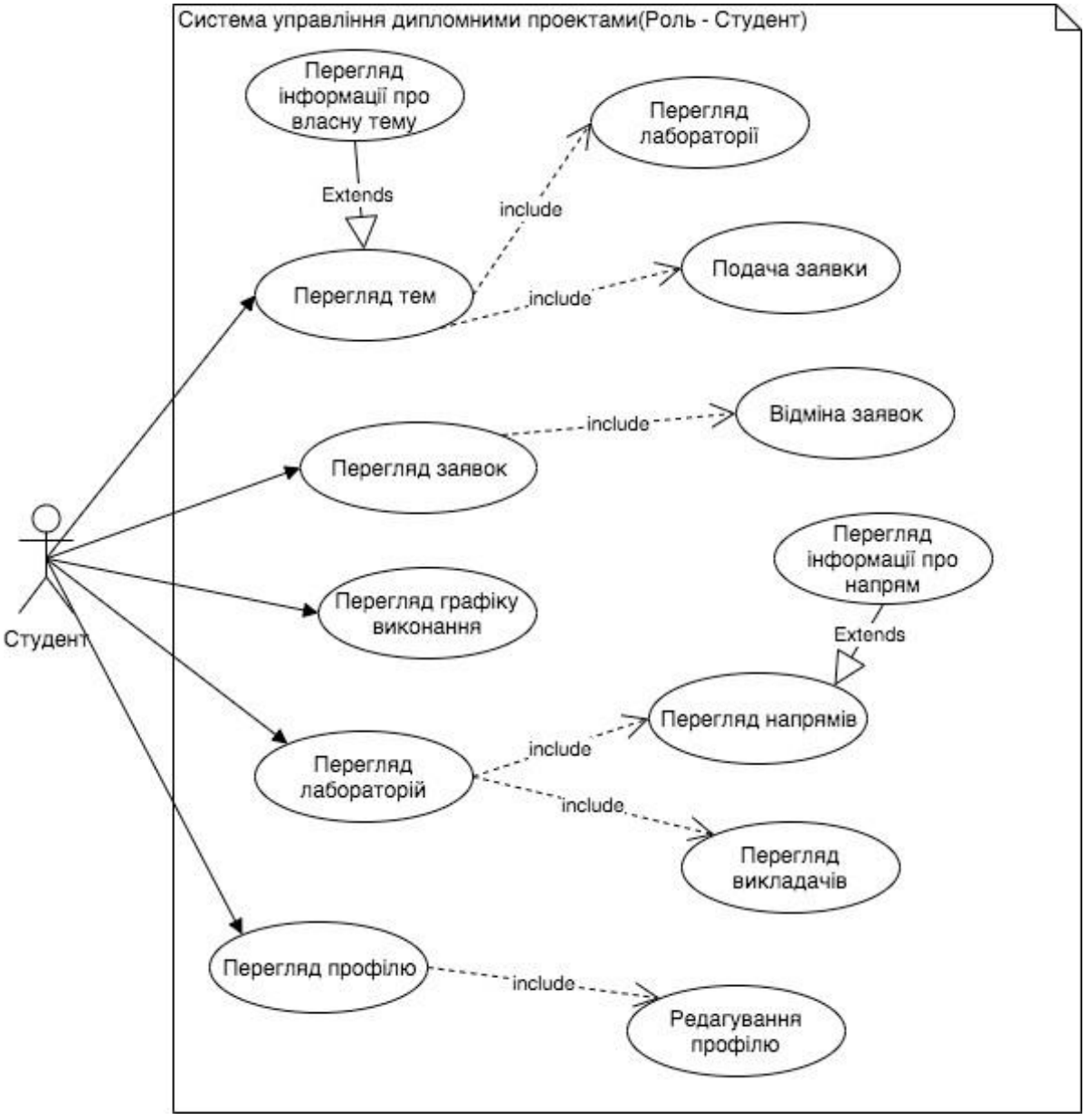


Рис. 3.2. Діаграма прецедентів веб-застосунку

Діаграма прецедентів веб-застосунку Користувачеві надається такі функції:

— перегляд профілю;
— редагування профілю;
— перегляд лабораторій;
— перегляд викладачів;
— перегляд напрямків лабораторії;
— перегляд інформації про напрями лабораторії;
— перегляд графіку виконання;
— перегляд заявок;
— відміна заявок;
— подача заявок;
— перегляд тем дипломних робіт;
— перегляд інформації про власну тему;
— авторизація в систему.

### 3.2. База даних

Побудова бази даних на базі даних MySQL. Архітектура бази даних повністю відповідає тематиці роботи. На малюнку 3.3 показана схема бази даних:

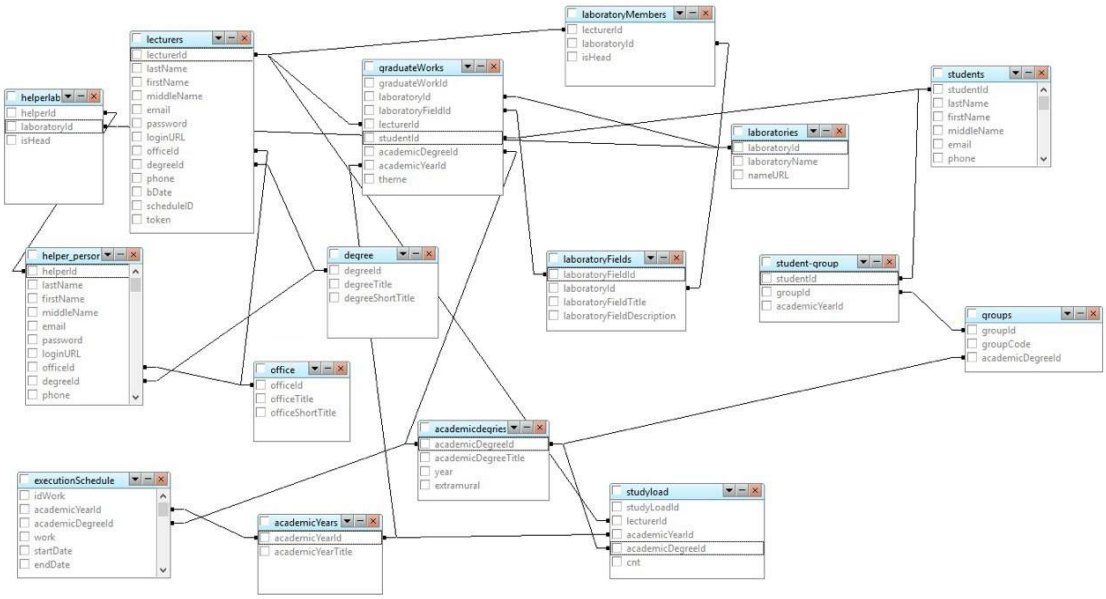


Рис. 3.3. Схема бази даних

### 3.3. Створення веб-застосунку за допомогою React.js

Для забезпечення зручності роботи з даними розроблено веб-додаток, написаний мовою програмування JavaScript. Це пов'язує проблему кросплатформності і дозволяє запускати програму на будь-якому пристрої, що має вихід в Інтернет.

Щоб створити проект, вам потрібно встановити бібліотеку create-react-app. Це полегшує створення програмного та бібліотечного коду, а також виконання різноманітних поточних завдань розробки, таких як тестування, збірка та підготовка.

Щоб створити новий проект, вам потрібно запустити команду create-react-app project_name, а потім у поточному каталозі буде створено новий проект з усіма необхідними залежностями, а також на сервері, щоб ви могли легко впоратися з ним. Команда yarn start запускає сервер, відстежує файли та перебудовує програму, коли ви вносите зміни до цих файлів. Параметр --open (або просто -o) автоматично

відкриває ваш браузер за адресою http: // localhost: 3000 /.

Основною одиницею React є компоненти, які використовуються як будівельні блоки. Компоненти можуть приймати різні параметри. Ці параметри називаються реквізитами.

Назви та значення стилів зазвичай відповідають тому, як CSS працює в Інтернеті. На додаток до способу написання імені, React використовує стиль camelCase, наприклад backgroundColor, а не background-color. Те ж саме стосується назв подій.

Папка компонентів містить основні компоненти програми, папка configs містить налаштування, компоненти містять сторінки програми, маршрути - відображення, магазин - управління станом системи, стилі - основні стилі, утиліти - допоміжні функції.

Для управління станом системи використовувалася бібліотека Redux. Redux — це стан контейнера для програм JavaScript. Bin допомагає користувачам оптимізувати код за допомогою програм. Крім того, це передбачає покращення досвіду вручну, наприклад, редагування коду в реальному часі в поєднанні з налагоджувачем, який працює під час роботи. На рисунку 3.4 представлена файлова структура веб-клієнта:

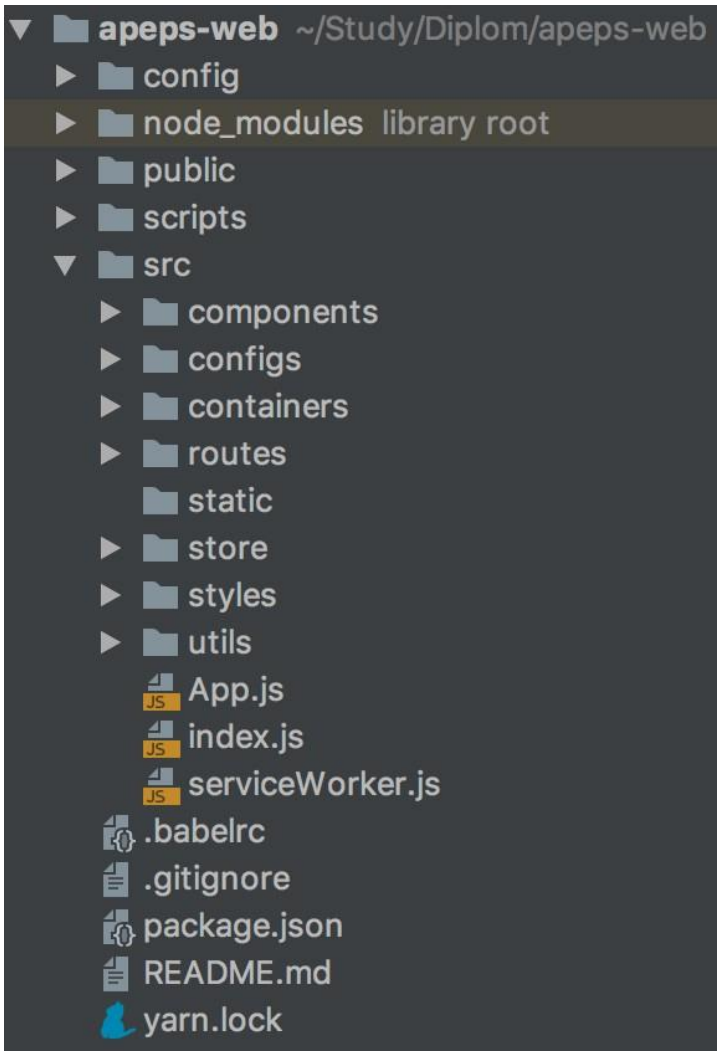

Рис. 3.4. Файлова структура веб-клієнта

Сховище було налаштовано за допомогою функції createStore бібліотеки redux, в якій параметри редуктора і редюсера скидаються, де редуктор — rootReducer, а в полі підсилювача відображається функція composeWitDools бібліотеки. Ми додаємо sagaMiddleware до параметрів проміжного програмного забезпечення за допомогою функції applyMiddleware з бібліотеки redux, а також створюємо createLogger з бібліотеки redux-logger, якщо ми знаходимося в періоді розробки. Після ініціалізації магазину запускається sagaMiddleware.

Store (Малюнок 3.5) складається з дій, які пересилають дані з програмами в сховище, також є єдиним джерелом інформації для сховища і для їх відправки необхідно запустити команду store.dispatch ().

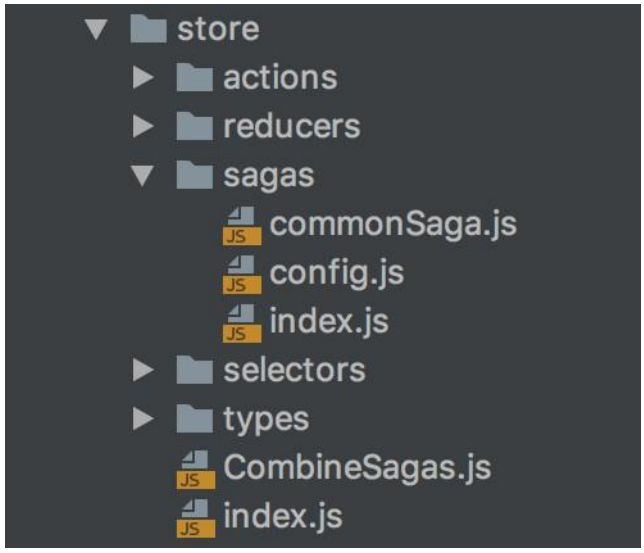


Рис. 3.5. Файлова структура системи керування станом системи

Магазин також містить редуктори, які вказують, як змінюється стан програм у відповідь на дії, надіслані до магазину. Важливо пам’ятати, що дії не описують, як змінюється програма, а лише описують те, що сталося. Редуктори - це чисті функції детермінована функція без побічних ефектів. Визначена функція:

- функція, яка завжди повторює одне й те саме значення для певного входу. Допоміжна функція — це функція, яка змінює щось за межами функції. Потім усі редуктори об’єднуються в один – rootReducer за допомогою функції composredreducers бібліотеки redux. Це типи дій, які потрапляють в редуктори і виводять їх у змінну для кожного модуля.

Selectors містить селектори створені для мемоізації даних. Reselect має функцію createSelector для створення мемоізованих селекторів. Селектор перераховувати дані тільки тоді коли один з його аргументів змінився. Селектори

можуть використовуватись в якості параметрів для інших селесторів.

В файлі config.js прописані налаштування для запитів на сервер, а в файлі commonSaga.js написана універсальна saga, яка приймає на всіх всі actions, якімістять слово _REQUEST кидає запит на сервер, обробляє його в залежності від налаштувань прописаних в файлі config.js.

Для взаємодії магазину з компонентами реагування використовується вища функція замовити підключення з бібліотеки react-redux. Функція вищого порядку — це функція, яка повторює іншу функцію і може приймати інші функції як аргументи. Функція connect приймає mapStateToProps і mapDispatchToProps як параметри.Для того, щоб react знав про store потрібно обернути наш застосунок в Provider з бібліотеки react-redux, який приймає в якості props store.

В якості маршрутизатора використовувався компонент Router з бібліотеки React Router. Компонент Router очікує лише один елемент як дочірній. Маршрутизатор приймає історію реквізитів, що дозволяє легко керувати історією сеансу програми. Створено компонент програми, який охоплює всю програму.

Для написання стилів використовувалася бібліотека styled-components, що також дозволяє використовувати різні теми зі стилями. Перевага цього підходу в тому, що вам не потрібно створювати файли css.

Для сповіщень було написано, щоб написати компонент NotificationProvider, який використовує React Context API.

Модальний компонент вікна був написаний за допомогою Portals, який відтворює будинок вузла за межами єпархійного дому батьківського компонента.

### 3.4. Створення серверної частини за допомогою Express.js

Ланцюжок був побудований на платформі Node.js. Ця платформа дозволяє створювати сервіси rest api за допомогою мови програмування Javascript.

Щоб створити проект, вам потрібно налаштувати експрес, створити екземпляр

програми та вказати порт, на якому програма працюватиме.

Для обробки запиту http post у Express.js версії 4 і вище, вам потрібно встановити проміжний програмний модуль під назвою body-parser.Модуль body-parser отримує всю частину тіла вхідного потоку запитів і надає його req.body. Проміжне програмне забезпечення було включено до складу Express.js, але тепер його потрібно встановлювати окремо. Цей модуль аналізатора тіла аналізує дані кодування JSON, буфера, рядка та URL-адрес надіслано за допомогою HTTP-запиту публікації.

Щоб мати можливість надсилати запити на сервер з іншого домену, потрібно встановити модуль cors.

Щоб фіксувати помилки, потрібно написати проміжне програмне забезпечення (функцію проміжної обробки), яке перехоплює помилки і в результаті передає цю помилку.

Для надання статичних файлів використовується функція проміжної обробки express.static. Щоб надати доступ до файлів, необхідно вказати каталог, в якому лежать статичні файли.

Щоб автоматично перезапустити сервер, коли файл змінився, вам потрібно встановити модуль nodemon.Для запуску сервера необхідно виконати команду nodemon app.js.

Сервер виконує такі функції:

— виконує зв’язок з базою даних

— виконує завантаження файлів;

— виконує відправку електронних листів;

— реалізує зв’язок з клієнтською частиною за допомогою http запитів.

Папка api містить модуль маршрутизації та системні модулі. У папці db є модуль до бази даних. У папці пошти є модуль для відправки електронних листів. У загальній папці є статичні файли. Папка node_modules містить встановлені залежності для розробки сервера.Файл package.json містить список підключених

модулів для даного проекту з npm, команди для запуску проекту. Список модулів з package.json:

— body-parser версії ^1.18.3;

— cors версії ^2.8.5;

— express версії ^4.16.4;

— multer версії ^1.4.1;

— mysql версії ^2.17.1;

— nodemailer версії ^6.1.1;

— nodemon версії ^1.19.0.

Файлову структуру сервера зображено на рисунку 3.6:

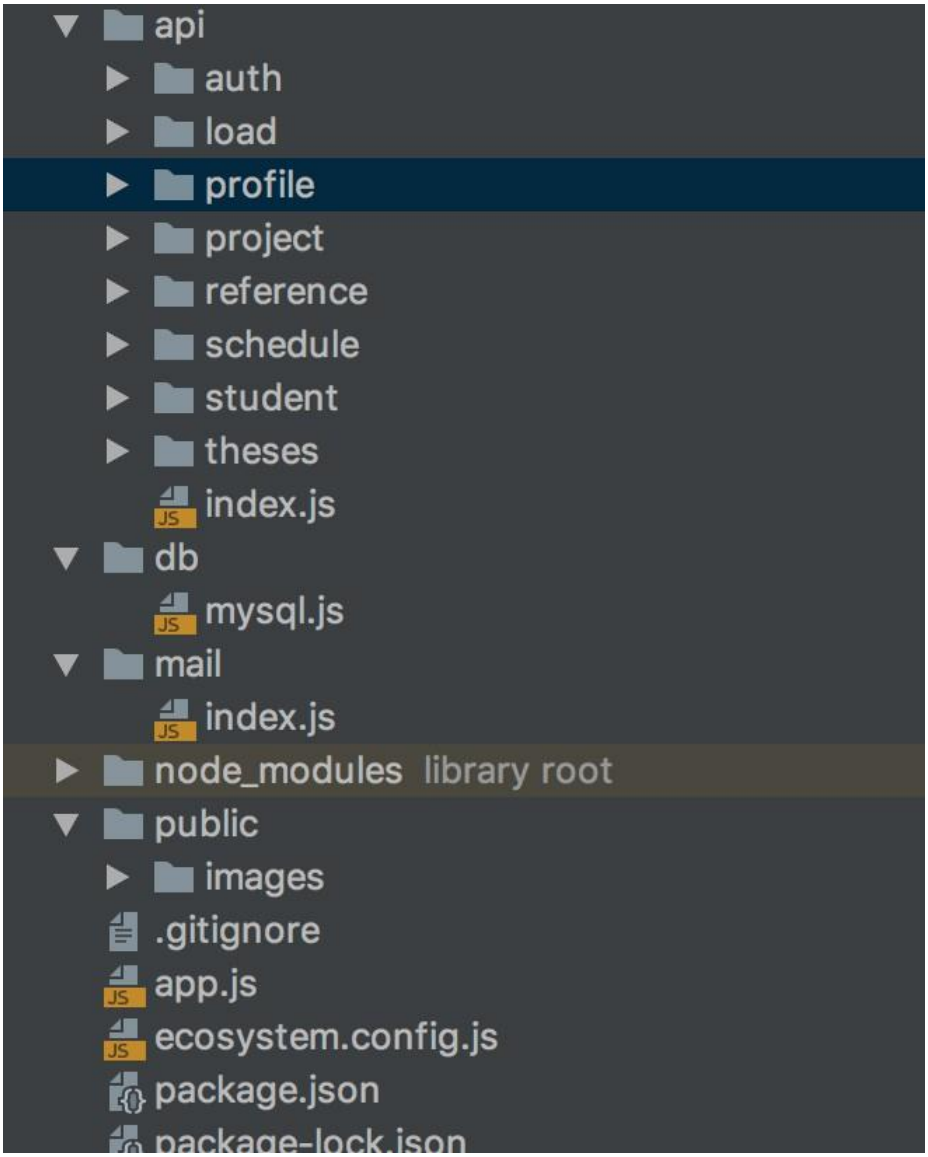


Рис. 3.6. Файлова структура сервера

Щоб додати функціональність підключення бази даних MySQL до надбудови Express, все, що вам потрібно зробити, це завантажити модуль mysql у доповнення.

Щоб завантажити файли на сервер, необхідно встановити модуль multer, встановити параметри типу та розміру файлів, вказати нове ім’я файлу та каталог, де він буде зберігатися.

Найбільша відмінність, яке ви можете тут помітити, полягає в тому, що Express за умовчанням дає вам роутер. Вам не потрібно вручну розбирати URL, щоб напишіть, що обіграти, замість цього ви визначаєте маршрутизацію програми за

допомогою app.get, app.post, app.put і так далі, і вони вже транслюються у відповідні HTTP-запити .Однією з найпотужніших концепцій, реалізованих Express, є шаблон.

Проміжне програмне забезпечення. Щоб обробляти помилки в Express, необхідно створити спеціальний проміжний обробник - проміжне програмне забезпечення з чотирма вхідними параметрами. Відстеження помилок має бути останньою функцією, доданою app.use, і приймати наступний зворотний виклик. Його можна використовувати для поєднання кількох обробників помилок

# РОЗДІЛ 4.
# РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКА НА БАЗІ JAVASCRIPT ВИКОРИСТОВУЮЧИ БІБЛІОТЕКУ REACT

Щоб використовувати веб-програму в режимі реального часу, у вас повинен бути встановлений будь-який Інтернет-браузер. Ви повинні переконатися, що ваша система відповідає цим вимогам. Хоча, як правило, це не проблема, коли мова йде про вимоги до обладнання, ви можете помітити, що це, наприклад, зовсім інша історія, коли мова заходить про процес. Наприклад, користувачі Firefox у Windows 2000 помітять, що вони не зможуть оновити Firefox 12 до 13 найближчим часом, оскільки Mozilla відтепер зменшила підтримку цієї операційної системи. Підтримка операційних систем Google Chrome починається з Windows XP SP2, OS X 10.5.6, Ubuntu 10.04, Debian 6, OpenSuse 11.3 і Fedora Linux 14.

Щоб реалізувати доступ до системи лише авторизованим корисувачам, було розробленно авторизацію (рисунок 4.1).

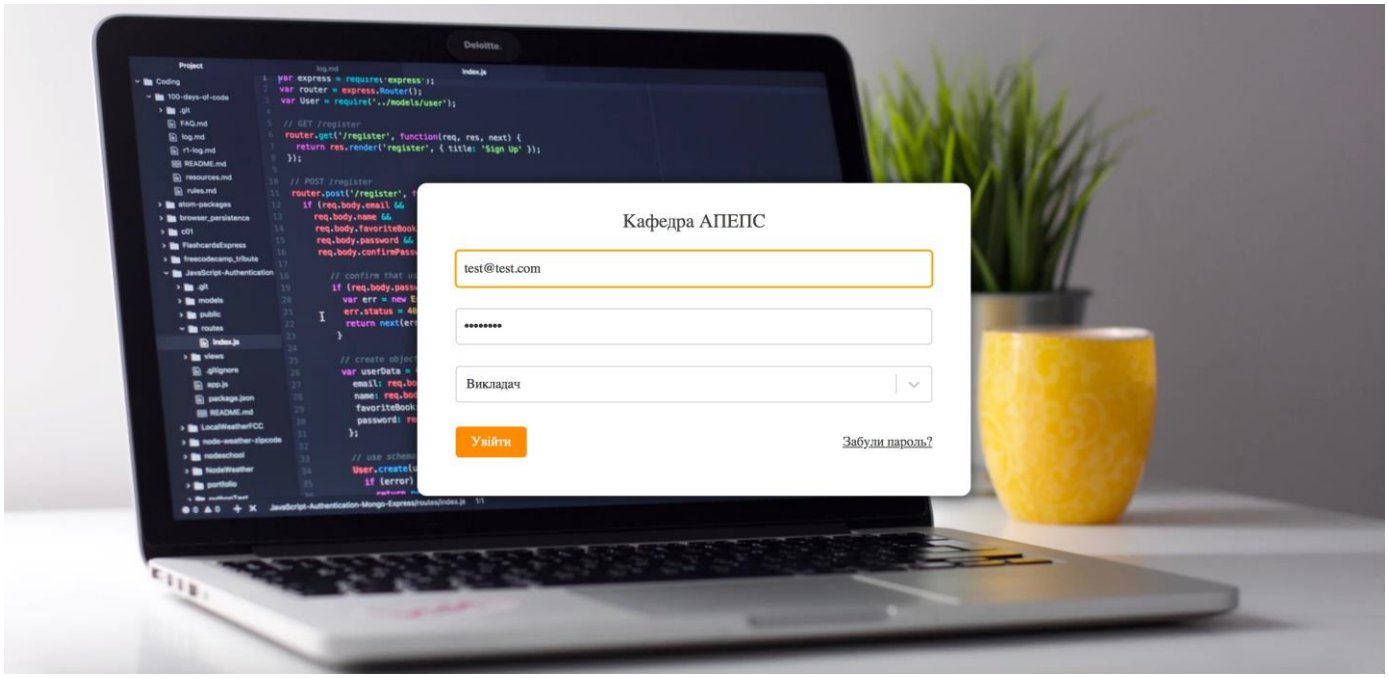


Рис. 4.1. Сторінка авторизації

Якщо користувач забув пароль, то він має можливість надіслати пароль на свою електронну адресу (рисунок 4.2).

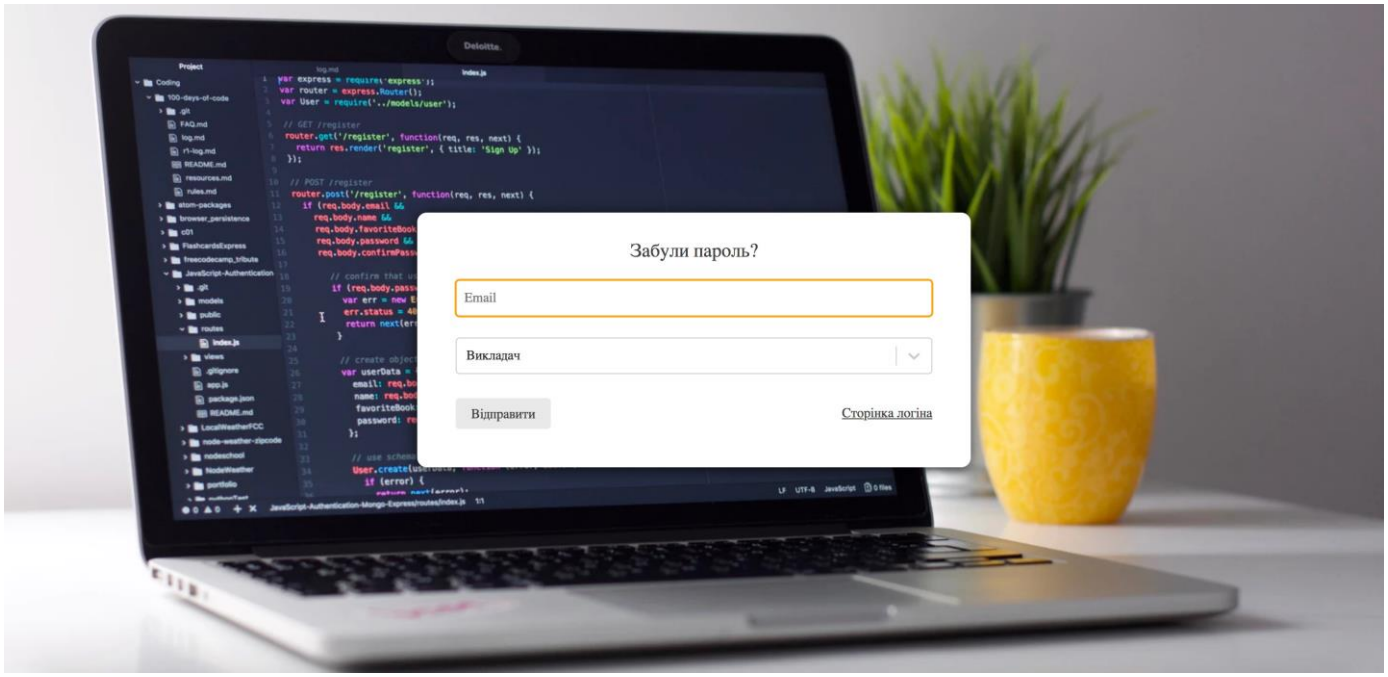


Рис. 4.2. Сторінка відновлення паролю

Після успішної авторизації копіювальний апарат має можливість шукати, переглядати дипломні роботи, фільтрувати дані про лабораторію, бібліографію та викладача, а також шукати документи.

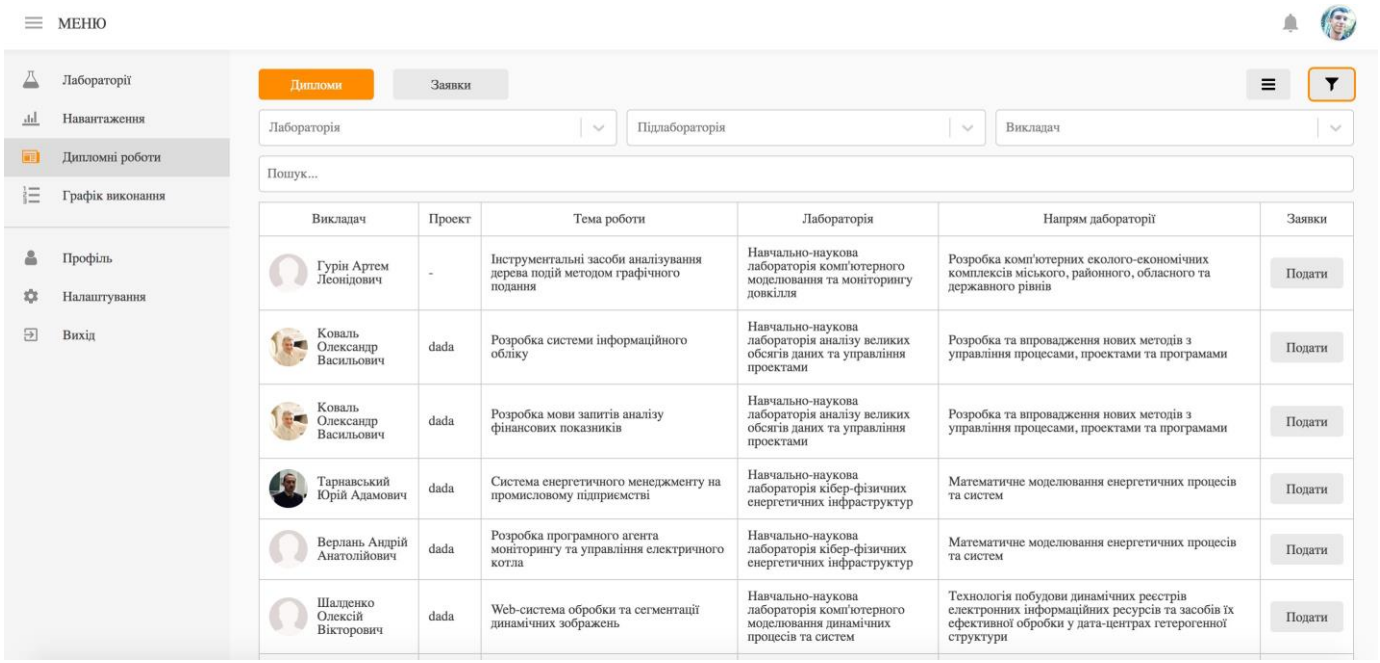

Рис. 4.3. Сторінка тем дипломних робіт

Студент має можливість подати максимум 7 дипломних робіт. Для підтвердження відправлення заявки студент повинен підтвердити свої наміри відправити заявку в модальному вікні (рисунок 4.4).

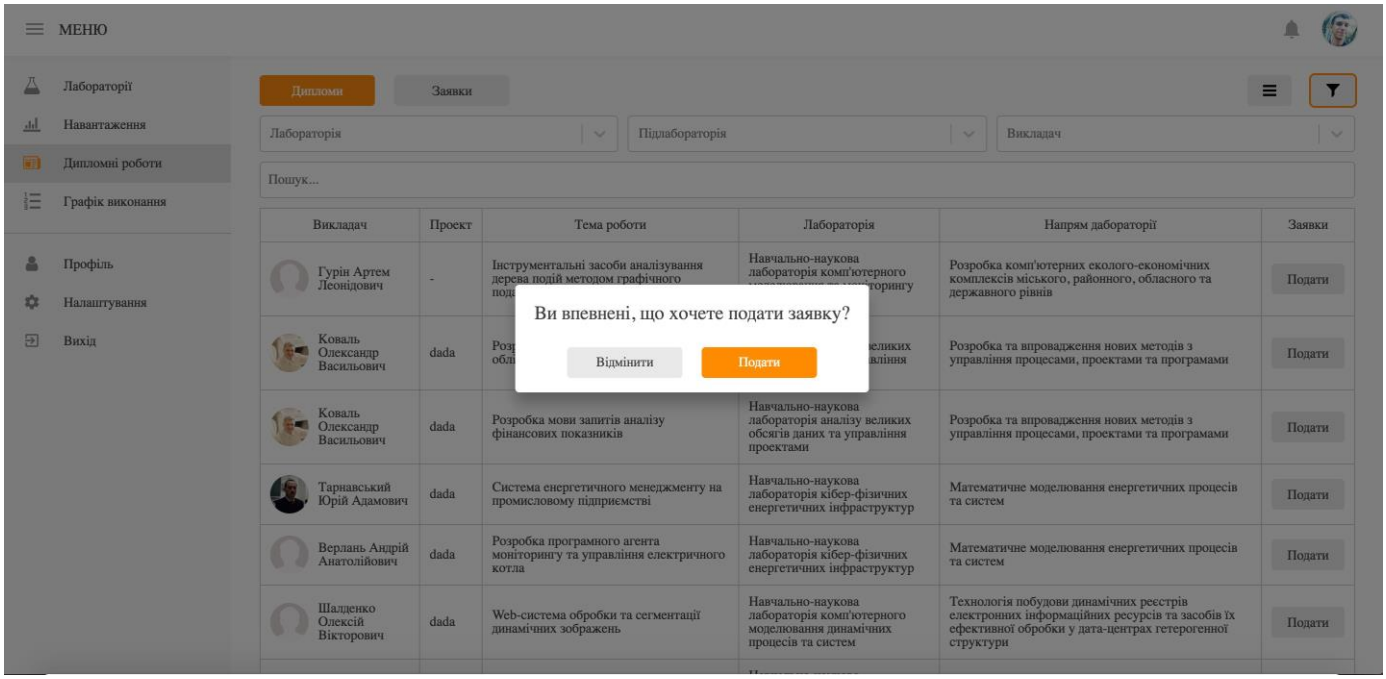


Рисунок. 4.4. Модальне вікно підтвердження відправки заявки

Студент має можливість переглянути свої заявки. Відбувається також зміна зовнішнього вигляду дипломних робіт (рисунок 4.5).

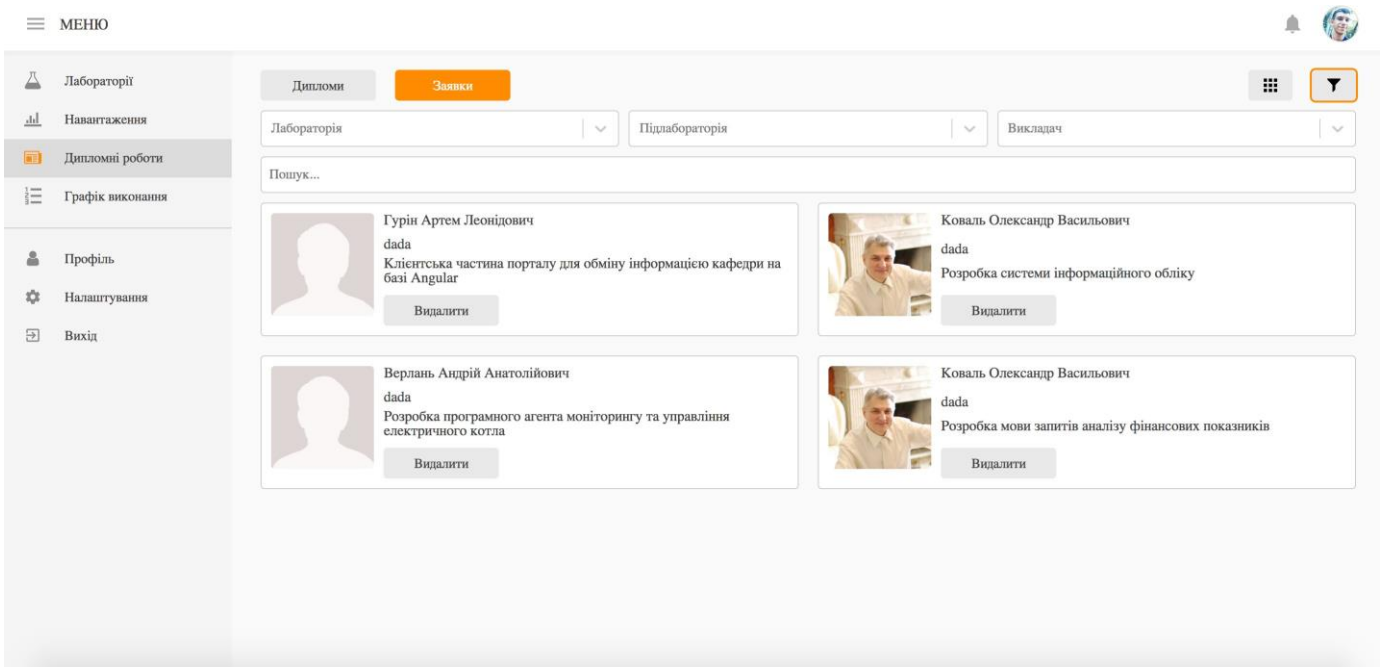


Рис. 4.5. Сторінка заявок на дипломні роботи

Щоб видалити заявку, студент повинен підтвердити свої наміри видалити. Застосунок має кросбраузерну та адаптивну версію, а тому його можна використовувати на мобільних пристроях (рисунок 4.6).

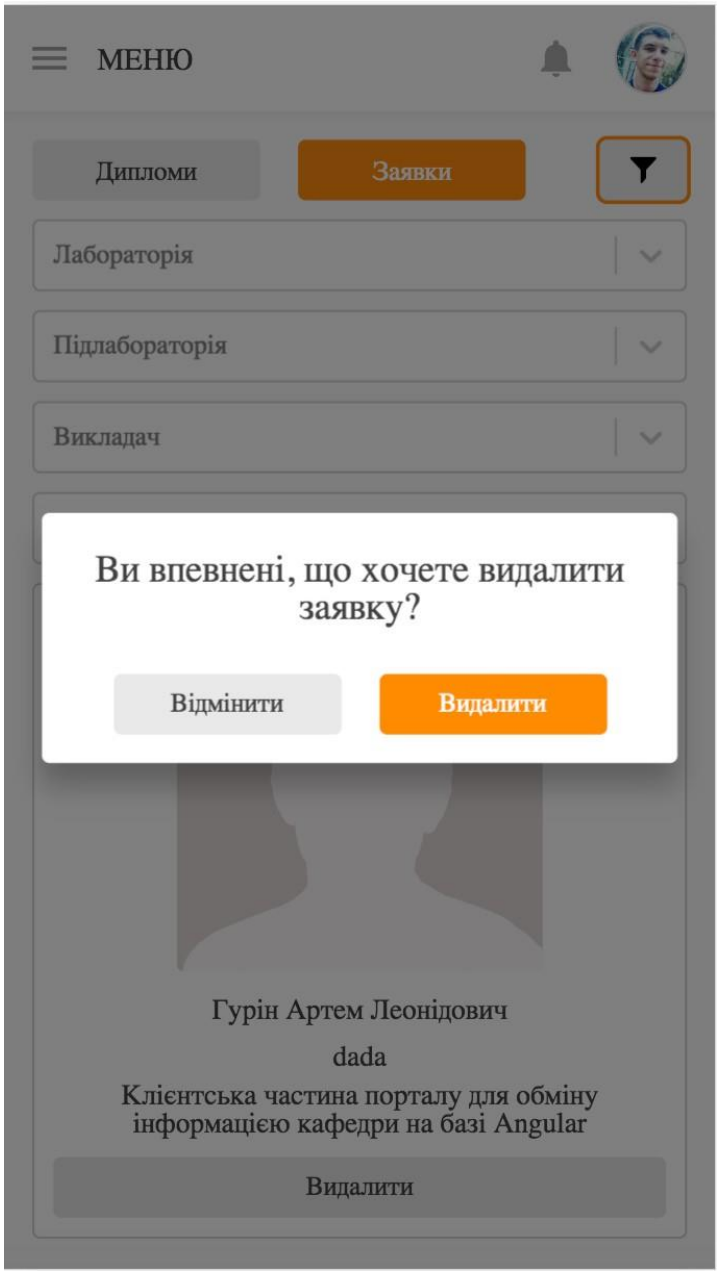


Рис. 4.6. Модальне вікно підтвердження видалення заявки

Після того як викладач прийняв заявку студента, студент має змогу переглядати інформацію про свою дипломну роботу (рисунок 4.7), а саме:

— контактні дані викладача;

— інформацію про тему дипломної роботи;

— графік виконання дипломної роботи.

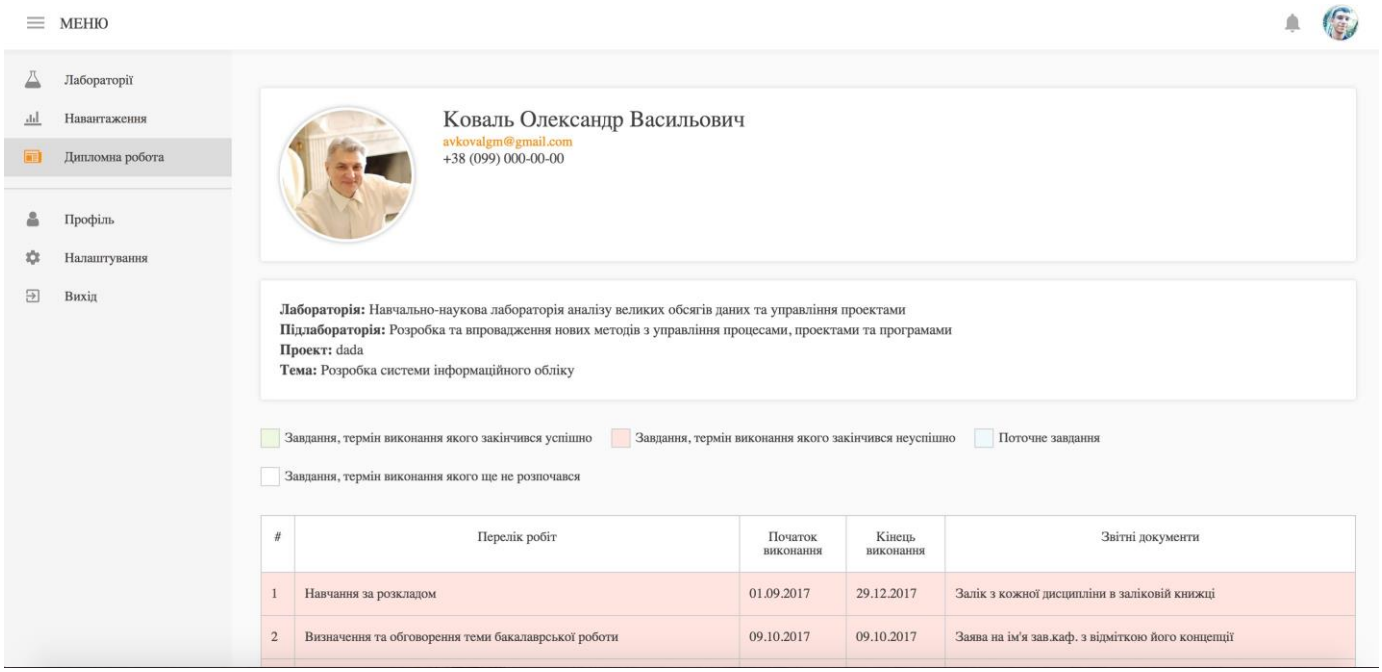


Рис. 4.7. Зразок інтерфейсу дипломної роботи студента

Студент має можливість переглядати свій профіль (рисунок 4.8).

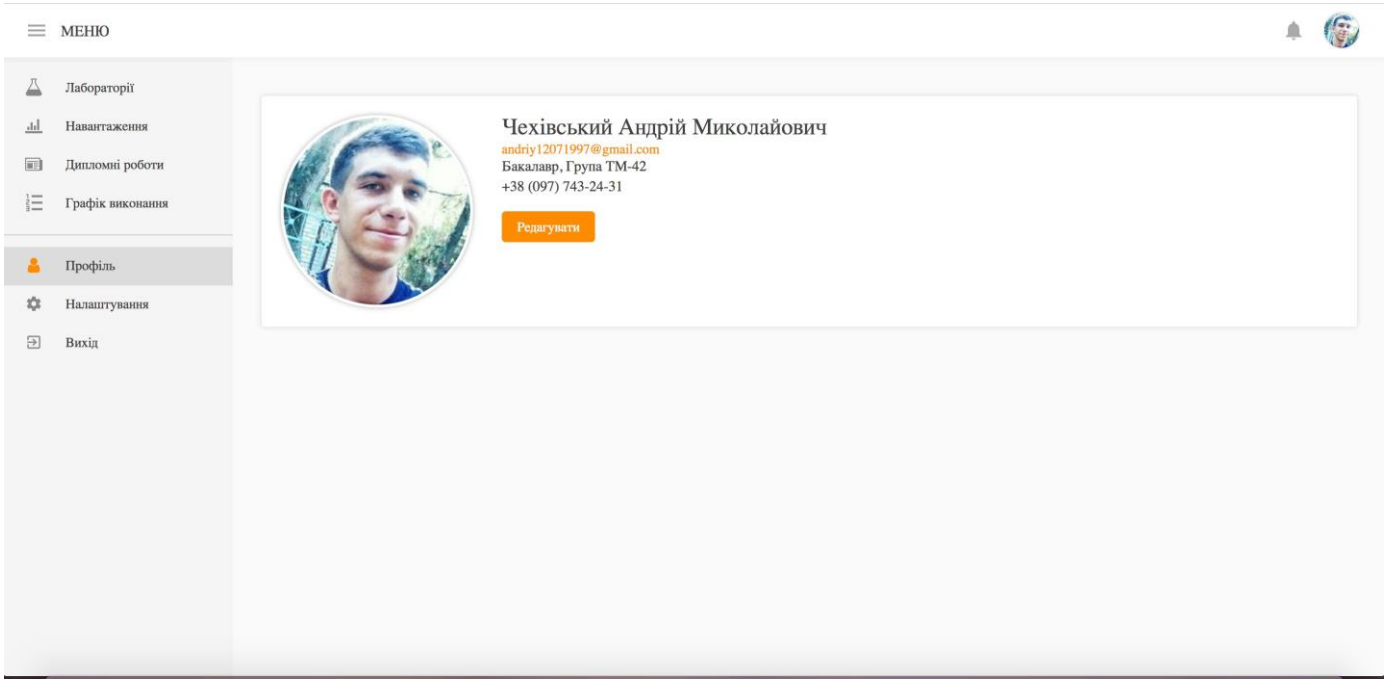


Рис. 4.8. Зразок інтерфейсу профіля студента

Студент має можливість змінити свою аватарку. При натисканні на аватар користувача пропонується вибрати фотографію(рис. 4.9).

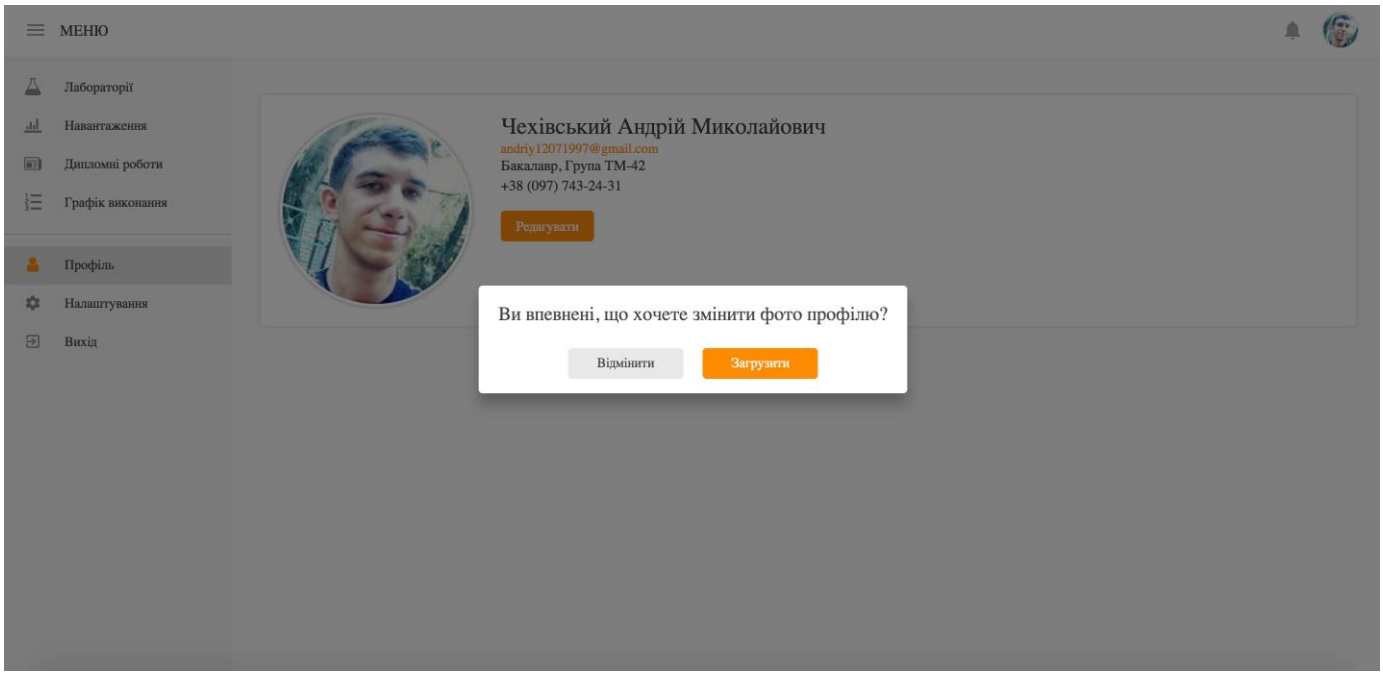


Рис. 4.9. Зразок інтерфейсу вибору фото для зміни аватара користувача

Після вибору фотографії відкривається модальне вікно, в якому користувач може змінити аватар, а також зберегти його (рис. 4.10).

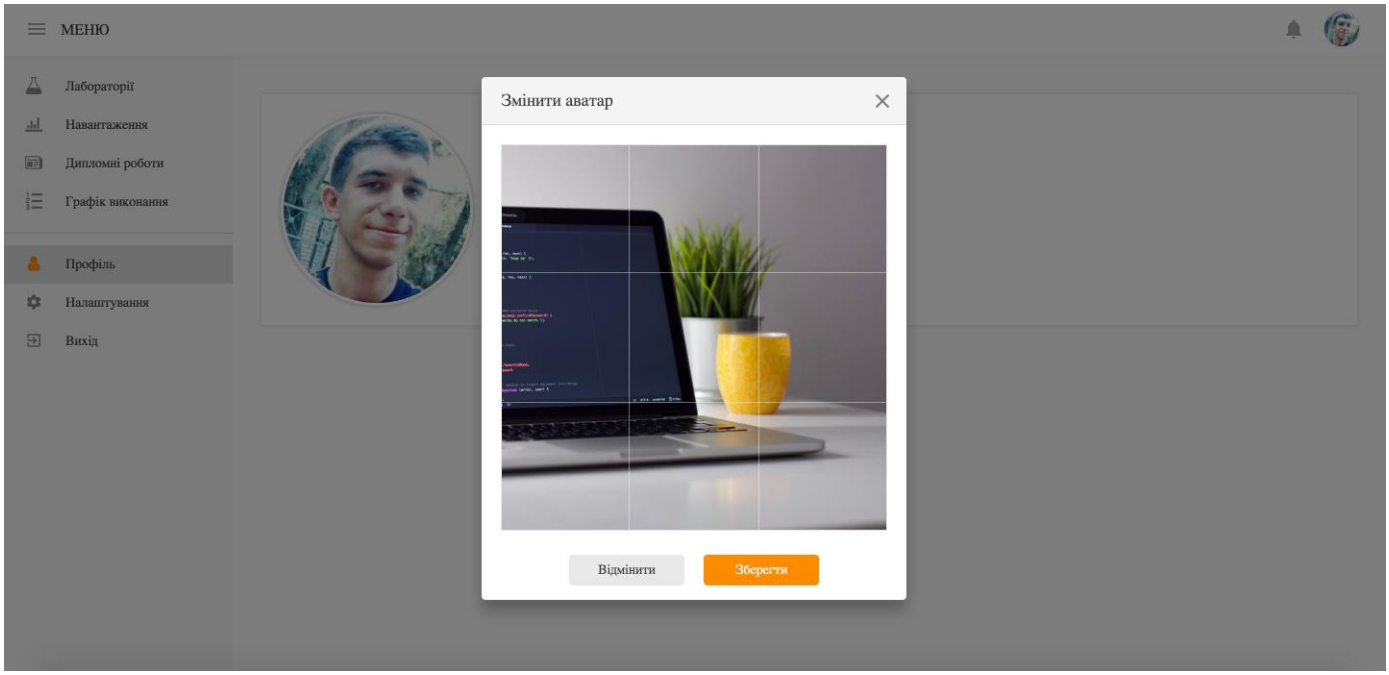


Рис.4.10. Зразок інтерфейсу зміни аватара користувача

Студент має можливість змінювати свою електронну адресу, номер телефону, а також кольорову тему застосунку (рисунок 4.11).

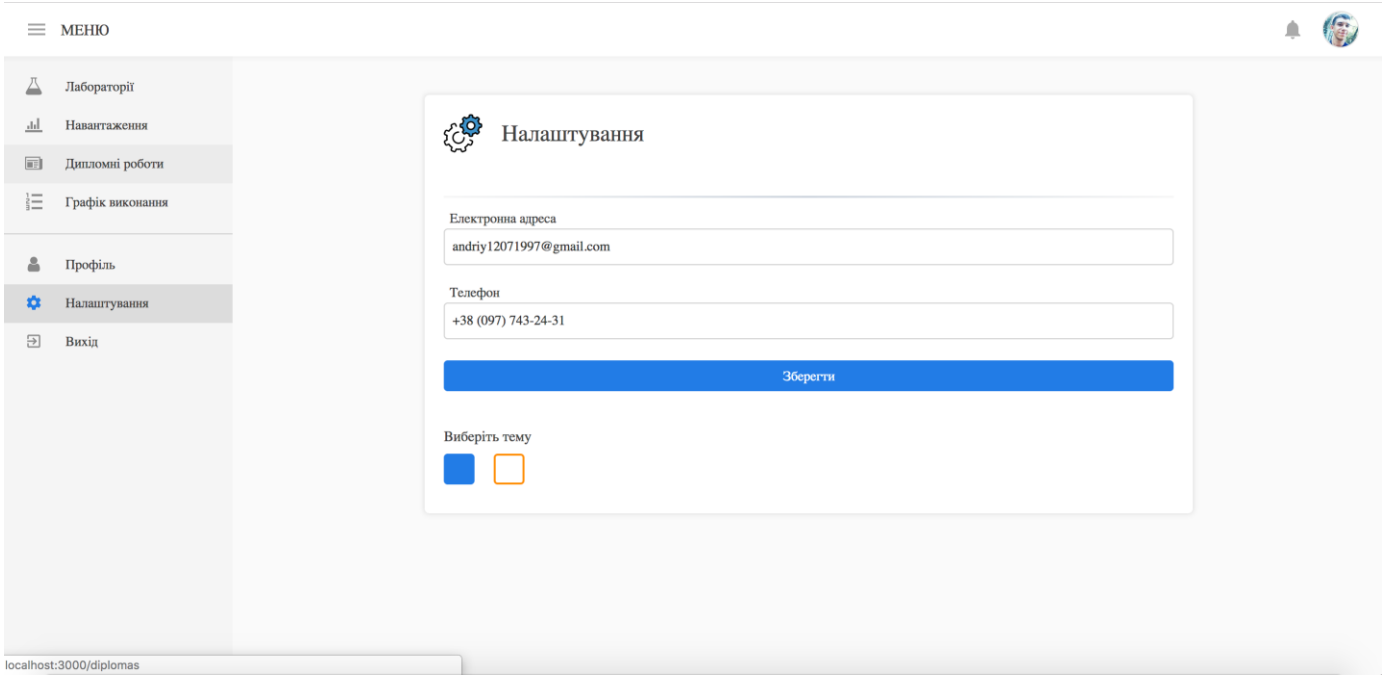


Рис. 4.11. Зразок інтерфейсу налаштувань користувача

Студент має можливість переглядати лабораторії (рисунок 4.12).

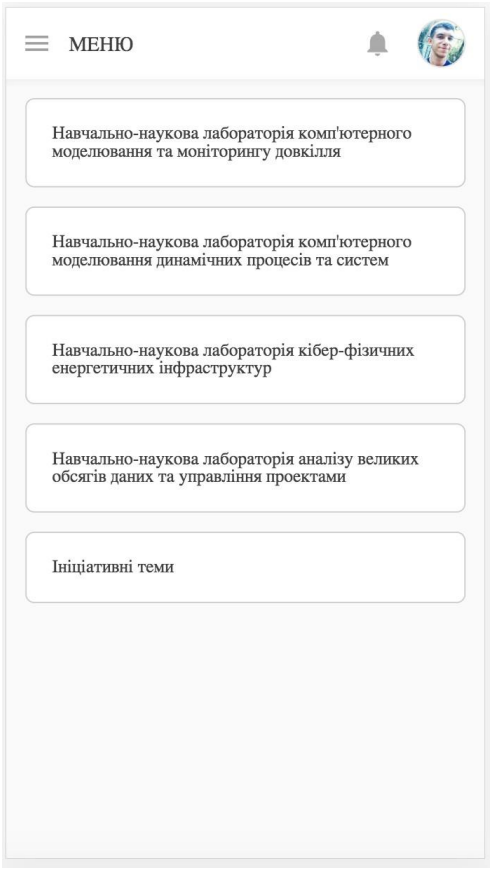


Рис. 4.12. Зразок інтерфейсу лабораторій

Студент має можливість переглянути інформацію про напрямок роботи лабораторії та результати роботи лабораторії (рисунок 4.13).

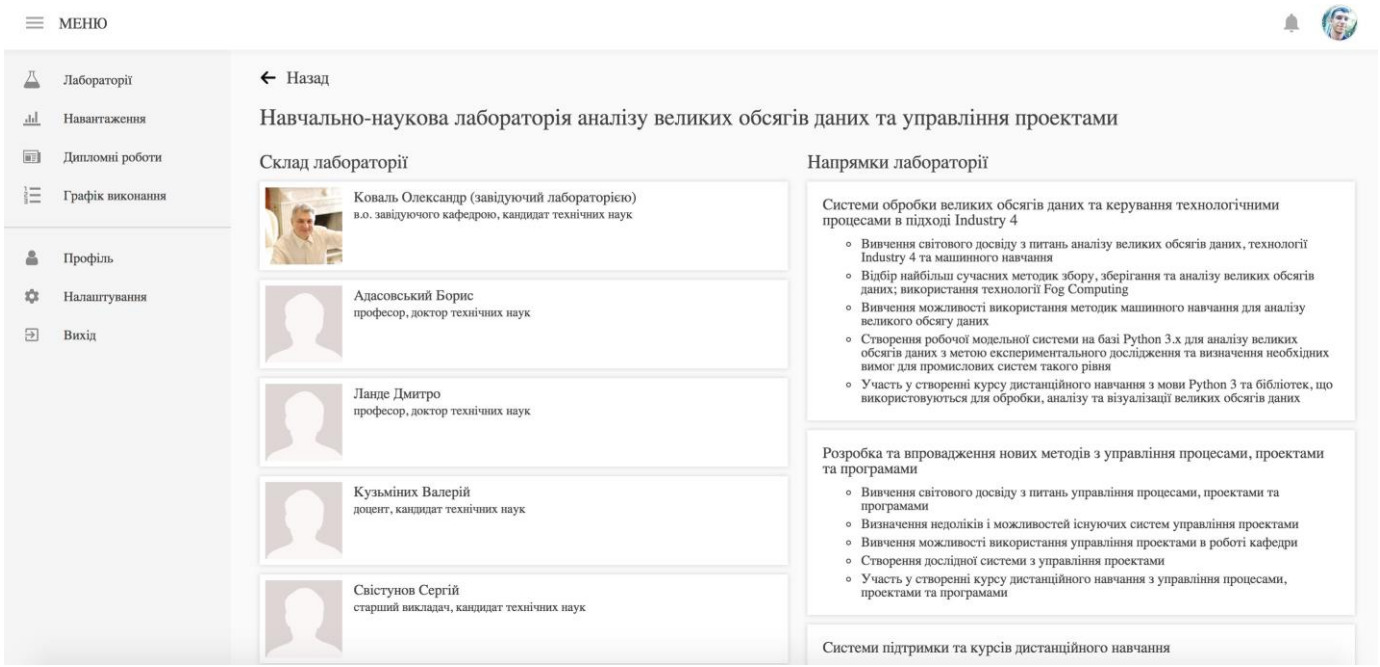


Рис. 4.13. Зразок інтерфейсу напрямків та складу лабораторії

При натисканні на викладача студент переноситься до дипломної роботи, де за фільтрами буде обраний викладач (рисунок 4.14).

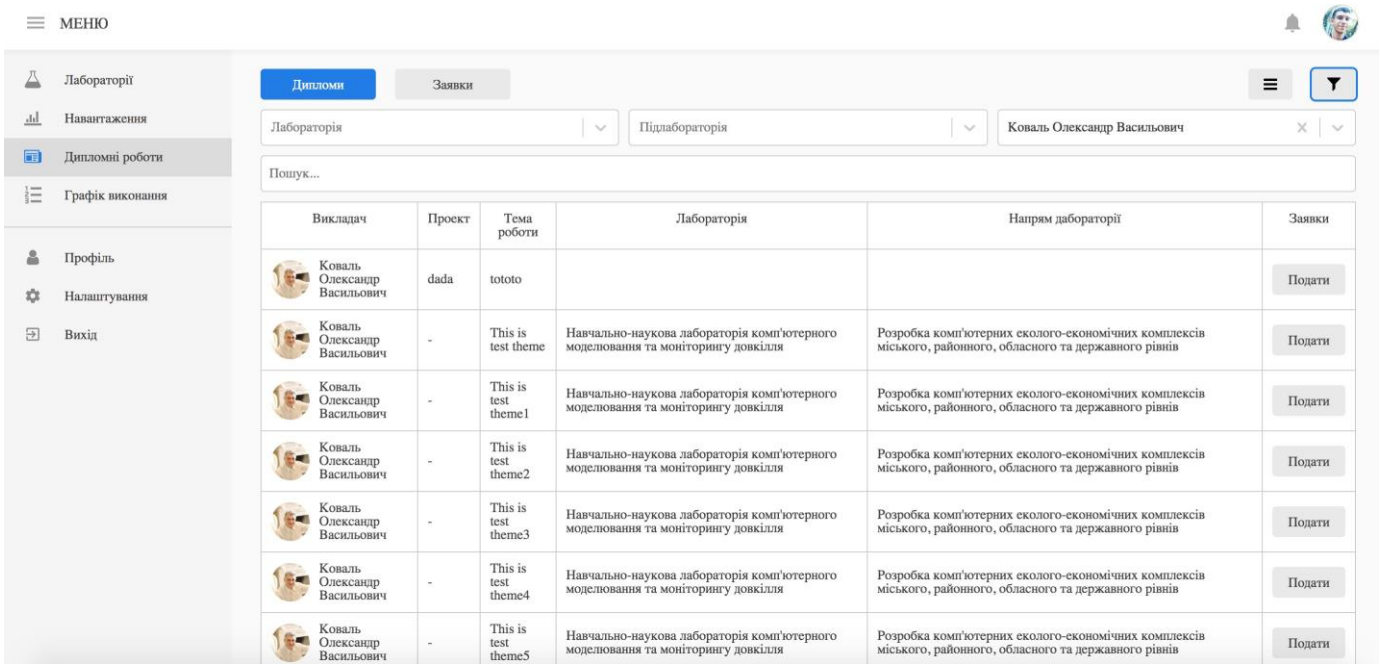


Рис. 4.14. Зразок інтерфейсу фільтрування тем дипломних робіт

Студент має можливість переглядати навантаження для викладачів (рисунок 4.15)


Рис. 4.15. Зразок інтерфейсу навантаження на викладачів

Студент має можливість переглянути розклад дипломної роботи (рисунок 4.16).

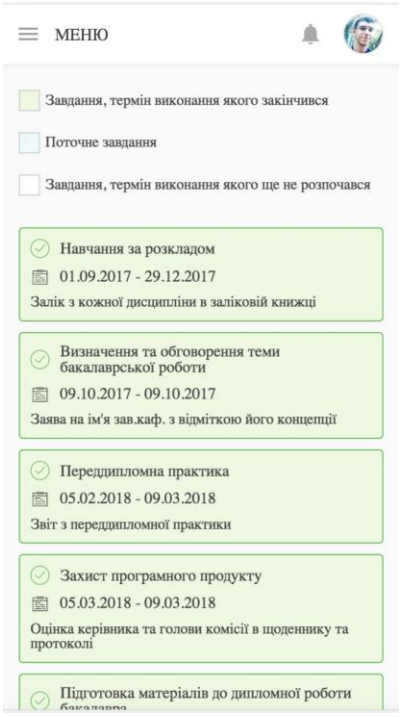


Рис. 4.16. Зразок інтерфейсу графіку виконання дипломної роботи

Також в застосунку є контексне меню. Для того, щоб його відкрити потрібно натиснути на аватар користувача в правому верхньому куті (рисунок 5.17).

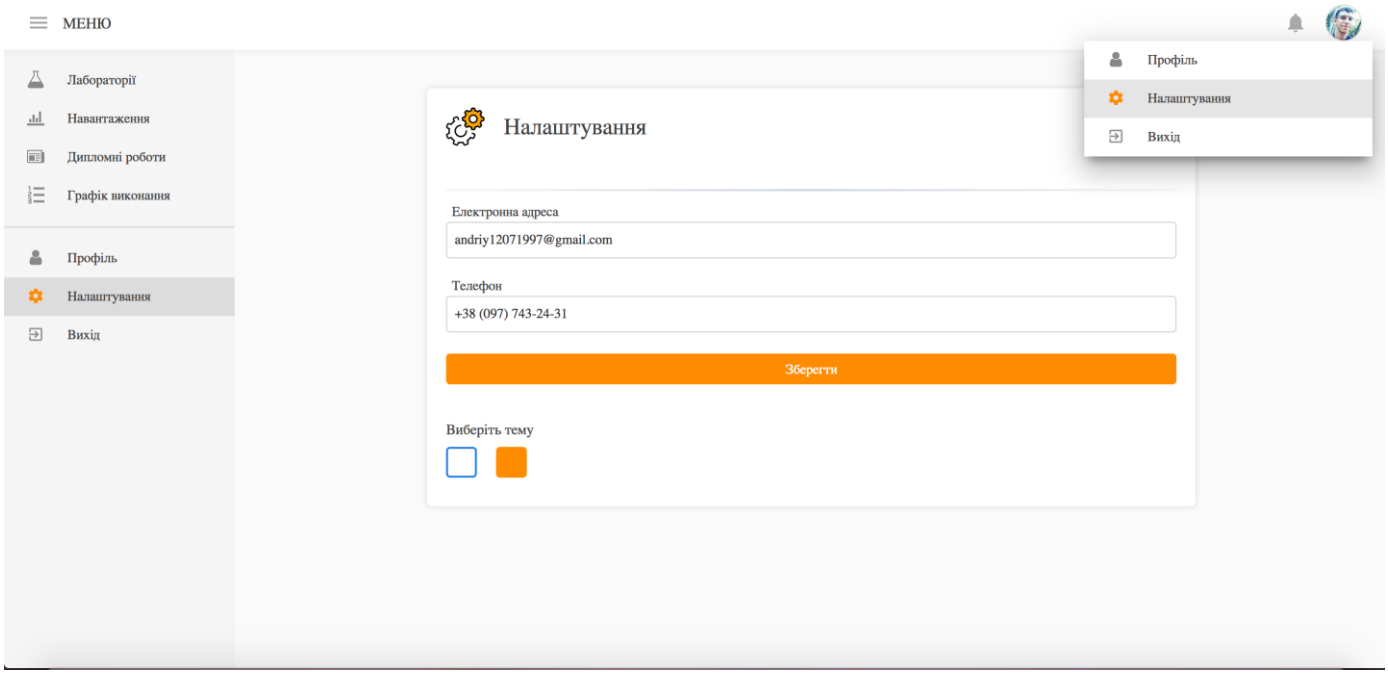


Рис. 4.17. Зразок інтерфейсу контекстного меню

Також бокове меню може скриватися, а на мобільних пристроях виїжджати збоку (рисунок 4.18).

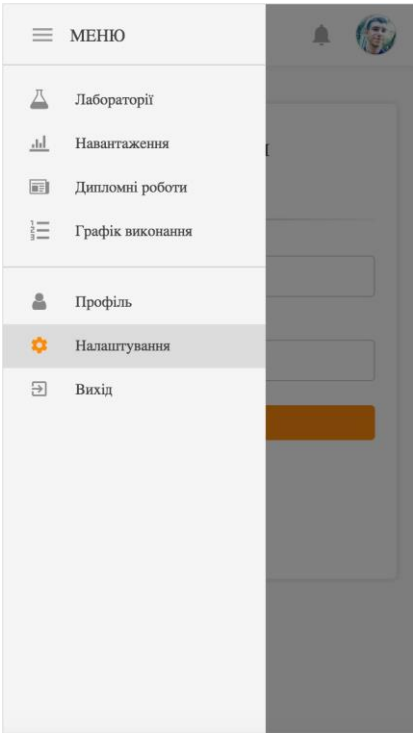


Рис. 4.18. Зразок інтерфейсу бокового меню

## 4.1. Технічні вимоги до середовищ використання

Мінімальними вимогами до процесорів, що працюють у браузері Windows, є Internet Explorer 8. Для цього потрібен процесор 233 МГц; мінімум 64 МБ ОЗУ (для Windows XP), рекомендовано 512 МБ RAM; принаймні 150 МБ (Windows XP), 70 МБ (Windows Vista) вільного місця на диску.

## 4.2. Загальний опис роботи веб-застосунку

Технологія використання програми така: користувач заходить в систему, після чого користувач бачить перелік дипломних робіт, на які він має можливість подати заявку. Після подачі заявки користувач чекає, поки викладач прийме заявку користувача. Після прийняття заявки користувач бачить інформацію про дипломну роботу.

### 4.2.1. Анотація веб-застосунка

Розроблений програмний продукт представляє собою веб-застосунок і дозволяє користувачам шукати та переглядати теми дипломних робіт, подавати і відміняти заявки на теми дипломних робіт, переглядати інформацію по своїй дипломній роботі, переглядати навантаження на викладачів. Модуль написано мовою програмування JavaScript, з використанням бібліотеки React.js яка базуеться на мові програмування JavaScript, бібліотеки керування данними та  станом усього веб-додатка за допомогою системи Redux.

### 4.2.2. Загальні відомості

Для роботи програми необхідно встановити платформу Node.js. Разом з нею встановлюється менеджер пакетів npm. Після цього потрібно встановити create- react-app. Для цього виконуємо команди “npm install -g create-react-app”.

Для запуску додатку потрібно відкрити проект у WebStorm або відкрити в терміналі директорію з проектом і виконати наступні команди: “npm install && npm start”.

Основний модуль був написаний на мові JavaScipt за допомогою бібліотеки React у середовищі розробки Webstorm.

### 4.2.3. Функціональне призначення

Система має функції пошуку та перегляду тем дипломних робіт, подачі та відміни заявок на теми дипломних робіт, перегляду інформації по своїй дипломній роботі після прийняття заявки викладачем, перегляду графіка виконання дипломної роботи, навантаження на викладачів, інформації про склад та напрямки лабораторії, редагування та перегляду профілю, входу в систему, відновлення паролю.

### 4.2.4. Опис логічної структури

Логічна структура програми складається з класів та функції, які виконують поставлені задачі. Програма містить директорію компонентів, де знаходяться спільні компоненти для користувацького інтерфейсу. За стилізацію додатку відповідає директорія стилів. За маршрутизацію додатку відповідають модулі в директорії шляхів. За збереження даних та їх зміну відповідають модулі сховищ, які знаходяться в директорії сховищ. Також існують допоміжні модулі, які формують структуру програми — це конфіги, допоміжні функції тощо.

Усі модулі програми є незалежними та здатні до масштабування. Саме завдяки правильній побудові структури програми, вона є гнучкою, легкою в подальшій розробці та підтримці.

### 4.2.5. Технічні засоби

Модулі розроблено у середовищі розробки Webstorm, на комп'ютері, що використовував операційну систему windows 10. Мовою програмування було обрано JavaScript та бібліотеку React. Мобільний додаток написаний для операційної системи macOS, Windows та Linux.

### 4.2.6. Виклик та завантаження

Для запуску веб-застосунку потрібно відкрити будь-який бразуер в операційній системи macOS, Windows або Linux.

Для використання даного модулю не потрібно ніяких дій, оскільки він автоматично спрацьовує після запуску клієнтського додатку.

### 4.2.7. Вхідні та вихідні дані

Вхідними даними є інформація, яку користувач вводить в застосунку. Вихідними даними є списки тем дипломних робіт, заявок, навантаження,графік виконання, а також сторінки профілю і лабораторій.

# ВИСНОВКИ

В результаті виконаної роботи було розроблено систему, яка надає можливість шукати та переглядати теми дипломних робіт, подавати і відміняти заявки на дипломні роботи, переглядати інформацію про свою дипломну роботу після прийняття заявки викладачем, графік виконання дипломної роботи, інформацію про склад та напрямки лабораторії, навантаження на викладачів.

Проаналізовано існуючі системи управління проектами, визначенно їх переваги та недоліки. Проаналізовано вимоги до нової системи управління дипломними проектами студента.

Базу даних розроблений за допомогою системи керування базами даних MySQL. Її будова відповідає функціям та вимогам розроблюваної системи. З використанням бібліотеки React.js та мови програмування JavaScript реалізовано графічний веб-застосунок для управління дипромними проектами. Для зв’язку з базою даних та взаємодії з клієнтами за допомогою API було створено Node.js сервер.

Завдяки тому, що програмна система написана за допомогою сучасної мови програмування та її провідних бібліотек, гарантується з часом підтримка всіх компонентів та їх подальша модернізація.

Дана система значно спрощує процес управління дипломними проектами. Реалізовані можливості програмного продукту повністю задовільняють поставленій задачі.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. SOAP and XML-RPC API Deprecation Notice [Електронний ресурс] режим доступу https://developer.atlassian.com/server/jira/platform/soap-and-xml-rpc-api-deprecation-notice/ (дата звернення 16.10.2021р). – Назва з екрана
2. Choosing a default language [Електронний ресурс] режим доступу https://confluence.atlassian.com/adminjiraserver071/choosing-a-default-language-802592304.html (дата звернення 17.10.2021р). – Назва з екрана
3. Йон Снейдер. Эффективное программирование TCP/IP /– ДМК Пресс, 2009 — 320 с.
4. OCLC [Електронний ресурс] режим доступу до ресурсу: https:// www.oclc.org/en/home.html?redirect=true (дата звернення 18.10.2021р). – Назва з екрана
5. Дэвид Флэнаган. JavaScript. Подробное руководство, 6-е издание, 2012, 1080с.
6. Выразительный Javascript [Электронный ресурс] режим доступа: https:// karmazzin.gitbooks.io/eloquentjavascript_ru/content/ (дата звернення 19.10.2021р) – Назва з екрана
7. Резиг Джон, Бибо Беэр. Секреты javascript ниндзя. — М.: «Вильямс», 2013. — 416 с.
8. Марина Дмитриева. JavaScript. — С-П.: БХВ-Петербург, 2004. — 336 с.
9. Современные возможности ES-2015 [Электронный ресурс] режим доступа: https://learn.javascript.ru/es-moder (дата звернення 19.10.2021р) – Назва з екрана
10. Banks A., Porcello E. Learning React: Functional Web Development with React and Redux. — O'Reilly Media, 2017. — 350 p.
11. Sidelnikov G. React.js Book: Learning React JavaScript Library From Scratch. — River Tigris LLC, 2016. — 350 p.

12. Bertolli M. React Design Patterns and Best Practices: Build easy to scale modular applications using the most powerful components and design patterns. — Packt Publishing, 2017. — 320 p.

13. Carlos R. React Cookbook: Create dynamic web apps with React using Redux, Webpack, Node.js, and GraphQL. — Packt Publishing, 2018. — 580 p.

14. Gorgon Z. React Explained: Your Step-by-Step Guide to React. – Amazon Digital Services LLC, 2019. — 305 p.

15. Wieruch R. The Road to learn React: Your journey to master plain yet pragmatic React.js. — Amazon Digital Services LLC, 2017. — 202 p.

16. Сэм Руби. Гибкая разработка веб-приложений 2014. — 448 с. — (Для профессионалов).

17. The MIT License (MIT) [Електронний ресурс] режим доступу: https://opensource.org/licenses/MIT (дата звернення 5.11.2021р) – Назва з екрана

18. Стив Макконнелл. Совершенный код = Code complete. — СПб.: Питер, 2005. — С. 896.

19. Большая книга CSS3 [Электронный ресурс] режим доступа:ftp:// ftp.micronet-rostov.ru/linux-support/books/programming/HTML-CSS/Дэвид Сойер Макфарланд — Большая книга CSS3.pdf. (дата зверенення 6.11.2021р) – Назва з екрана.

20. Коннолли Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли, К. Бегг. — М. : Вильямс, 2003. — 1440 с.

21. Браун И. Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript / И. Браун — Санкт-Петербург: Питер, 2017. — 336 с

22. Pereira C. Node.js Building APIs with Node.js. — Apress, 2016. — 136 p.

Додаток А

# Текст програми

```
// App.js
/* REACT */
import React from 'react'

/* MODULES */
import { Switch, Router, Route, Redirect } from 'react-router-dom'
import { connect } from 'react-redux'
import { ThemeProvider } from 'styled-components'

/* CUSTOM MODULES */
import RouteWithSubRoutes from '/src/routes/routeWithSubRoutes'
import { Layout } from '/src/components/HOC'
import history from './utils/history'
import { AuthService } from '/src/utils'
import { Loader } from '/src/components/UI'
import GlobalLoader from '/src/components/GlobalLoader'
import GlobalStyle from '/src/styles/global'
import { NotificationProvider } from './components/HOC'
import AuthPage from './containers/Auth'
import ForgotPasswordPage from '/src/containers/Auth/ForgotPassword'
import { routesSelector } from '/src/store/selectors/authSelectors'

/* ACTIONS */
import { loginAction, tryAutoLogin, loadedApp } from './store/actions/
authActions'
import { referenceAction } from './store/actions/referenceActions'

/* STYLES */
import { getTheme } from './styles/theme'

@connect(
```

```
    state => ({
      auth: state.auth,
      settings: state.settings,
      ROUTES: routesSelector(state)
    }),
    { loginAction, tryAutoLogin, loadedApp, referenceAction }
  )
  class App extends React.Component {
    componentDidMount () {
      const { tryAutoLogin, referenceAction, loadedApp } = this.props

      if (AuthService.getCredentials()) {
        tryAutoLogin(() => { referenceAction() })
      } else {
        loadedApp()
      }
    }

    render () {
      const { auth, settings, ROUTES } = this.props
      const theme = getTheme(settings.theme)
      // Can use different switches or private route and login route for protect
routes
      let routes = (
        <Switch>
          {(
            ROUTES && ROUTES.map(route => (
              <RouteWithSubRoutes
                key={route.path}
                {...route}
              />
            ))
          )}
          <Redirect to='/' />
        </Switch>)

      if (!auth.token) {
        routes = (<Switch>
```

```
        let content = (
          <Layout>
            {routes}
          </Layout>
        )if (!auth.isLoaded) content = (<Loader
        size='3rem'>Авторизація...</Loader>)

        return (
      <Router history={history}>
            <ThemeProvider theme={theme}>
       <NotificationProvider>
              <GlobalStyle />
              {content}
                                  {auth.isLoaded   &&   settings.isGlobalLoading   &&
<GlobalLoader>Завантаження...</GlobalLoader>}
              </NotificationProvider>
            </ThemeProvider>
          </Router>
        )
      }
    }
    export default App

    // Layout/index.js
    import React, { Component, Fragment } from 'react'
    import { connect } from 'react-redux'

    import Header from 'src/components/Header'
    import SideDrawer from 'src/components/Navigation/SideDrawer'

    import history from
    '/src/utils/history' import {
    MainContainer } from './style'
    import { settingsAction, toggleSideDrawerAction } from
'/src/store/actions/ settingsActions'
    import { isMobileSelector } from
    '/src/store/selectors/settingsSelectors' import {
```

```
        navigationItemsSelector } from '/src/store/selectors/authSelectors'

        @connect
          (
          state
          => ({
            settings:
            state.settings,
            auth: state.auth,
            isMobile: isMobileSelector(state),
            navigationItems:
            navigationItemsSelector(state)
          }),
          { settingsAction,
          toggleSideDrawerAction }, null,
          { pure: false }
        )
        class Layout extends Component {
          componentWillMount () {
            const clientWidth = document.documentElement.clientWidth
             this.props.settingsAction({ clientWidth, isOpenSideDrawer: clientWidth
             >=1000 })
}

componentDidMount () {
  window.addEventListener('resize', this.updateViewState) history.listen((location)
  => {
      if (location.pathname === '/login' || location.pathname === '/forgot-
      password') {

 if (this.props.auth.token) { history.go(1)
                }
              } else if (!this.props.auth.token) {
                history.push('/login')
              }
              this.props.settingsAction({ isShowAnimation: false })
```

```
    })
  }

  componentWillUnmount () {
    window.removeEventListener('resize',
    this.updateViewState)
  }

  updateViewState = () => {
      this.props.settingsAction({clientWidth:
document.documentElement.clientWidth })
  }

  render
    () {
    cons
    t {
      settings: { isOpenSideDrawer, isShowAnimation, isMobile },
      navigationItems,
      auth: { token },toggleSideDrawerAction, children
    } = this.props
    const isLoginPage = history.location.pathname === '/login' || !token

    return (
      <Fragment>
        {!isLoginPage && (
          <Fragment>
            <Header isMobile={isMobile} />
            <SideDrawer
              isOpen={isOpenSideDrawer}
              isShowAnimation={isShowAnima
              tion}
              toggleSideDrawer={toggleSideDrawerAction}
              navigationItems={navigationItems}
            />
          </Fragment>
```

```
            )}
            <MainContainer
              isLoginPage={isLoginPage}
              isOpenSideDrawer={isOpenSideDra
              wer}
              isShowAnimation={isShowAnimatio
              n}
            >
              {children}
            </MainContainer>
          </Fragment>
        )
      }
    }
    export default Layout

    // commonSaga.js
    import { call, put, takeEvery } from 'redux-saga/effects'

    import sagasConfig from './config'
    import { ApiCaller, AuthService } from '/src/utils'

    const successStatuses = [200, 201, 204]

    function * processRequest ({
      type, payload, onSuccess, onFail
    }) {
const { path, method,requiresAuth,successType,failType, multipart
      } =
      sagasConfig[type]
      let token = null
      let role = null
      if (requiresAuth) {
        ({ token, role } = yield call(AuthService.getCredentials))
      }
```

```
  const url = yield call(path,
  payload) const response = yield
  call(
    ApiCaller, url, method, payload, token, role, multipart
  )

  if
    (successStatuses.includes(response.status)
    ) { if (response.status === 204) {
      yield put({ type: successType, payload })
    } else {
      yield put({ type: successType, payload: response.data })
    }
    if (typeof onSuccess ===
      'function') { yield
      call(onSuccess, response.data)
    }
  } else {
    if (response.status === 401) {
      AuthService.unsetCredentials()
      window.location.reload()
    }
    yield put({ type: failType, payload:
    response.data }) if (typeof onFail ===
    'function') {
      yield call(onFail, response.data)
    }
  }
}

export const commonSaga = function *
  () { yield takeEvery(
    ({ type }) => /(.*)_REQUEST/.test(type) &&
      !sagasConfig.ignoreTypes.includes(type),
    processRequest
```

```
  )
}


export default Modal

import { useState } from 'react';
import styles from './style.module.css';
import cn from 'classnames';

const Paginator = ({ setPage, totalPageCount, page, portionSize = 10 }) => {
  const [pageSize, setPageSize] = useState(10);

  let countPages = Math.ceil(totalPageCount / pageSize);
  let pages = [];
  for (let i = 1; i <= countPages; i++) {
    pages.push(i);
  }

  let portionCount = Math.ceil(countPages / portionSize);
  let [portionNumber, setPortionNumber] = useState(1);
  let leftPortionNumber = (portionNumber - 1) * portionSize + 1;
  let rightPortionNumber = portionNumber * portionSize;

  return (
    <div className={styles.paginator}>
      {portionNumber > 1 && (
        <button
          className={styles.btn}
          onClick={() => setPortionNumber(portionNumber - 1)}
        >
          &#x3C;
        </button>
      )}
      {pages
        .filter((p) => p >= leftPortionNumber && p <= rightPortionNumber)
        .map((p) => {
          return (
            <span
              className={cn(
                {
```

```
                  [styles.selectedPage]: page === p,
                },
                styles.pageNumber
              )}
              key={p}
              onClick={() => {
                setPage(p);
              }}
            >
              {p}
            </span>
          );
        })}
      {portionCount > portionNumber && (
        <button
          className={styles.btn}
          onClick={() => setPortionNumber(portionNumber + 1)}
        >
          &#x3E;
        </button>
      )}
    </div>
  );
};

export default Modal;

import { useEffect, useState } from 'react';
import { NavLink } from 'react-router-dom';
import styles from './style.module.css';
import duration from '../../assets/images/duration.svg';
import dollar from '../../assets/images/dollar.svg';
import calendar from '../../assets/images/calendar.svg';
import playIcon from '../../assets/images/play.svg';
import Preloader from '../../common/Preloader';
import noImage from '../../assets/images/no-image.png';
import { API_KEY, baseURL } from '../../common/api/api';
import { calcTime, convertMoney } from '../../common/helpers/helpers';

const CurrentStudent = (props) => {
  useEffect(() => {
```

```
    let cleanupFunction = false;
    const fetchData = async () => {
      setIsLoad(true);
      const response = await fetch(
        `${baseURL} /${props.match.params.id}?api_key=${API_KEY}`
      ).then((res) => res.json());
      if (!cleanupFunction) setFilm(response);
      setIsLoad(false);
    };
    const fetch = async () => {
           ).then((res) => res.json());
      Object.keys(response.results).forEach((key) => {
        if (!cleanupFunction) setVideo(response.results[key]);
      });
    };
    fetchData();
    fetch ();
    return () => (cleanupFunction = true);
  }, [props.match.params.id]);

  if (isLoad) return <Preloader />;
  return (
    <div className={styles.container}>
      <div className={styles.nav}>
        <NavLink to="/">Home</NavLink>
      </div>
      <div
        className={styles.item}
        <div className={styles.wrapper}>
          <div>
            <div className={styles.container }>
              <div className={styles.top }>
                <a
                  target="_blank"
                  rel='noreferrer'
                >
                  <img className={styles.playIcon} src={playIcon} alt="play"
/>
                </a>
              </div>
                        </div>
              ) : (
```

```
              <img src={noImage} alt="without poster" />
            )}
          </div>
          <div className={styles.info}>
            <h2 className={styles.title}>{title}</h2>
            <p className={styles.desc}>{overview}</p>
            <div className={styles.rate}>
              <span>
               Show
              </span>
              <span className={styles.rate}> {vote_average}</span>
            </div>
          </div>
        </div>
        <div className={styles.studentInfo}>
          <div className={styles.time}>
            <img className={styles.duration} src={duration} alt="duration" />
            <span>Running time: {calcTime(film.runtime)}</span>
          </div>
          <div className={styles.wrao}>
            <img className={styles.mag} />
            {(
              <span>Budget: no information</span>
            )}
          </div>
          <div className={styles.date}>
            <img className={styles.calendar} src={calendar} alt="calendar" />
            <span>Date : {student. _date}</span>
          </div>
        </div>
        <Stundents studentID={props.match.params.id} />
      </div>
    );
};

export default CurrentStudent;
```