Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Створення веб-сайту ”Дитячий лікар, м. Кременець” засобами React та Node.js

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Терновий О.В.
(підпис) (прізвище та ініціали)

Керівник _____ Дуда О.М.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Литвиненко Я.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Голотенко О.С.
(підпис) (прізвище та ініціали)

Тернопіль
2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
« 19 » червня 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Терновому Олегу Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Створення веб-сайту "Дитячий лікар, м. Кременець" засобами React та NODE.JS

Керівник роботи Дуда Олексій Михайлович, к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 07 » лютого 2023 року № 4/7-133
2. Термін подання студентом завершеної роботи 20 червня 2023р.
3. Вихідні дані до роботи Дані про веб-сайти приватних медичних закладів. Навчальні та наукові публікації щодо прогресивних веб-технологій.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Постановка задачі створення веб-сайту «дитячий лікар» м. Кременець. 1.1. Аналіз предметної області. 1.2. Формування вимог до веб-сайту «Дитячий лікар» м. Кременець. 1.3. Варіанти використання веб-сайту «Дитячий лікар» м. Кременець. 1.4. Програмно-алгоритмічна архітектура веб-сайту «Дитячий лікар» м. Кременець. 1.5. Підсистема зберігання даних веб-сайту «Дитячий лікар» м. Кременець. 2. Проектування та реалізація веб-сайту «дитячий лікар» м. Кременець. 2.1. Моделювання архітектури веб-сайту «Дитячий лікар» м. Кременець. 2.2. Структурна модель веб-сайту «Дитячий лікар» м. Кременець. 2.3. Проектування UML-діаграм класів веб-сайту «Дитячий лікар» м. Кременець. 2.4 Структура каталогів веб-сайту «Дитячий лікар» м. Кременець. 2.5 Програмна реалізація веб-сайту «Дитячий лікар» м. Кременець. 2.6 Тестування «React»-компонентів веб-сайту «Дитячий лікар» м. Кременець. 2.7 Висновок до другого розділу. 3. Безпека життєдіяльності, основи охорони праці. 3.1 Долікарська допомога при ураженні електричним струмом. 3.2 Естетичне оформлення робочого місця працівника в ІТ. Висновки. Перелік джерел. Додатки
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1.Титульна сторінка. 2. Тема та мета роботи. 3. Завдання роботи. 4. Актуальність теми. 5. UML-діаграма актора «Пацієнт» 6. UML-діаграма входу … 7. UML-діаграма актора «Адміністратор» 8. Технології розробки. 9. Архітектура веб-сайту … 10. Структура MVC … 11. Діаграма класів MVC … 12. Програмна реалізація. 13. Інтерфейс веб-сайту … 14. Інтерфейс веб-сайту … 15. Висновки. 16. Доповідь завершено.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці		05.06.2023	08.06.2023

7. Дата видачі завдання 23 січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	23.01.2023	*Виконано*
2.	Підбір джерел про підсистеми зберігання даних та технологій написання (фреймворків)	24.01.2023-26.01.2023	*Виконано*
3.	Опрацювання джерел по темі кваліфікаційної роботи	27.01.2023-31.01.2023	*Виконано*
4.	Виконання дослідження щодо варіантів використання інформаційної системи веб-сайту та моделювання архітектури системи	01.02.2023-07.02.2023	*Виконано*
5.	Оформлення розділу “Постановка задачі створення веб-сайту «дитячий лікар» м. Кременець”	08.02.2023-15.03.2023	*Виконано*
6.	Оформлення розділу “Проектування та реалізація веб-сайту «дитячий лікар» м. Кременець”	16.03.2023-04.06.2023	*Виконано*
7.	Виконання завдання до підрозділу “Безпека життєдіяльності”	05.06.2023-06.06.2023	*Виконано*
8.	Виконання завдання до підрозділу “Основи охорони праці”	07.06.2023-08.06.2023	*Виконано*
9.	Оформлення кваліфікаційної роботи	09.06.2023-11.06.2023	*Виконано*
10.	Нормоконтроль	12.06.2023-13.06.2023	*Виконано*
11.	Перевірка на плагіат	14.06.2023	*Виконано*
12.	Попередній захист кваліфікаційної роботи	15.06.2023	*Виконано*
13.	Захист кваліфікаційної роботи	19.06.2023	

Студент _____ Терновий О. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дуда О. М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Створення веб-сайту «Дитячий лікар» м. Кременець засобами React та Node.js. // Кваліфікаційна робота освітнього рівня «Бакалавр» // Терновий Олег Володимирович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41 // Тернопіль, 2023 // С. 57, рис. – 10, табл. – 0, кресл. – 16, додат. – 8, бібліогр. – 34.

Ключові слова: react, бібліотека, веб-сайт, користувач, структура, фронтенд, шаблон.

Кваліфікаційна робота присвячена розробці веб-сайту для дитячої клініки «Дитячий лікар» м. Кременець з метою покращення досвіду пацієнтів та оптимізації адміністративних процесів. Робота складається з двох основних розділів, кожен з яких присвячений окремим аспектам процесу розробки веб-сайту.

В першому розділі кваліфікаційної роботи виконано аналіз предметної області, сформовано вимоги до функціоналу веб-сайту. Подано діаграми варіантів використання для різних способів взаємодій користувачів з інтерфейсом веб-сайту.

В другому розділі кваліфікаційної роботи описано процес розробки та реалізації програмних елементів веб-сайту «Дитячий лікар». Змодельовано інформаційно-технологічну архітектура веб-сайту. Розроблено структурну модель веб-сайту. Описано процес реалізації програмних елементів веб-сайту з використанням сучасних веб-технологій та бібліотек.

Результатом виконання кваліфікаційної роботи є практична реалізація веб-сайту, який підвищує зручність надання послуг юних пацієнтам дитячої клініки «Дитячий лікар» м. Кременець та їх батькам.

ANNOTATION

The «Children'S Doctor» Kremenets Town Website Development by Means of React and Node.Js // Qualification work of the educational level «Bachelor» // Ternovyi Oleh Volodymyrovych // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-41 // Ternopil, 2023 // P. 57, fig. – 10, tabl. – 0, chair. – 16, annexes. – 8, references – 34.

Keywords: frontend, library, react, structure, template, user, website.

The qualification work is dedicated to The «Children'S Doctor» Kremenets Town Website Development by Means of React and Node.Js in order to improve the patient experience and optimize administrative processes. The work consists of two main sections, each of which is devoted to separate aspects of the website development process.

The first section of the qualification paper considered the analysis of the subject matter, requirements for the website functionality are formed. Use case diagrams for different ways of user interaction with the website interface are presented.

In the second section of the qualification work, it is considered the process of developing and implementing program elements of the «Children'S Doctor» Kremenets Town website. The information technology architecture of the website is modeled. The structural model of the website is developed. The process of implementing the program elements of the website using modern web technologies and libraries is described.

The result of the qualification work is the practical implementation of the website, which increases the convenience of providing services to young patients of the children's clinic " Children'S Doctor " in Kremenets and their parents.

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

БД – База даних – це організована структура, призначена для зберігання, зміни й обробки взаємопов’язаної інформації.

ІКТ – Інформаційні та комунікаційні технології.

ІТ – Інформаційні технології.

ОС – Операційна система.

ТЗ – Технічне завдання.

CSS (англ. Cascading Style Sheets) – каскадні таблиці стилів.

HTML (англ. Hyper Text Markup Language) – мова розмітки гіпертекстових документів).

MVC (англ. Model View Controller) – цей шаблон використовується для розділення проблем програми.

SEO (англ. Search Engine Optimization) – це комплекс заходів щодо поліпшення сайту для його ранжування в пошукових системах.

SPA (англ. Single Page App) – веб-сайт, який взаємодіє з користувачем, динамічно переписуючи поточну веб-сторінку новими даними з веб-сервера.

UI (англ. User interface) – користувацький інтерфейс. Кінцевий результат роботи дизайнера, те, що побачить користувач.

UML (англ. Unified Modeling Language) – уніфікована мова моделювання.

ЗМІСТ

ВСТУП

Актуальність теми. Внаслідок COVID-19 та війни збільшився попит у сфері дистанційного надання послуг з використанням сучасних інформаційних технологій, завдяки цьому популярність веб-сайтів продовжує збільшуватися. Їх можна використовувати у сфері малого бізнесу, щоб надавати якісні послуги. Водночас великі компанії у різних галузях господарської діяльності потребують наявності власних веб-сайтів для збільшення аудиторії користувачів та зміцнення ринкових позицій. Не зважаючи на обширний перелік доступних онлайн конструкторів, створення якісного веб-сайту потребує фахового підходу та знань [1]. На даний час з`являється обширний перелік фреймворків, які дозволяють суттєво спростити роботу досвідчених веб-розробників та допомогти новачкам у цій сфері зробити перші веб-проекти. Завдяки цьому можна розробити якісні інтерфейси веб-застосунків та їх структури. Навички розробки веб-застосунків будуть корисні працівникам суміжних галузей [2]. Тому розробка веб-сайтів та веб-застосунків є актуальним напрямком сучасних досліджень.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є підвищення якості обслуговування клієнтів медичної клініки «Дитячий лікар» м. Кременець. Для досягнення поставленої мети необхідно виконати задачі:

- Провести аналіз предметної області.
- Виконати проектування структури веб-сайту.
- Сформувати моделі даних.
- Розробити серверну та клієнтські частину веб-сайту.
- Провести тестування та виправлення помилок.
- Перенести та розгорнути усі частини веб-сайту на хостинг.

Практичне значення одержаних результатів. Практично розроблено, протестовано та розміщено на хостинг веб-сайт «Дитячий лікар», що функціонує в м. Кременець.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ СТВОРЕННЯ ВЕБ-САЙТУ «ДИТЯЧИЙ ЛІКАР» М. КРЕМЕНЕЦЬ

1.1 Аналіз предметної області

Для розробки веб-інтерфейсів зазвичай використовуються «HTML», «CSS», «JavaScript», «PhP» або сформовані на їх основі фреймворки. В залежності від ТЗ та рівня досвіду працівника, який займається безпосереднім написанням коду, можуть використовуватися різні програмні засоби. На даний час веб-розробники регулярно реалізують веб-застосунки у виді «SPA». На основі точної програмно-алгоритмічної реалізації завантажується лише один веб-документ. Він оновлює вміст усього HTML-документа за допомогою «JavaScript API». Це дозволяє використовувати веб-сайти без зайвого завантаження цілих нових веб-сторінок із сервера. Як наслідок відбувається підвищення продуктивності та динамічний обмін даними. Проте такому способу притаманні ряд недоліків, зокрема складнощі із «SEO», більше зусиль для підтримки стану, навігації, моніторингу [3].

Перед тим як перейти до програмної реалізації, потрібно провести попередній аналіз, та визначити мету створення веб-сайту. В перспективі це допоможе уникнути непорозумінь з замовником та продовжити розробку орієнтуючись на його потреби.

Доречним буде створення структури сторінок, формування процесів взаємодії між ними, реалізацію функціональних можливостей кожної окремої сутності. Важливим етапом є вибір шаблону відображення для максимального задоволення потреб клієнтів та оптимізації швидкості функціонування сервісів та веб-сайту в цілому [4].

Дизайн є важливим елементом роботи сучасного онлайн-сервісу. Він потребує аналізу потреб користувачів на яких буде зорієнтований онлайн-сервіс. Потрібно відповідально підійти до вибору кольорової гами, яка буде притаманна усьому веб-інтерфейсу. В реалізації веб-інтерфейсу повинна бути

ефективно організована ілюстративна, функціональна та декоративна графіка. Це допоможе розробнику витрачати менше часу на проектування дрібних інтерфейсу. В переважній більшості проектування дизайну відбувається засобами онлайн-сервісу «Figma», «Software». При цьому зручно відображати інтерфейс проекту, функціональні зв'язки між сторінками чи відображення інформації. Водночас використовуються інші графічні редактори, зокрема «Adobe Photoshop» тощо [5].

Після проектування інтерфейсу розробник може приступити до його програмної реалізації та створення серверної частини. При цьому потрібно враховувати особливості веб-технологій. Оскільки від них залежить швидкодія інтерфейсу та програмної частини. Це безпосередньо впливає на взаємодію з клієнтами. Швидкість завантаження елементів чи перехід на іншу сторінку впливає на враження користувачів від використання цих веб-сайтів та онлайн-сервісів [6].

Після завершення етапів проектування та розробки потрібно провести перевірку коректності роботи візуальних складових інтерфейсів, зокрема граматичні помилки, шрифт, відображення елемента відносно макету. Також потрібно провести тестування функціональних елементів відповідно до вимог замовників.

З проведеного аналізу можна зробити висновок, що створення якісного веб-інтерфейсу потребує суттєвих не тільки грошових та часових затрат.

1.2 Формування вимог до веб-сайту «Дитячий лікар» м. Кременець

В процесі дипломного проектування необхідно не тільки розробити веб-сайт «Дитячий лікар» м. Кременець, який буде висвітлювати інформацію про послуги клініки та створити онлайн-систему, яка буде ефективно управляти процесами обслуговування клієнтів.

Тому сформуємо вимоги до функціональних можливостей веб-сайту «Дитячий лікар» м. Кременець.

При цьому повинні бути реалізовані функціональні можливості:

– Ефективний багатокроковий алгоритм самостійної реєстрація пацієнтів.

– Онлайн-бронювання зустрічей. Пацієнти можуть записатися онлайн на фізичний прийом до лікаря. При цьому вони можуть самі обрати лікаря та зручний час прийому, який відповідає їхньому особистому графіку. Це допоможе суттєво скоротити час очікування та підвищити ефективність роботи клініки.

– Редагування вмісту сторінки адміністраторами.

– Перегляд адміністраторами даних про клієнтів, статистики використання онлайн-сервісів.

– Для ролі користувача «Лікар» створити список пацієнтів, які записані на прийом.

– Автоматичне надсилання повідомлень засобами Telegram, про новий запис на прийом.

– Оперативний облік пацієнтів та працівників.

– Оперативний облік робочого часу персоналу клініки.

В процесі розробки веб-сайту «Дитячий лікар» м. Кременець. перелік функціональних можливостей може бути змінений, зокрема доповнений.

Структура веб-сайту «Дитячий лікар» м. Кременець повинна бути реалізована на основі шаблону проектування «MVC».

Враховуючи подані особливості роботи веб-сайту «Дитячий лікар» м. Кременець сформуємо перелік безпекових вимог:

– Усі дані користувача в процесі реєстрації, аунтентифікації або виконання запису на прийом повинні перевірятися за допомогою регулярних виразів.

– В БД веб-сайту повинна зберігатися хешована інформація щодо логіну та паролю.

Враховуючи опис до функціональних можливостей веб-сайту сформуємо вимоги щодо організації доступу до нього:

– Доступ користувачів до інтерфейсу веб-сайту «Дитячий лікар» м. Кременець повинен здійснюватися засобами програм інтернет-браузерів для ПК та смартфонів під управлінням ОС Android та IOS.

– Онлайн-доступ повинен бути реалізований на основі стандартного протоколу HTTPS з використанням SSL сертифікатів.

Інтерфейс веб-сайту повинен коректно відображати спродуковані веб-сторінки на різних версіях найпоширеніших браузерів, зокрема Chrome, Opera, Safari та Edge найновіших версій.

1.3 Варіанти використання веб-сайту «Дитячий лікар» м. Кременець

Веб-сайт «Дитячий лікар» м. Кременець, повинен містити інтегровані функціональні набори адміністратора та користувачів. На основі проаналізованого опису предметної області сформуємо ієрархічну діаграму спадкування прав на привілеї користувачів (див. рисунок 1.1).

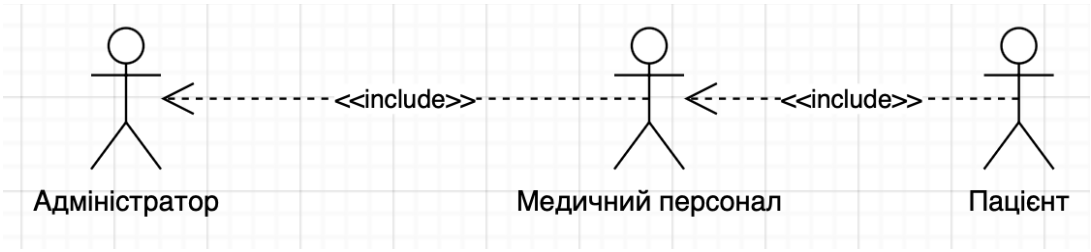


Рисунок 1.1 – UML-діаграма варіантів використання

Веб-сайт «Дитячий лікар» м. Кременець повинен ефективно реалізовувати набори структури для:

– ознайомлення клієнтів інформацією про клініку;

– надання інформації про графік роботи;

– перелік послуг;

– допомогти швидко та без зайвих дій отримати інформацію про можливості консультування в лікарів;

– вибір лікаря;

– бронювання вакцини з переліку можливих.

Сформуємо діаграми використання для кожного автора. На рисунку 1.2 зображена діаграма використання актора «Пацієнт» веб-сайту «Дитячий лікар» м. Кременець.

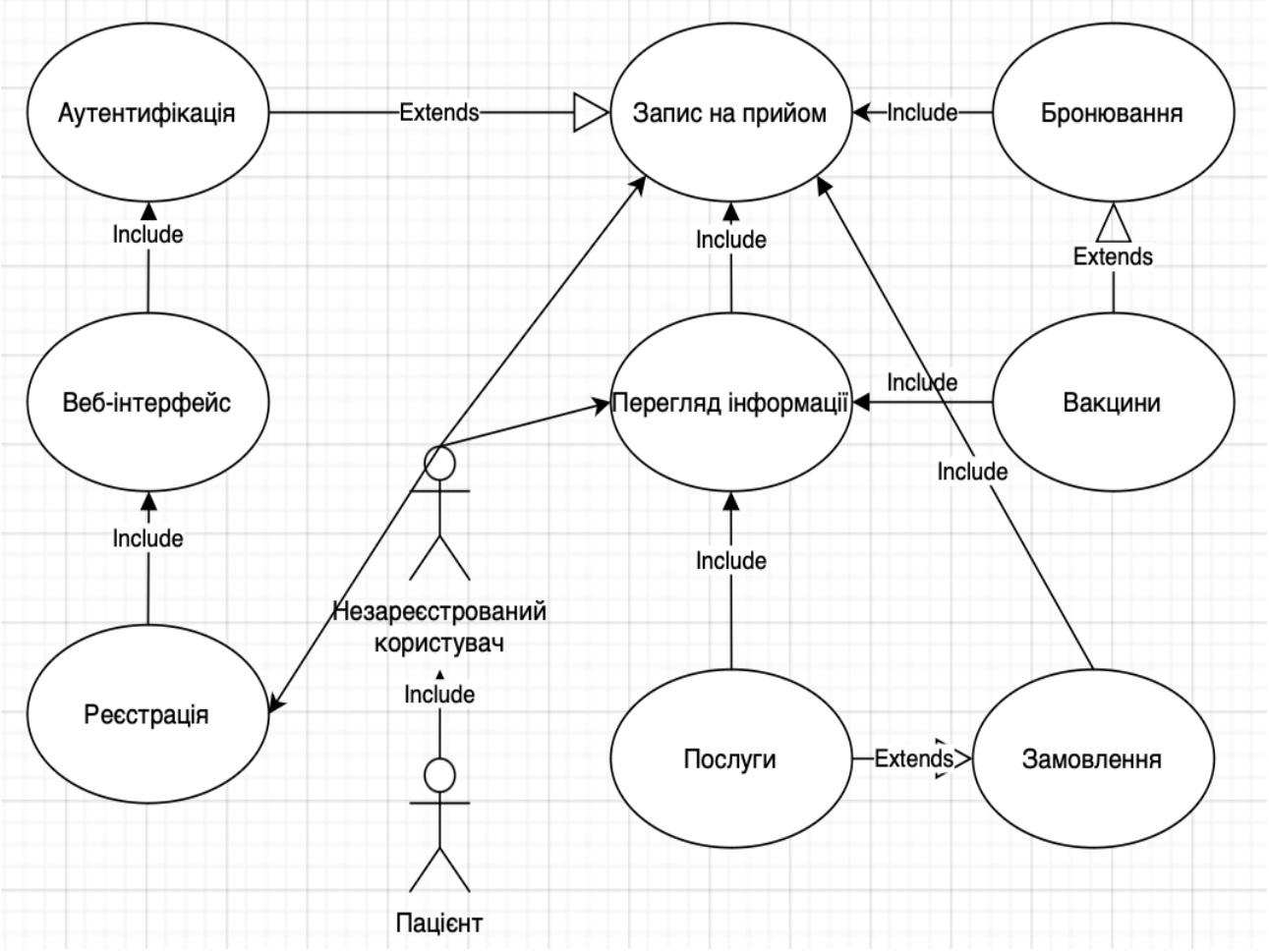


Рисунок 1.2 – UML-діаграма «Uses Case» актора «Пацієнт»

«Пацієнт» може перейти з головної сторінки до реєстраційної або аутентифікаційної. Після вводу даних, зокрема логіну та паролю відбувається перевірка даних форми та вивід повідомлення про успішний вхід або помилку через некоректні дані. В цей момент дані користувача надіслані до сервера [7]. Його буде скеровано на сторінку входу. UML-діаграма «Uses Case» зареєстрованого пацієнта подана на рисунку 1.3.

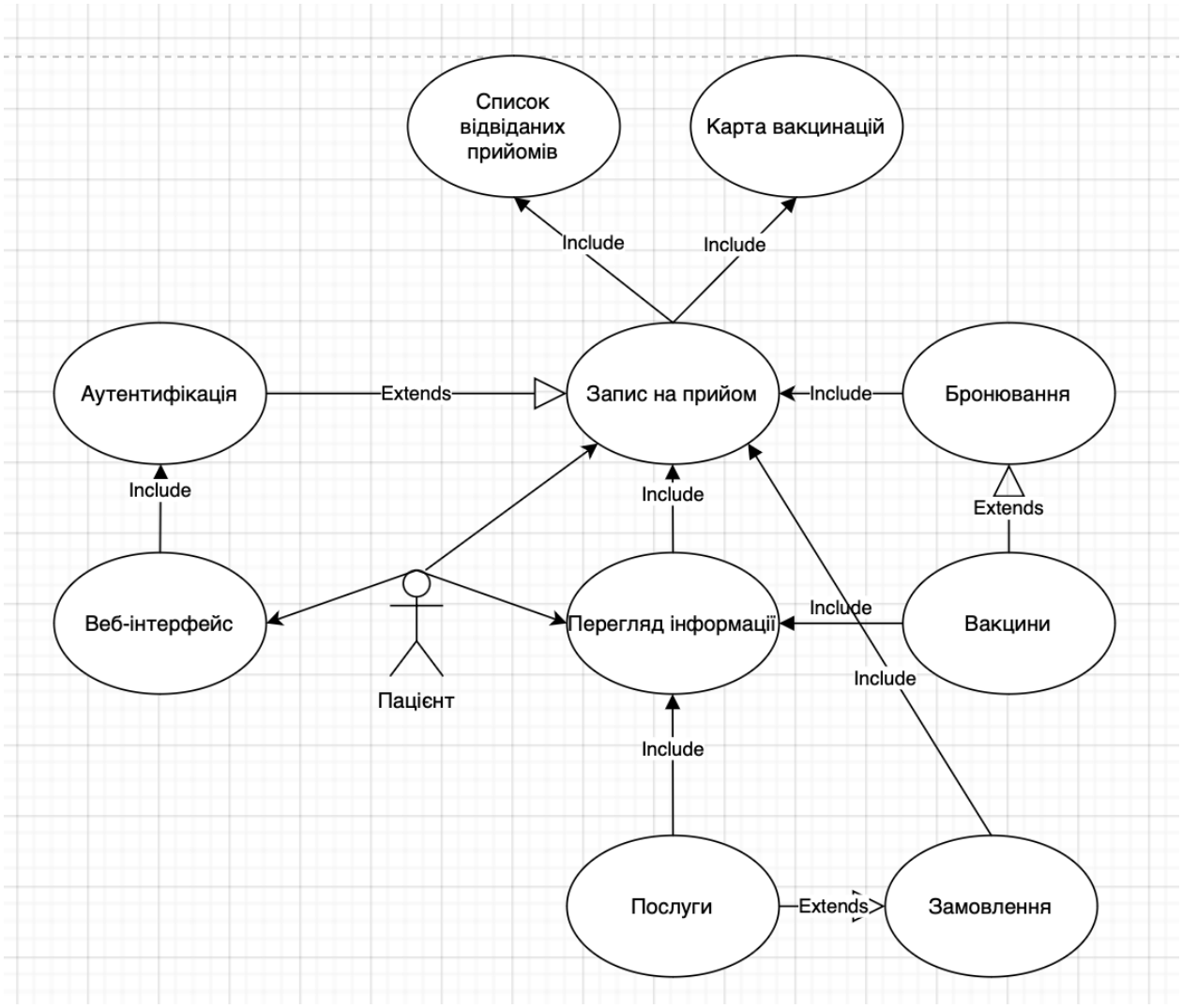


Рисунок 1.3 – UML-діаграма «Uses Case» входу користувача

Зазначена діаграма сформована на основі UML-діаграми «Uses Case» користувача. На головній сторінці веб-інтерфейсу потрібно перейти до аутентифікаційної форми для перевірки на коректності введених даних. Потім доступні два варіанти подій – помилка або повідомлення про успішний вхід та перенаправлення на стартову сторінку. Водночас дані користувача зберігаються в локальному сховищі браузера, щоб при переході на іншу сторінку або її оновленні не втрачатись [8]. Для актора стають доступними функції замовлення послуг, бронювання вакцин, списку відвіданих прийомів та карта вакцин.

Для актора адміністратора доступні інші набори функціональних можливостей подані на рисунку 1.4.

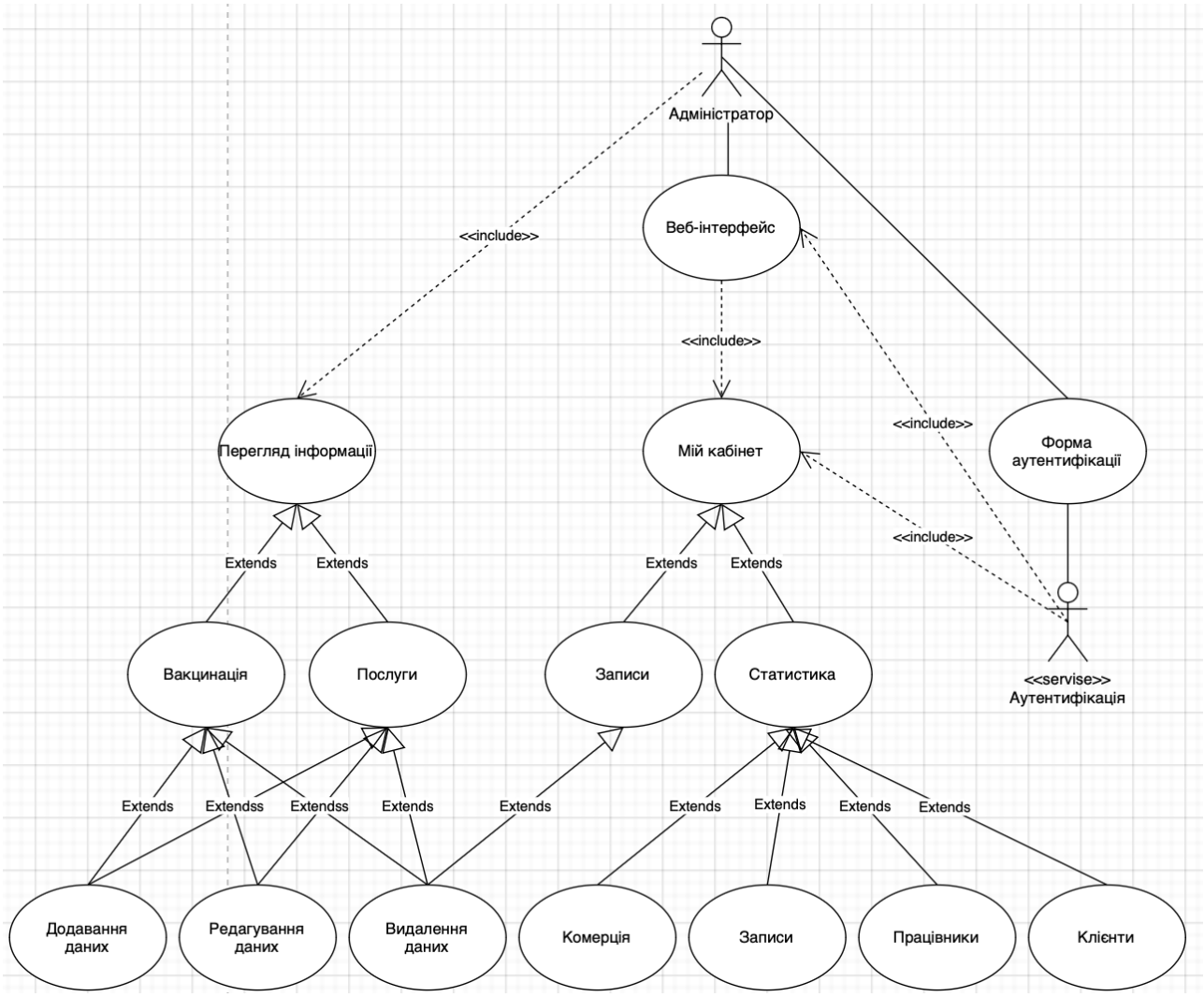


Рисунок 1.4 – Діаграма «Uses Case» актора «Адміністратора»

Ця діаграма є продовженням діаграми актора «Зареєстрований користувач». При вході актор «Адміністратор» вводить реєстраційні дані та отримує доступ до процедур редагування, додавання та видалення даних на веб-сторінках «Вакцинація» та «Послуги». Веб-сторінка «Записи» доступна лише для акторів «адміністратор» та «Лікар». Вони можуть змінювати статус акторів «Пацієнт». Після цього відбувається видалення відповідних записів з БД.

Користувач веб-сайту із рівнем доступу «Лікар», повинен мати функціонал переглядати список пацієнтів, які записані до нього на прийом не залежно від дати. А також позначати їх як «Оглянув» для добавлення їх в іншу БД, з якої вже беруться дані для сторінки адміністратора.

1.4 Програмно-алгоритмічна архітектура веб-сайту «Дитячий лікар» м. Кременець

Для спрощення процесів розробки та підвищення продуктивності веб-сайту «Дитячий лікар» м. Кременець доцільно використати бібліотеки та фреймворки.

Фреймворк, готова модель програмно-алгоритмічного комплексу для прискорення процесу розробки. На її основі можна сформувати власний програмний код. Він дозволяє задати структуру, визначити правила та містить необхідний набір інструментів для створення веб-застосунку [9]. Зазвичай фреймворки використовуються саме у веб-розробці. Вони забезпечують:

– стабільність функціонування програмних засобів;
– ефективну роботу з БД та файловою системою;
– обробку виключень та помилок;
– захист.

Фреймворки ефективні для створення складних веб-застосунків та простих веб-сайтів [10].

Фреймворки дозволяють зменшити імовірність виникнення помилок. Вони працюють швидше та ефективніше при більшому навантаженні, ніж веб-застосунки, що написані безпосередньо на «JavsScript» або «PhP». Вмонтовані засоби аутентифікації надають можливості заблокувати доступ до базових функцій фреймворку. Вибір фрейворку для розробки веб-сайту «Дитячий лікар» м. Кременець проводитимемо між «Angular», «React» та «Vue js».

Порівнюючи «Angular» та «React», відзначимо наявність «всього в комплекті» «batteries included». В «Angular» є все необхідне для розробки складних веб-застосунків. Зокрема маршрутизація, форми, валідація, інтернаціоналізація тощо. А «React» складається з бібліотек, які можна додавати по мірі необхідності. Це надає більшу гнучкість, але і потребує більшої уваги при налаштуванні та інтеграції [11].

«Vue.js» – то фреймворк , що знаходиться між «Angular» та «React». Він має гнучкішу архітектуру, ніж «Angular», але не настільки модульний, як «React». «Vue.js» має зручний та простий в освоєнні синтаксис, який дозволяє швидко та ефективно створювати прототипи та невеликі веб-застосунки.

«Angular» та «React» зазвичай працюють швидше за «Vue.js» у більшості складніших веб-застосунках, оскільки вони використовують віртуальну DOM. Це створює менше затримок при оновленні веб-сторінки [12].

Проаналізувавши подані характеристики фреймворків зупинимо вибір на «React», оскільки він має доволі високу продуктивність, багатофункціональність та можливість повторного використання компонентів. Це дозволяє суттєво зменшити обсяги програмно-алгоритмічної розробки.

На даний час «React-Bootstrap» замінює «JavaScript Bootstrap». Кожен окремий компонент фреймворку був створений з нуля, без зайвих залежностей на кшталт «jQuery». Модель компонентів «React» дає більше контролю форми та функцій кожного компонента. Кожен окремий компонент реалізовано з урахуванням доступності. Результуючий набір доступних за замовчуванням компонентів, функціональніший від «Bootstrap» [13].

«Redux» – це бібліотека управління станами, яка забезпечує передбачуваний і централізований спосіб управління станом додатку. Вона базується на принципах «Flux» – шаблоном управління потоком даних у веб-застосунку. Основна ідея «Redux» у тому, щоб зберігати стан веб-застосунку в єдиному сховище, оновлювати його в передбачуваний і контрольований спосіб за допомогою дій і редукторів.

Загальний огляд процесу функціонування «Redux»:

1. Стан програми зберігається в єдиному сховищі. Цей об'єкт доступний лише для читання. Його можна змінити лише шляхом відправлення дій.

2. «Actions» – це звичайні JavaScript-об'єкти, які описують те, що відбувається в застосунку. Вони відправляються компонентами або іншими частинами програмного коду, щоб викликати зміни стану.

3. «Reducers» – це функції, які приймають вхідну інформацію про поточний стан і потрібну дію. Вони повертають новий стан та відповідають за його оновлення на основі відправленої дії.

4. «Сховище» «слухає» відправлені дії і передає їх редукторам для оновлення стану. Коли стан оновлюється, «сховище» ігнорує всіх зареєстрованих підписників, щоб вони могли повторно спродукувати інтерфейс на основі оновлених даних.

«Redux» надає проміжне програмне забезпечення, яке дозволяє розширити функціональність веб-інтерфейсів. Проміжне ПЗ може перехоплювати відправлені дії та виконувати додаткове опрацювання перед їх передачею редукторам. Це може бути використано для ведення журналів, обробки помилок та виключення асинхронних операцій [14].

1.5 Підсистема зберігання даних веб-сайту «Дитячий лікар» м. Кременець

Сучасні веб-сайти та веб-застосунки зазвичай використовують реляційні або не реляційні БД. Оскільки веб-сайт «Дитячий лікар» м. Кременець, призначений для зберігання різнотипових інформаційних сутностей, то використаємо не реляційну БД.

Одна з головних переваг «NoSQL» баз даних – це їх масштабованість. Вони мають розподілену архітектуру, тому легко масштабуються горизонтально, або на множині серверів. БД «NoSQL» можуть автоматично розподіляти дані на різних серверах. Це суттєво підвищує швидкість читання даних у розподіленому середовищі [15].

Вибиратимемо БД між «Couchbase» та «MongoDB». Обидві СКБД мають ряд переваг та недоліків.

Зокрема переваги Couchbase:

– Гнучка масштабовність: Couchbase забезпечується за допомогою процедур кластеризації та шарування.

– Висока доступність: формує вбудовану можливість забезпечення доступності даних на основі реплікації та автоматичного відновлення після збоїв.

– Швидкий доступ до даних завдяки інтегрованому механізму кешування та підтримці різнотипових операцій з даними у режимі реального часу.

Недоліки «Couchbase»:

– Високі вимоги до мережевого обладнання, серверів з метою забезпечення оптимальної продуктивності.

– Висока вартість використання з комерційною метою [16].

Переваги «MongoDB»:

– Гнучка схема даних дозволяє зберігати дані різних типів та суттєво змінювати їх структуру у будь-який момент часу.

– Широкі можливості формування запитів для пошуку та фільтрації даних на основі «JSON»-нотації.

– Простота використання інтерфейсу дозволяє швидко почати роботу з СКБД.

Недоліки «MongoDB»:

– Недостатня підтримка транзакцій може призвести до втрати даних при збоях.

– Відсутній вбудований механізм кешування даних [14].

Отже зважаючи на подані переліки переваг та недоліків зупинимо вибір на «MongoDB». Вона гнучкіша та простіша у використанні.

1.6 Висновок до першого розділу

В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр» проаналізовано предметну область. Сформовано перелік вимог до веб-сайту. Розглянуто варіанти використання веб-сайту. Спроєктовано програмно-алгоритмічну архітектуру веб-сайту «Дитячий лікар» м. Кременець. Сформовано підсистему зберігання даних веб-сайту «Дитячий лікар».

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБ-САЙТУ «ДИТЯЧИЙ ЛІКАР» М. КРЕМЕНЕЦЬ

2.1 Моделювання архітектури веб-сайту «Дитячий лікар» м. Кременець

Для веб-сайту «Дитячий лікар» м. Кременець, оберемо трирівневу клієнт-серверну архітектуру подану на рисунку 2.1.

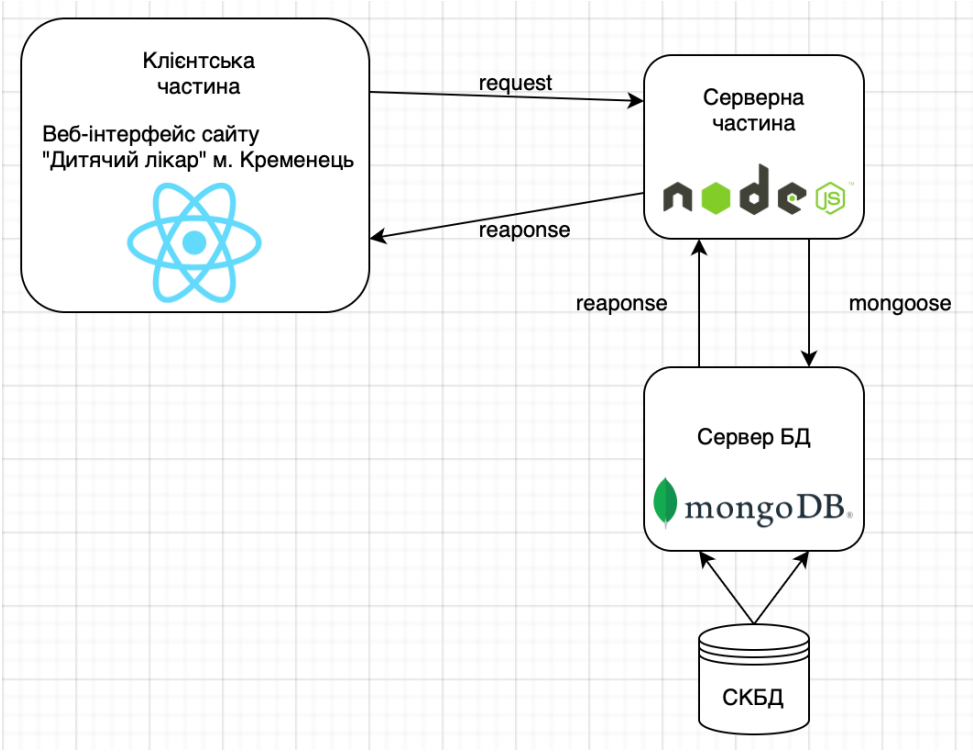


Рисунок 2.1 – Архітектура клієнт-сервер веб-сайту «Дитячий лікар» м. Кременець

У поданій архітектурі «Сервер», відповідальний за прийом запитів і відправку відповідей клієнтам, використовує виключно власні ресурси.

«Клієнт» є програмно-алгоритмічною реалізацією візуального представлення користувацького інтерфейсу.

«Сервер СКБД» надає серверній частині збережену в БД інформацію та відповідає за керування ресурсами.

Базовий принцип роботи поданої архітектури полягає в тому, що група серверів отримують запити від клієнтів, обробляють їх та, при потребі, звертаються до сервера БД. Такий підхід до розподілу операцій суттєво знижує навантаження на окремий сервер

Після надсилання «Клієнтом» запиту до «Сервера», він опрацьовує його та повертає «Клієнту» результат. «Сервер» може обслуговувати декілька «Клієнтів» одночасно. Якщо декілька запитів надходять одночасно, то вони розміщуються в черзі та виконуються «Сервером» послідовно. Іноді запити можуть мати різні пріоритети. Запити з вищим пріоритетом повинні оброблятися першими. Це називається «event loop», «microtasks» та «macrotasks» [17].

Цикл опрацювання отриманих запитів працює за простою концепцією. Існує неперервний цикл, у якому механізм «JavaScript» очікує на завдання, виконує їх, а потім переходить в сплячий режим очікування нових завдань. В цілому програмний механізм працює за алгоритмом, в якому поки є завдання вони виконуються починаючи з найпершого. В іншому випадку відбуваєтсья очекування виклику та перехід до його опрацювання.

Усі мікрозавдання мають вищий пріорітет до виконання будь-якої іншої процедури обробки подій або візуалізації будь-якого макрозавдання.

Такий програмно-алгоритмічний механізм є важливим, оскільки це ґарантує, що програмне середовище одинакове для всіх мікрозадач [16].

Реалізовані функцій «Сервера»:

– зберігання, доступ, редагування та видалення даних;
– обробка клієнтських запитів;
– надсилання відповідей «Клієнтам».

Реалізовані функції «Клієнта»:

– формування та надсилання запиту до «Сервера»;
– формування користувальницького інтерфейсу;

– надсилання запитів на оновлення, додавання або видалення даних та отримання їх результатів.

Функцій «Сервера СКБД»:

– під’єднання до «Сервера»;

– зберігання інформації;

– обробка інформації;

– передача інформації;

В цілому розглянута архітектура визначає взаємодію між комп’ютерними веб-системами. Для цього використовується стек мережевих протоколів. Він описує правила взаємодіють комп’ютерів у мережі. Основні мережеві протоколи веб-сайу «Дитячий лікар» м. Кременець – це «TCP/IP» та «HTTPs». Перший визначає правила передачі інформації від відправника до отримувача. Другий протокол – це протокол передачі гіпертексту. Він служить для передачі даних у будь-яких форматах. «HTTPs» безпечніший порівняно зі звичайним «HTTP» завдяки додаванню до базового протоколу функціоналу для шифрування даних [18].

Веб-сайт «Дитячий лікар» м. Кременець, сформований на клієнт-серверній взаємодії, містить три основних компоненти:

– представлення даних;

– прикладний компонент;

– компонент управління та зберігання ресурсів.

2.2 Структурна модель веб-сайту «Дитячий лікар» м. Кременець

Ефективно сформована структура забезпечує веб-сайту коректность його логічності структури та зручність використання. Якщо структура погано опрацьована то не буде зручної навігація та пошуку категорії.

Структура веб-сайту є ключовим технічним інструментом для «SEO». Некоректне формування структури веб-сайту суттєво впливає на його пошукову оптимізацію. Тому при розробці архітектури веб-сайту необхідно

однозначно зафіксувати розташування кожного розділу та підрозділу, щоб задовольнити інформаційні потреби користувачів та ефективно реагувати на вимоги пошукових роботів. Для цього доцільно проаналізувати перелік функціональних модулів веб-сайту [19].

Пошукові роботи працюють індексуючи посилання на сайти. Чим коректнішою і простішою буде структура сайту, тим менше ресурсів витрачатимуть пошукові роботи і тим швидше будуть скануватися ресурси. Це покращить індексацію веб-сайту.

Ефективний шлях від головної сторінки до основних компонентів веб-сайту повинен бути не більше трьох переходів, щоб максимально спростити процеси навігації [20].

На рисунку 2.2 зображена узагальнена структура веб-сайту «Дитячий лікар» м. Кременець.

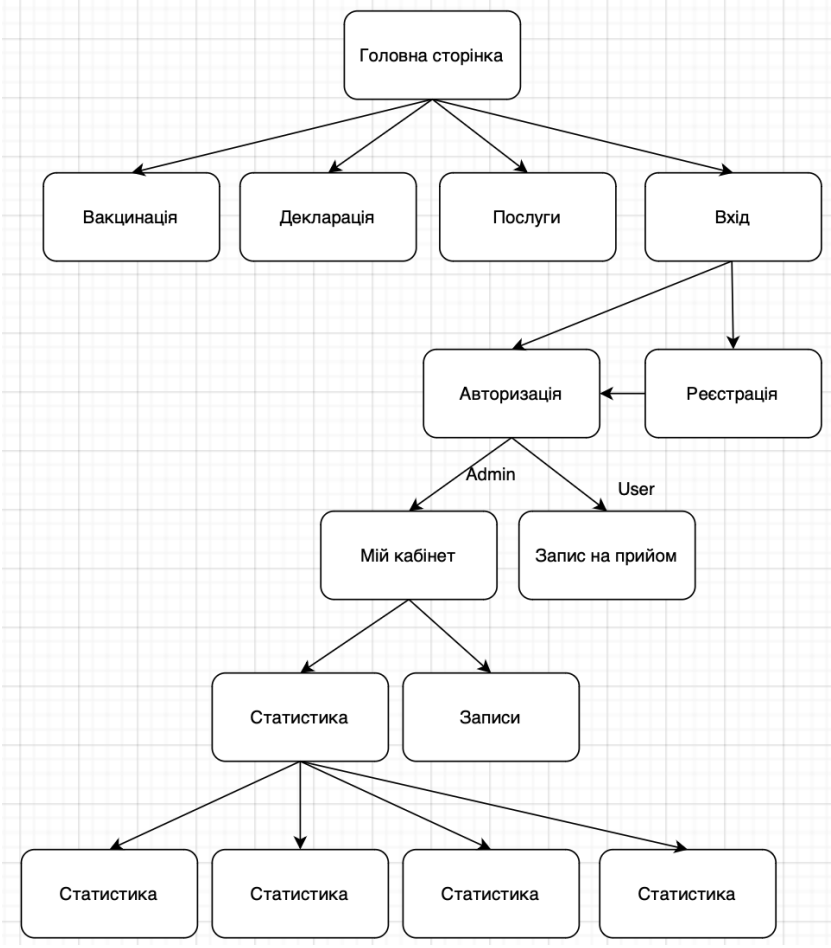


Рисунок 2.2 – Узагальнена структура веб-сайту «Дитячий лікар» м. Кременець

Зазначені нюанси повинні беззаперечно враховуватися не етапі розробки веб-сайту, адже коректна структура веб-сайту – запорука його успіху та ефективності.

2.3 Проектування UML-діаграм класів веб-сайту «Дитячий лікар» м. Кременець

Згідно сформованих у попередніх параграфах вимог до веб-сайту «Дитячий лікар» м. Кременець, його структура буде реалізована на основі шаблону проектування MVC.

Суть даного підходу – це відділення бізнес логіки від інтерфейсу. Бізнес логіка – це програмно-алгоритмічний функціонал, який реалізовано в конкретному веб-застосунку. Наприклад процедура реєстрація нового користувача визначає особливості хешування пароля, збереження даних користувача в базу даних, додавання товарів у кошик, оплата замовлення, збереження адреси користувача тощо. При цьому задіяний безпосередньо графічний інтерфейс, тобто елементи форми, кнопки текст тощо. [21].

Між UI інтерфейсом користувача та рівнем бізнес логіки повинен бути контролер – це сполучна ланка. Завдяки контролеру відбувається сильна зв'язаність процесів. Тому їх можна буде підміняти. У шаблоні «mvc» інтерфейс позначається «View», а контролер так і позначається. Шар бізнес логіки позначається «Model» [22]. Наприклад відображення містить форму з кнопкою «Зберегти». Користувач заповнює форму та натискає «Зберегти». При натисканні цієї кнопку виконуються операції всередині «Контролера». Зокрема можна відслідковувати події, перевіряти коректність даних та якщо все добре то передати їх з форми в модель. Вона містить бізнес правила, наприклад опис процесів, реєстрації користувача, збереження пароля, хешування тощо. Контролер при цьому не повинен містити будь-якої логіки Максимум опис процесів перевірки даних. Він повинен завжди залишатися «тонким» та бути

сполучною ланкою між UI і «Моделлю». На рисунку 2.3 використана структура шаблону «MVC».

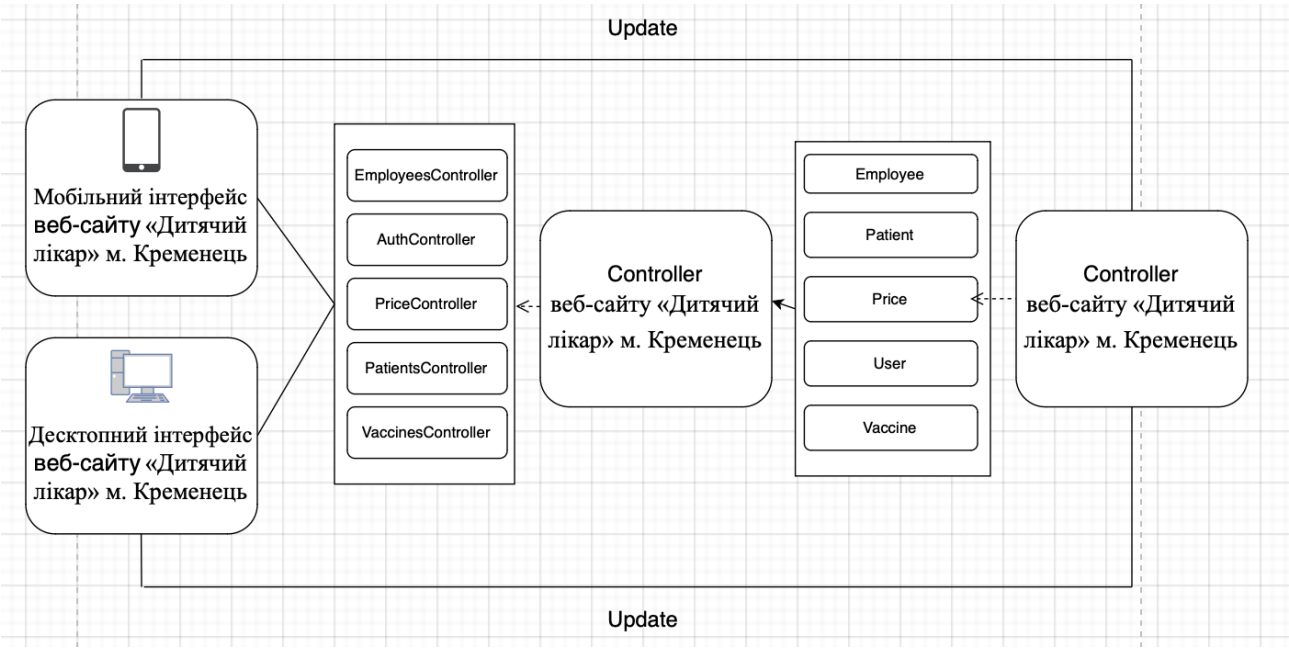


Рисунок 2.3 – Структура MVC шаблону веб-сайту «Дитячий лікар» м. Кременець

На рисунку 2.4 зображено діаграму класів, яка демонструє роботу «MVC» шаблону серверної частини веб-сайту «Дитячий лікар» м. Кременець.

У класі «Model» реалізовано взаємодію з базою даних «Mongo DB». Атрибутами містяться дані про пацієнта, користувача, прайсу на вакцини. У них зберігається вся інформації у форматі «JSON».

На основі класу «View» реалізовано графічний інтерфейс веб-сайту «Дитячий лікар» м. Кременець. Тут передається на фронтенд атрибут «pageName», у якому зберігається назва сторінки для ініціалізації при завантаженні графічного інтерфейсу. Доступні методи «Update()» та «Render()». Перший метод оновлює всю необхідну інформацію про інтерфейс. Другий метод завантажує інформацію та виводить графічний інтерфейс.

У класі «Controller» реалізовано обробку проміжної інформації між «Model» та «View».

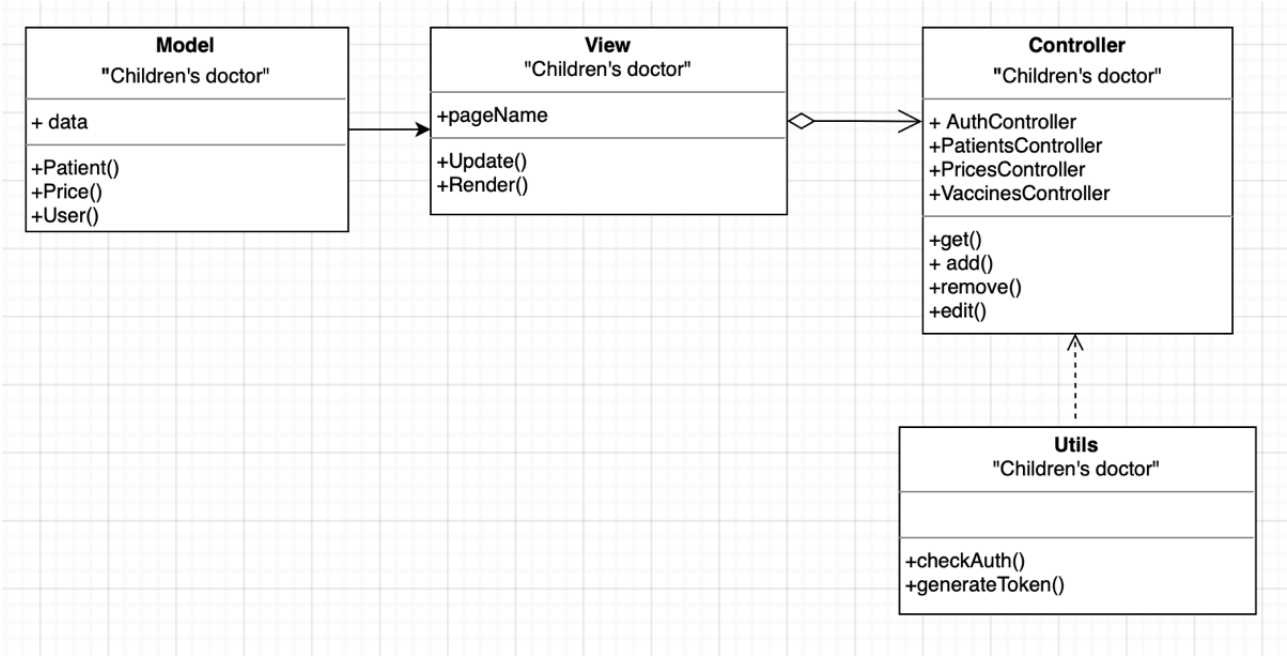


Рисунок 2.4 – Діаграма класів «MVC» шаблону веб-сайту «Дитячий лікар» м. Кременець

Присутні атрибути: «AuthController», «EmployeesController», «oldPatientsController», «PatientsController», «PricesController» та «VaccinesController». Є методи «get()», «add()», «remove()», «edit()», «login()», «registration()» та «getMe()». Для перших чотирьох виконуються загальні запити. Решта три методи відповідають за реєстрацію та ініціалізацію користувачів. А метод getMe() перевіряє чи користувач увійшов до системи за допомогою токена, який ґенерується при авторизації.

2.4 Структура каталогів веб-сайту «Дитячий лікар» м. Кременець

Узагальнена структура веб-сайту «Дитячий лікар» м. Кременець складається з двох основних компонентів це «client» та «server». Перший містить інтерфейс кожної окремої сторінки. Завдяки цьому виконується взаємодія користувача з веб-інтерфейсом. За допомогою серверної частини виконується підключення та взаємодія з БД. Водночас тут описані роути між сторінкими та узагальнена структура «MVC» (див. рисунок 2.5).

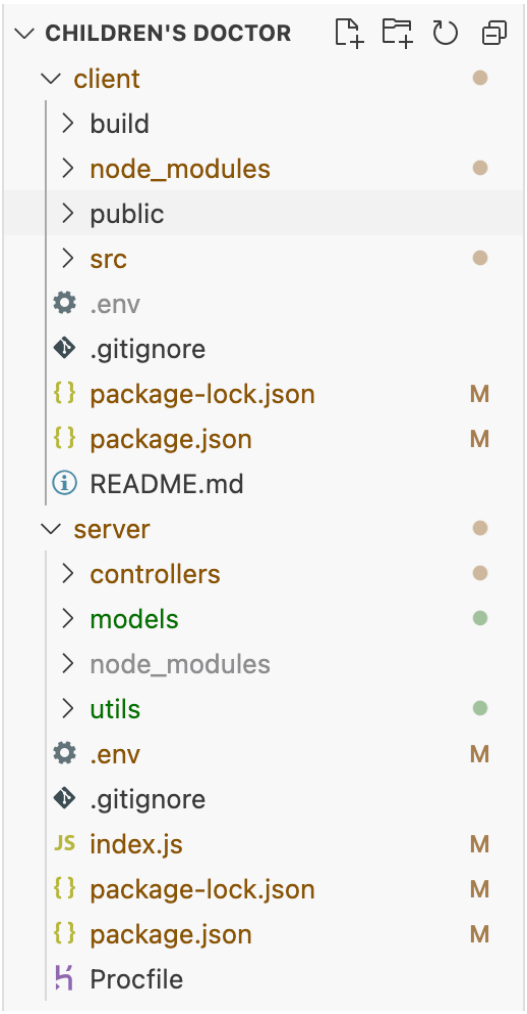


Рисунок 2.5 – Структура каталогів веб-сайту «Дитячий лікар» м. Кременець

Розглянемо детальніше ці два компонента. У клієнтській частині (див. додаток А знаходить основний «.html» файл, який ініціалізує роботу всього веб-інтерфейсу. У папці «src» знаходяться сторінки з описаними інтерфейсами.

У папці «Components» знаходяться основні функціональні елементи веб-сайту, які відображаються на кожній сторіці. Це «header», «footer» та «phone» бічна панель інформації для мобільної версії.

Наступною є папка з усіма використовуваними фото. «LogReg» відповідає за функціональні елементи з формою реєстрації та логіну.

Папка «pages» містить головні сторінки веб-інтерфейсу сайту «Дитячий лікар» м. Кременець та його окремі елементи. Зокрема модальні вікна для запису на прийом до лікаря та елементи взаємодія з вікнами додавання та редагування даних. Окремим компонентом є каталог «statistics», в якому

знаходиться основна веб-сторінка «statistics.jsx» та дочірні каталоги «pages» та «components». У першому знаходяться сторінки, «роути» які описані у батьківському файлі. В другому знаходяться дані для статистики та «sidebar».

У веб-сайті «Дитячий лікар» м. Кременець використовується бібліотека «Redux» для керування процесами авторизації користувача. У цій папці описані «редюсери» для керування статусом користувача та «store» – об'єкт, який з'єднує ці частини разом.

У файлі «axios.js» подано конфігурації для HTTP-клієнта.

«App.js» це файл для компонента програми. Компонент програми є основним компонентом у React, який діє як контейнер для всіх інших компонентів. «index.css» це файл «CSS», що відповідає «index.js». «index.js» це файл «javascript», що відповідає «index.html». Цей файл містить рядок коду говорить про те, що «App Component» потрібно завантажити в елемент «html» з ідентифікатором «root». «App.test.js» призначений для тестування усіх компонентів. «.env» містить окремі змінні середовища користувача, які замінюють змінні. «gitignore» – вказує навмисно невідстежувані файли, які потрібно ігнорувати. «package.json» файл містить список важливих залежностей вузла. «package-lock.json» автоматично ґенерується для будь-яких операцій, де «npm» змінює або «node_modules» дерево, або «package.json». Він описує точне DOM-дерево, яке було зґенеровано на даний час, щоб наступні інсталяції могли ґенерувати ідентичні дерева, незалежно від проміжних оновлень залежностей.

Серверна частина (див. додаток Б) має схожу структуру файлів. У папці «Components» знаходяться файли, в яких описані методи для взаємодії з БД. Папка «models» містить файли з описами моделей. У папці «utils» знаходяться два файли «checkAuth.js» та «generateAccessToken.js». В першому описано процес перевірки існування даних щодо користувача з токеном присвоєним у момент аутентифікації. Другий ґенерує цей токен при аутентифікації.

Далі розміщено множину файлів, зокрема «.env», «gitignore», «package.json» тощо. Вони призначення так само як і клієнтській частині.

2.5 Програмна реалізація веб-сайту «Дитячий лікар» м. Кременець

Для розробки веб-сайту «Дитячий лікар» м. Кременець було використано бібліотеку «React js», «react bootstrap», «redux», «express» та «node js» з СКБД «Mongo DB». Застосунки, які використовують даний стек веб-технологій називають «MERN», тобто «MongoDB», «Express», «React», «Node».

Для базового функціонування веб-сайту «Дитячий лікар» м. Кременець в файлі «index.js» серверної частини потрібно описати базові налаштування для роботи сервера та порта (лістинг 2.1).

Лістинг 2.1 – Код файлу «index.js»

```
const PORT = process.env.PORT || 5005;
const app = express();
app.use(cors());
app.use(express.json());
const start = async () => {
     try {
          await mongoose.connect(process.env.URL);
          app.listen(PORT, () => {
          console.log("Server started");
     });
} catch (err) {
     console.log(err);
}
};
start();
```

У цьому файлі в зміну «PORT» записується на якому порті буде працювати веб-сайт «Дитячий лікар» м. Кременець. У файлі «.env» записано адресу хостингу на якому розміщено серверну частину. Якщо він відсутній призначається «localhost 5005». Далі ініціалізуємо застосунок модулем «express». За домомогою функції «cors()» можна буде відправляти запити на сервер з різних IP адрес. «Express.json()» отримує дані із сервера у форматі «JSON». У функції «start()» підключається БД. «app.listen()» використовується для зв’язування та прослуховування з’єднань на вказаному хості та порту. Для переходу між сторінками та виконання «CRUD» операцій, тобто чотирьох

основних функцій управління даними «створення», «читання», «оновлення» і «вилучення» [23]. У цьому ж файлі описані «роути» (див. лістинг 2.2)

Лістинг 2.2 – «Роути» для сторінки з вакцинами

```
app.get("/vaccines", VaccinesController.getVaccines);
app.post("/vaccines", VaccinesController.addVaccine);
app.delete("/vaccines/:id", VaccinesController.removeVaccine);
app.put("/vaccines/:id", VaccinesController.editVaccine);
```

Як було зазначено в попередніх параграфах веб-сайт «Дитячий лікар» м. Кременець сформовано на основі шаблоні «MVC», який включає в себе контролери та моделі серверної частини. Модель кожної сутності подана в окремому файлі. Приклад моделі «User» поданий в лістингу 2.3.

Лістинг 2.3 – Код моделі «User»

```
const { Schema, model } = require("mongoose");
const User = new Schema({
     username: { type: String, unique: true, required: true },
     password: { type: String, required: true },
     role: { type: String, required: true },
});
module.exports = model("User", User);
```

З бібліотеки «mongoose» імпортуються потрібні модулі. Дана модель містить три поля: «username», «password» та «role», яка має два значення «admin» та «user». Поля містять свої властивості їх тип може бути «string», «number» тощо. «unique» перевіряє унікальність логіну, а «required» задає обов’язковість заповнення поля.

Подібну структуру містить решта моделей «patient», «price», «vaccine». Для кожної моделі створені відповідні контролери, в яких описано функціонал для «входу», «реєстрації», «отримання», «додавання», «видалення» та «редагування» даних.

У файлі «AuthController.js» описано структуру трьох функцій: «реєстрації», «аутентифікації» та «авторизації» користувача.

У додатку В подано програмний код функції для реєстрації нового користувача. Зазначеня функція є асинхронною та приймає два аргументи «req» та «res». Перший – це дані які функція отримала з «фронтенд» частини веб-сайт «Дитячий лікар» м. Кременець. Другий – це то, що потрібно передати у фронтенд. У блоці «try catch» відбувається перевірка даних та додавання користувача в БД. На початку перевіряється чи всі поля були коректно заповненні. Потім перевіряється чи не існує користувача з введеними «username» та «password». Якщо такий є то ґенерується відповідне повідомлення. Якщо дані успішно перевірено то за допомогою відповідної моделі створюється обліковий запис нового користувача, а в цей момент пароль хешується за допомогою функцій відповідної бібліотеки [24]. Відповідні дані записуються в БД та передається повідомлення у «фронтенд», що користувач був створений.

У додатку Д описана функція аутентифікації попередньо створеного облікового запису користувача.

Зазначена функція також є асинхронною. Вона приймає подібні до описаних аргументи. На початку її виконання відбувається перевірка коректності даних. Потім розшифровується пароль. За допомогою бібліотеки «jwt» створюється відповідний токен, який в подальшому буде використовуватися, щоб дізнатися чи користувач авторизований та щоб при кожному оновленні веб-інтерфейсу він зберігався:

```
const token = jwt.sign({ id: user._id }, process.env.JWT_KEY, {
     expiresIn: "30d",
});
```

Токен створюється на основі унікального «id» користувача та ключа, який також записано у файл «.env». Якщо все пройшло успішно дані передаються на форонтенд. У додатку Е описана функція для перевірки авторизації користувача. У ній також записується унікальний токен користувача та передаються відповідні дані на фронтенд.

Щоб при кожному оновленні сторінки користувача не «деавторизувало» викликається функція «checkAuth()». В ній видобувається з «headers» токен та декодується за допомогою бібліотеки «jwt». При успішній операції викликається параметр «next()» та відбувається перехід до виконання наступної функції (див. додаток Ж).

У лістингу 2.4 подано код функці отримання даних з БД, наприклад прайсу.

Лістинг 2.4 – Запит для отримання даних «Прайс» з БД

```
const getPrices = async (req, res) => {
     try {
          const data = await Price.find();
          if (data) {
                res.status(200).json(data);
     } else {
          res.status(400).json({ success: false });
     }
     } catch (err) {
          res.status(400).json({ success: false });
          console.log(err);
     }
};
```

В цій функції записуються дані в змінну «data» та за допомогою відповідної моделі шукаються та повертаються на сервер у форматі «json». Якщо отримано статус «200» або при виникненні помилки, на сервер повертається повідомлення про збій.

У лістингу 2.5 подано «js»-код функції додавання нових даних в БД наприклад «прайсу» послуг.

Лістинг 2.5 – «js»-код запиту для додавання даних в БД

```
const addPrice = async (req, res) => {
     try {
          if (req.body.title && req.body.price) {
                const { title, price } = req.body;
                const newPrice = new Price({ title: title, price:
price });

const isSuccess = await newPrice.save()
```

```
     if(isSuccess){
          res.status(200).json({success:true})
     }
}
} catch (error) {
     res.status(400).json({ success: false });
     console.log(error);
     }
};
```

В ній дані з «фронтенду» передаються як параметр відповідній моделі. Потім нова модель зберігається в БД методом «save()».

У лістингу 2.6 подано «js»-код функції видалення даних з БД, наприклад «прайсу» послуг.

Лістинг 2.6 – «js»-код запит видалення даних з БД

```
const removePrice = async (req, res) => {
     try {
          const id = req.params.id;
     if (id) {
          await Price.findByIdAndDelete(id)
          res.status(200).json({success:true})
     }
     } catch (err) {
          console.log(err);
          res.status(400).json({ success: false });
     }
};
```

Видалення потрібних даних відбувається за допомогою унікального «id», що передаються методу «findByIdAndDelete(id)». Він викликається у відповідній моделі. Після успішної операції на сервер відправляється відповідне повідомлення. У лістингу 2.7 подано «js»-функцію редагування даних в БД, наприклад «прайсу» послуг.

Лістинг 2.7 – «js»-запит для редагування даних з БД

```
const editPrice = async (req, res) => {
     try {
          const id = req.params.id;
          const {title, price} = req.body
     if (id) {
```

```
            const newPrice = await Price.findById(id);
            newPrice.title = title;
            newPrice.price = price;
            const result = await newPrice.save()
            if(result) {
                  res.status(200).json({ success: true });
            } else {
                  res.status(400).json({ success: false });
            }
      }
      }
      catch (err) {
            res.status(400).json({ success: false });
            console.log(err);
}     };
```

Дані, які потрібно редагувати також знаходяться за допомогою «id». Потім отримані нові дані з «фронтенд»-частини записуються в нову модель та зберігаються в БД методом «save()».

У фронтенді є папка «Redux», в якій знаходяться функції «логіну», «реєстрації» та «отримання» даних про те чи є на веб-сторінці зареєстрований користувач незалежно від рівня доступу. Тут отримуються булеві значення щодо аутентифікації користувача, які експортуються та використовуються, щоб відображати вибрану інформацію для окремого користувача.

Для роботи з «Redux» потрібний відповідний «state», тобто об’єкт в якому будуть зберігатися отримані з сервера дані (див. лістинг 2.8).

Лістинг 2.8 – «state» для зберігання даних про користувача

```
const initialState = {
  user: null,
  token: null,
  isLoading: false,
  status: null,
  role: null,
};
```

За допомогою «редюсера» «logout» цей «state» обнуляється для деініціалізації користувача в системі. Також присутні «extraReducers», в яких заповнюється «state» відповідно до дії «реєстрації», «логіну» чи «перевірки

авторизації». Для цього експортуються окремі булеві значення, «редюсери» та звичайні дані з сервера (див. лістинг 2.9).

Лістинг 2.9 – Експорт булевих значень з «redux»

```
export const checkIsAuth = (state) => Boolean(state.auth.token);
export const isAdmin = (state) => Boolean(state.auth.role == "admin");
export const isDoctor = (state) => Boolean(state.auth.role == "doctor");
export const docName = (state) => state.auth.user.username
export const { logout } = authSlice.actions;
```

Функція «checkIsAuth» відповідає за визначення рівня доступу для облікового запису звичайного користувача в процесі подачі заяви на прийом. Булеві значення «isAdmin», «isDoctor» забезпечують відображення веб-сторінок для цих рівнів доступу. Змінна «docName» – це ім’я лікаря, який увійшов в систему. Воно використовується на сторінці запису, щоб відображати пацієнтів лише авторизованого лікаря. Функція «Logout» очищує дані щодо авторизованого користувача при виході. Щоб отримати доступ до цих «змінних» потрібно їх імпортувати у відповідному місці та за допомогою функції «useSelector()» добути дані зі стану сховища «Redux».

2.6 Тестування «React»-компонентів веб-сайту «Дитячий лікар» м. Кременець

«React застосунки», які створенні за допомогою команди «Create React App», містять стандартну підтримку «React Testing Library». Завдяки цьому можна створювати тести власних компонентів [25].

«Jest» – це фреймворк для тестування на «JavaScript», який створює середовище для написання та виконання тестів. Бібліотека тестування «React «надає різноманітні тестові помічники, які виконують тести на основі взаємодії користувачів, а не специфіки реалізації компонентів. Обидва варіанти

тестування попередньо упаковані з процесу створення «React-застосунку», який сформовано завдяки таким самим принципам. Тому прогнозоване використання програмних елементів повинно бути схожим на їх фактичну реалізацію [26].

Завдяки зазначеним ресурсам можна проводити «unit»-тестування, яке містить:

– тестування асинхронних функцій;

– тестування подій;

– тестування компонентів з асинхронним завантаженням даних;

– тестування «роутингу»;

– інтеграційне тестування в поєднанні з redux toolkit;

– тестування селектора тощо.

Після створення «React-застосунку» для веб-сайту «Дитячий лікар» м. Кременець, можна формувати тести. Для даної роботи розроблено три тести, зокрема:

– тестування компонента з асинхронним завантаженням даних з сервера – файл «AddVac.test.js»;

– інтеграційне тестування в поєднанні з «react router dom v6» – файл «Header.test.js»;

– інтеграційне тестування в поєднанні з «Redux Toolkit»;

У додатках З та И подано лістинги для двох перших тестів. У лістингу 2.10 подано код тестування «редюсера» з «Redux Toolkit».

Лістинг 2.10 – Інтеграційне тестування в поєднанні з «Redux Toolkit»

```
import { authSlice, logout } from "./authSlice";
describe("LogOut", () => {
     test("logout", () => {
     expect(
     authSlice({
          user: {username: "admin", id: "636fd939c8**********8d95",
     role: "admin"},
          token:
     "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYzNmZkOTM5Yzg4
MDBmYm**********************************************jUxfQ.UTGVCbXM
ZFCAjLNIAnujJ3yURJNhMni_BsYMsdDQxbM",
          isLoading: false,
```

```
            status: "success",
            role: "admin",
        },
        logout())
        ).toEqual({
        user: null,
        token: null,
        isLoading: false,
        status: null,
        role: null,
            });
        });
});
```

У «test case» передається очікуваний результат виконання. В даному випадку це – очищення «state». Оскільки «reducer» отримує перший аргумент «state», а другий аргумент «action», який і міняє цей «state». В «authSlice» перший аргумент – це дані про авторизованого користувача. В другий аргумент передамо «action» та на виході очікуємо, що «state» очиститься.

Тест запустити можна за допомогою команди «npm test AuthSlice.test.js». На рисунку 2.6 зображено результат тесту.

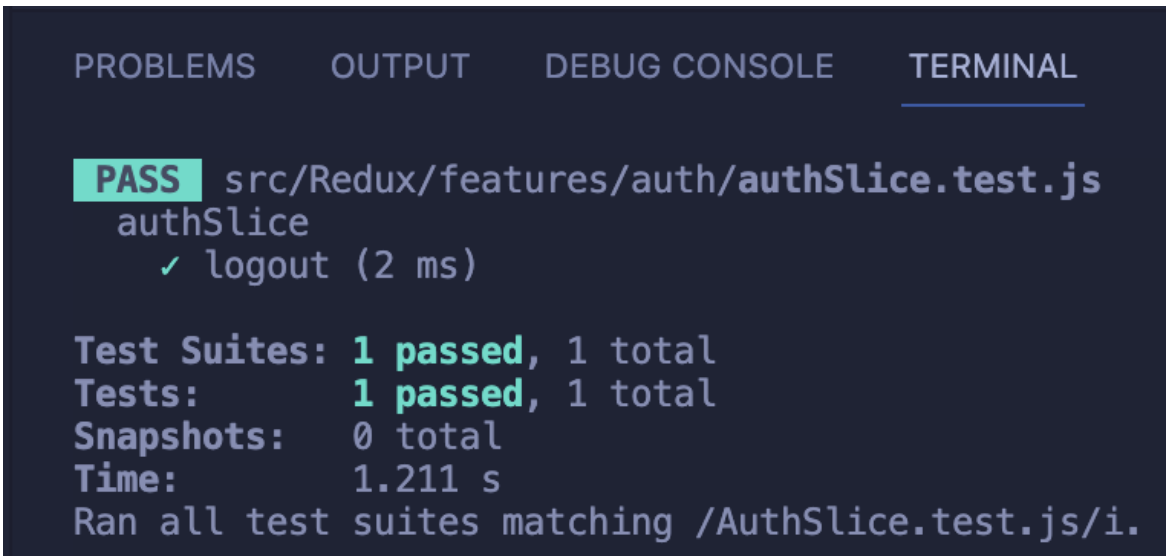


Рисунок 2.6 – Результат виконання тесту «редюсера» з «Redux Toolkit»

Зауважимо що «state» успішно змінився та тест успішно відбувся. Після завершення можна оцінити кількість запущених тестових файлів та кількість «test case», які присутні в усіх тестах.

2.7 Висновок до другого розділу

В другому розділі кваліфікаційної роботи освітнього рівня «Бакалавр» змодельовано архітектуру веб-сайту «Дитячий лікар» м. Кременець. Сформовано структурну модель веб-сайту «Дитячий лікар» м. Кременець. Спроектовано UML-діаграми класів веб-сайту «Дитячий лікар» м. Кременець. Розроблено структуру каталогів веб-сайту «Дитячий лікар» м. Кременець. Описано програмну реалізацію веб-сайту «Дитячий лікар» м. Кременець.

РОЗДІЛ 3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1 Долікарська допомога при ураженні електричним струмом.

Нещасні випадки ураження електричним струмом є поширеним явищем як у побуті, так і на роботі, і становлять значну загрозу для життя людини. Своєчасне надання першої допомоги може значно зменшити тяжкість травм і підвищити шанси на виживання. Розуміння важливості надання першої допомоги при ураженні електричним струмом має вирішальне значення для того, щоб люди були готові та ефективно реагували на такі надзвичайні ситуації.

Електричний струм, коли проходить через тіло людини, може мати серйозні наслідки для неї. Природа цього впливу призводить до того, що електричний струм може збудити нервову систему та спричинити скорочення м'язів. Залежно від сили струму і тривалості впливу, можуть виникнути різні ураження, від незначних пошкоджень до серйозних травм. Усім органам тіла, зокрема серцю, нервовій системі та м'язам, потрібен електричний імпульс для правильної функції. Однак, при ураженні електричним струмом, непередбачувана дія струму може порушити цей ритм та функції органів.

Види поразки електрикою:

– Електричний удар (шок) – вплив на весь організм, він не викликає опіків, а призводить до паралічу дихання і / або серця.

– Електрична травма – поразка зовнішніх частин тіла: електричні знаки, опіки, металізація шкіри.

Симптоми ураження електричним струмом:

– Несподіване падіння людини на вулиці або неприродне відкидання від джерела струму невидимою силою.

– Втрата свідомості, судоми.

– Виражені скорочення м’язів мимовільного характеру.

– Випадення неврологічних функцій — втрата пам’яті, порушення розуміння мови та зору, порушення орієнтації в просторі, зміна шкірної чутливості, реакції зіниці на світло.

– Фібриляція шлуночків і зупинка дихання — нерівний пульс і нерівне дихання.

– Опіки на тілі з різко окресленими кордонами [27].

Тривалість дії струму безпосередньо впливає на важкість ураження. Чим довше діє струм, тим сильніше ураження і більша ймовірність смерті. Струм високої напруги проводить до різкого скорочення м’язів, людина навіть може бути з силою відкинутий від джерела струму. Струм низької напруги провокує спазм м’язів призводить до тривалого мимовільному захопленню провідника руками. З плином часу зменшується опір шкіри, тому необхідно якомога раніше перервати контакт потерпілого з провідником.

Послідовність дій при наданні домедичної допомоги постраждалим при ураженні електричним струмом:

1. Перед наданням допомоги переконатися у відсутності небезпеки для себе, оточуючих, постраждалого та тільки за її відсутності перейти до наступного кроку.

2. Якщо постраждалий у свідомості, заспокоїти та пояснити свої наступні дії.

3. Здійснити виклик екстреної медичної допомоги та дотримуватись вказівок диспетчера прийому виклику.

4. Якщо постраждалий без свідомості, впевнитись, що дія електричного струму на постраждалого припинена. Якщо дія електричного струму на постраждалого припинена, слід надати домедичну допомогу.

5. Забезпечити постійний нагляд за постраждалим до приїзду бригади екстреної (швидкої) медичної допомоги;

6. При погіршенні стану постраждалого до приїзду бригади екстреної (швидкої) медичної допомоги повторно здійснити виклик екстреної медичної допомоги.

7. За можливості зібрати у постраждалого чи оточуючих максимально можливу інформацію стосовно обставин отримання травми. Всю отриману інформацію передати працівникам бригади екстреної (швидкої) медичної допомоги або диспетчеру служби екстреної медичної допомоги [28].

Всі дії повинні здійснюватися дуже швидко, без затримок, зайвих розмов і міркувань. Своєчасне надання допомоги дозволяє зберегти життя і зменшити тяжкість електротравми. Який б не був стан потерпілого, слід негайно викликати швидку або доставити людину до медичного закладу. Смерть від удару струмом може наступити і через кілька годин. Зовнішня картина не відображає внутрішніх пошкоджень після удару електричним струмом.

Якнайшвидше припинити контакт потерпілого з провідником струму. Розімкнути ланцюг за допомогою непровідних струм предметів (відтягнути провід дерев'яною палицею) або висмикнути вилку приладу з розетки, відключити струм. Відтягнути потерпілого від джерела струму за допомогою предметів, що не проводять струм, і не торкаючись тіла: дерев'яними палицями, дерев'яним стільцем, мотузкою, волоком на відстань не менше 10 м. Оцінити стан дихальної та серцево-судинної систем і чи в свідомості людина. Легенько поплескати по щоці, задати елементарні запитання. При необхідності провести реанімаційні заходи:

Первинна реанімація постраждалого (при відсутності пульсу та дихання):

– Непрямий масаж серця – найбільш ефективний протягом 3 перших хвилин після зупинки серця.

– Дихання рот в рот – по два повних видиху через кожних 30 натискань на проекцію серця.

– Тривалість реанімаційних заходів – до приїзду швидкої або до появи ознак життя.

– Первинна обробка опіків полягає в накладенні сухою марлевою пов'язки.

– Знеболюючі – при збереженні свідомості до приїзду швидкої людині можна дати знеболююче і заспокійливий [29].

Один із важливих аспектів є профілактика пошкодження електричним струмом. Для цього необхідно регулярно перевіряти та обслуговувати електрообладнання, підтримувати правила електробезпеки, використовувати пристрої захисного відключення (ПЗВ), запобігати перевантаженню ланцюгів та впроваджувати належні заходи із заземлення та ізоляції.

Отже, швидка та ефективна перша допомога при ураженні електричним струмом є критично важливою, оскільки вона сприяє мінімізації можливої шкоди та значно підвищує шанси вижити потерпілого. Проте не менш важливою є і профілактика та прийняття заходів для зниження ризику пошкодження електричним струмом. Відповідне навчання, свідоме використання захисного обладнання, перевірка технічного стану електрообладнання та дотримання правил безпеки можуть стати критичними у запобіганні подібних небажаних ситуацій [30].

3.2 Естетичне оформлення робочого місця працівника в ІТ

У сучасному світі інформаційні технології стали невід'ємною частиною успіху кожної організації. Як наслідок, компаніям важливо інвестувати в дизайн робочих місць своїх співробітників, щоб забезпечити максимальну продуктивність і задоволеність. Хоча дехто може критикувати естетичний дизайн робочого місця працівника як непотрібну витрату, це есе доводить, що створення естетично привабливого середовища може мати позитивний вплив на задоволеність і продуктивність працівників, що в кінцевому підсумку призводить до кращих результатів бізнесу [31]. У цьому розділі розглядається важливість естетичного дизайну на робочих місцях в ІТ і те, як він може покращити самопочуття та продуктивність працівників.

Планування робочого місця ІТ-працівника відіграє важливу роль у створенні комфортного та візуально привабливого робочого простору, що сприяє продуктивним робочим процесам. Це може включати в себе ергономічні меблі та обладнання, оптимізацію освітлення та повітряних потоків,

розміщення настільних аксесуарів та декору для оптимальної функціональності, а також персоналізацію аспектів робочого простору відповідно до вподобань та стилю роботи працівника.

Такі міркування щодо естетики та ергономіки не лише сприяють фізичному благополуччю, але й позитивно впливають на задоволеність працівника роботою, моральний дух і, зрештою, на продуктивність праці. У контексті роботи з інформаційними технологіями інвестиції в естетично привабливий дизайн робочого простору можуть також сприяти формуванню позитивної культури на робочому місці, яка заохочує творчість, інновації та співпрацю між членами команди, що, зрештою, призводить до покращення бізнес-результатів. Тому організаціям, що працюють в галузі інформаційних технологій, вкрай важливо приділяти першочергову увагу стратегічному дизайну робочих місць співробітників, щоб не лише максимізувати комфорт і ефективність, але й створити сприятливе і стимулююче робоче середовище, яке дає змогу працівникам працювати на оптимальному рівні. Отже, естетичний дизайн робочого місця IT-працівника є вирішальним фактором, який не повинен залишатися поза увагою організацій, що працюють у цій галузі [32].

Перш за все, естетично привабливе робоче місце може мати значний вплив на настрій та психічне благополуччя працівника. Візуально привабливий робочий простір може створити заспокійливий ефект і знизити рівень стресу, що призводить до підвищення продуктивності та задоволеності роботою. З іншого боку, тьмяний і не надихаючий робочий простір може мати протилежний ефект, що призводить до зниження мотивації, креативності та залученості працівників. Тому дуже важливо створити робочий простір, який буде естетично привабливим, комфортним і сприятиме продуктивності [33].

По-друге, естетично привабливе робоче місце також може позитивно вплинути на імідж та репутацію організації. Добре спроектований робочий простір може створити позитивне перше враження у відвідувачів і клієнтів, відображаючи цінності, культуру і прагнення до досконалості організації. Він також може допомогти залучити й утримати найкращі таланти, демонструючи

відданість організації створенню комфортного та привабливого робочого середовища.

Щоб створити естетично привабливе робоче місце, потрібно врахувати кілька елементів дизайну. Освітлення, колір, меблі та декор – все це важливі компоненти, які сприяють загальній естетичній привабливості робочого простору. Освітлення має бути яскравим, природним і рівномірно розподіленим, щоб створити відчуття відкритості та енергії. Колірна гамма повинна бути ретельно підібрана, щоб відображати бренд і культуру організації, створюючи при цьому відчуття гармонії та балансу. Меблі мають бути ергономічними, зручними та функціональними, забезпечуючи працівників необхідними інструментами та ресурсами для ефективного виконання своєї роботи. Нарешті, декор повинен бути привабливим і надихаючим, відображати цінності організації та заохочувати творчість та інновації [34].

На додаток до фізичних елементів дизайну, планування та організація робочого простору також можуть впливати на загальну естетичну привабливість. Робочий простір має бути організованим, вільним від захаращення та зручним для навігації, створюючи відчуття порядку та структури. Використання рослин, творів мистецтва та інших декоративних елементів також може сприяти естетичній привабливості робочого простору, створюючи відчуття тепла, енергії та творчості [32].

Отже, естетичний дизайн робочого місця працівника в IT має важливе значення для створення комфортного, привабливого та продуктивного робочого середовища. Вдало спроектований робочий простір може позитивно впливати на настрій і психічне благополуччя працівника, підвищувати продуктивність і задоволеність роботою, а також покращувати імідж і репутацію організації. Тому організації повинні визначати пріоритети у створенні робочого простору.

ВИСНОВКИ

В ході виконання даної кваліфікаційної роботи було успішно вирішено завдання розробки веб-сайту для дитячої клініки «Дитячий лікар» м. Кременець. Робота була розділена на три розділи, кожен з яких присвячений окремим виконання поставленого задання, починаючи від аналізу предметної області і закінчуючи реалізацією програмного забезпечення та аспектами безпеки життєдіяльності та охорони праці.

В першому розділі кваліфікаційної роботи:

– Проаналізовано предметну область.

– Сформовано перелік вимог до веб-сайту.

– Розглянуто варіанти використання веб-сайту.

– Спроєктовано програмно-алгоритмічну архітектуру веб-сайту «Дитячий лікар» м. Кременець.

– Сформовано підсистему зберігання даних веб-сайту «Дитячий лікар».

В другому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»:

– Змодельовано архітектуру веб-сайту «Дитячий лікар» м. Кременець.

– Сформовано структурну модель веб-сайту.

– Спроектовано UML-діаграми класів веб-сайту.

– Розроблено структуру каталогів веб-сайту «Дитячий лікар» м. Кременець.

– Описано програмну реалізацію веб-сайту «Дитячий лікар» м. Кременець.

Розділ «Безпека життєдіяльності, основи охорони праці» розширив рамки кваліфікаційної роботи, розглянувши найважливіші аспекти безпеки життєдіяльності та основи професійної гігієни. Описано процес надання першої медичної допомоги при ураженні струмом, підкреслено важливість забезпечення безпечного та здорового робочого середовища. Було обговорено естетичне оформлення робочого місця працівника в сфері інформаційних технологій,

визнаючи важливість ергономічних міркувань та створення привабливої та сприятливої робочої атмосфери.

В кваліфікаційній роботі продемонстровано успішну розробку програмно-алгоритмічного комплексі для веб-сайту медичної клініки «Дитячий лікар», м. Кременець. Завдяки ретельному аналізу, проектуванню та впровадженню, функціональні та структурні елементи веб-сайту було ефективно адаптовано до специфічних потреб та вимог медичної клініки. Це дозволило покращити якість обслуговування пацієнтів та оптимізувати адміністративні процеси. Крім того, в кваліфікаційній роботі підкреслено важливість елементів безпеки життєдіяльності та охорони праці при розробці та експлуатацію ІТ-систем.

Успішне завершення цієї кваліфікаційної роботи дозволило отримати цінні знання в галузі веб-розробки, включаючи практичне застосування різних технологій, важливість систематичного аналізу та проектування, а також важливість врахування добробуту співробітників. Набуті практичні навички сприяють розширенню знань у цій галузі та можуть слугувати основою для подальшого розвитку в царині розробки веб-сайтів для закладів охорони здоров'я.

ПЕРЕЛІК ДЖЕРЕЛ

1 Михайлусь, Денис Павлович. "Розробка програмного забезпечення для керування вмістом музичного блогу." (2022).

2 Бєлоусов, Д. В. "Розробка серверного функціоналу веб-застосунків на базі мікросервісної архітектури." (2022).

3 SPA (Single-page application) [Електронний ресурс] / nschonni, OnkarRuikar // mdn. – 2022. – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Glossary/SPA.

4 Кульбачний, Владислав Васильович. "Інформаційна система транспортно-логістичного підприємства." (2023).

5 Руденко, Орест Ігорович. "Веб-додаток для моніторингу транспорту в м. Ірпінь." (2022).

6 Долгополов, О. Б. "Дослідження процесу розробки та створення веб-сайтів." (2022).

7 Буланов, Ілля Денисович. Веб-застосування з продажу автозапчастин використанням технології PWA. BS thesis. КПІ ім. Ігоря Сікорського, 2022.

8 Башовий, В. М., В. В. Стаценко, and Д. В. Стаценко. "Визначення швидкості роботи сучасних фреймворків для створення web-інтерфейсів." Технології та інжиніринг (2022).

9 Касянчук, Дмитро Павлович. Модифікований метод статичного аналізу коду для рішення задачі згортки рядкових констант. MS thesis. КПІ ім. Ігоря Сікорського, 2023.

10 Лембас, Максим Сергійович. "Веб-застосунок для продажу музичних інструментів." (2023).

11 Скрипнік, Михайло Сергійович. "Створення MVP-порталу новин." (2022).

12 React. A JavaScript library for building user interfaces [Електронний ресурс] // Meta Platforms, Inc.. – 2022. – Режим доступу до ресурсу: https://reactjs.org/.

13 React Bootstrap [Електронний ресурс] // Meta Platforms, Inc. – 2022. – Режим доступу до ресурсу: https://react-bootstrap.github.io/.

14 Yesin, V. I., and V. V. Vilihura. "Основні категорії NewSQL баз даних та їх особливості." Radiotekhnika 211 (2022): 37-66.

15 Sql Vs Nosql Exact Differences [Електронний ресурс] // myservername. – 2022. – Режим доступу до ресурсу: https://uk.myservername.com/sql-vs-nosql-exact-differences.

16 Петельчук, Віталій Максимович. "Комп'ютерна мережа видавничого підприємства." (2023).

17 Кантор І. Event loop: microtasks and macrotasks [Електронний ресурс] / Ілля Кантор // javascript.info. – 2022. – Режим доступу до ресурсу: https://javascript.info/event-loop.

18 Концепції Інтернету, TCP/IP та HTTP [Електронний ресурс] // IBM. – 2023. – Режим доступу до ресурсу: https://www.ibm.com/docs/en/cics-ts/5.3?topic=web-internet-tcpip-http-concepts.

19 Романенко, В. М. "ПРОЄКТУВАННЯ ТА РОЗРОБЛЕННЯ СЕРВІСНОЇ АРХІТЕКТУРИ УПРАВЛІННЯ БІЗНЕС-ПРОЦЕСАМИ УНІВЕРСИТЕТУ. СЕРВІС «АНАЛІТИКА»."

20 Як вивести правильну структуру сайту [Електронний ресурс] // Elit Blog. – 2021. – Режим доступу до ресурсу: https://elit-web.ua/ua/blog/kak-vyvesti-tekuschuju-strukturu-sajta.

21 Шаблон контролера представлення моделі - пояснення архітектури та фреймворків MVC [Електронний ресурс] // Rafael D. Hernandez. – 2021. – Режим доступу до ресурсу: https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/.

22 Кукса, Р. Ю. "Мобільний додаток для підвищення продуктивності і тренування уваги користувача." (2022).

23 Мальцев, М. Є. "Розробка програмного засобу по адмініструванню футбольних полів." (2022).

24 Коваленко, Р. А. Інформаційна технологія двофакторної авторизації із застосуванням Telegram. MS thesis. Сумський державний університет, 2023.

25 Дорофєєва, К. С. "Розробка платформи для проведення онлайн-виставки." (2022).

26 Як тестувати React-додаток за допомогою Jest та бібліотеки тестування React [Електронний ресурс] // Alyssa Holland. – 2022. – Режим доступу до ресурсу: https://www.digitalocean.com/community/tutorials/how-to-test-a-react-app-with-jest-and-react-testing-library.

27 Американська асоціація серця. Невідкладна серцево-судинна допомога: для медичних працівників / Американська асоціація серця., 2020. – 101 с. – (9781616697662).

28 Порядок надання домедичної допомоги постраждалим при ураженні електричним струмом або блискавкою [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0356-22#n800

29 Ле Бодур К. Швидка медична допомога: Перший на місці події / К. Ле Бодур, Д. Бержерон, К. Веслі., 2018. – 656 с. – (978-0134988467; № 11).

30 Маслова, Оксана Володимирівна, Ірина Петрівна Гончарова, and О. В. Маслов. "РИЗИКООРІЄНТОВНІ ПІДХОДИ В ОХОРОНІ ПРАЦІ." (2023).

31 Долубовська, Ольга Романівна. Соціальна відповідальність в управлінні готельно-ресторанним бізнесом (на прикладі ресторану «Соната»). MS thesis. 2022.

32 Міллер Р. Поштовх до здорового робочого місця: Як здорові люди, культура та будівлі ведуть до високої продуктивності / Р. Міллер, Ф. Вільямс, М. О'Ніл., 2018. – 320 с. – (Wiley). – (978-1119480129).

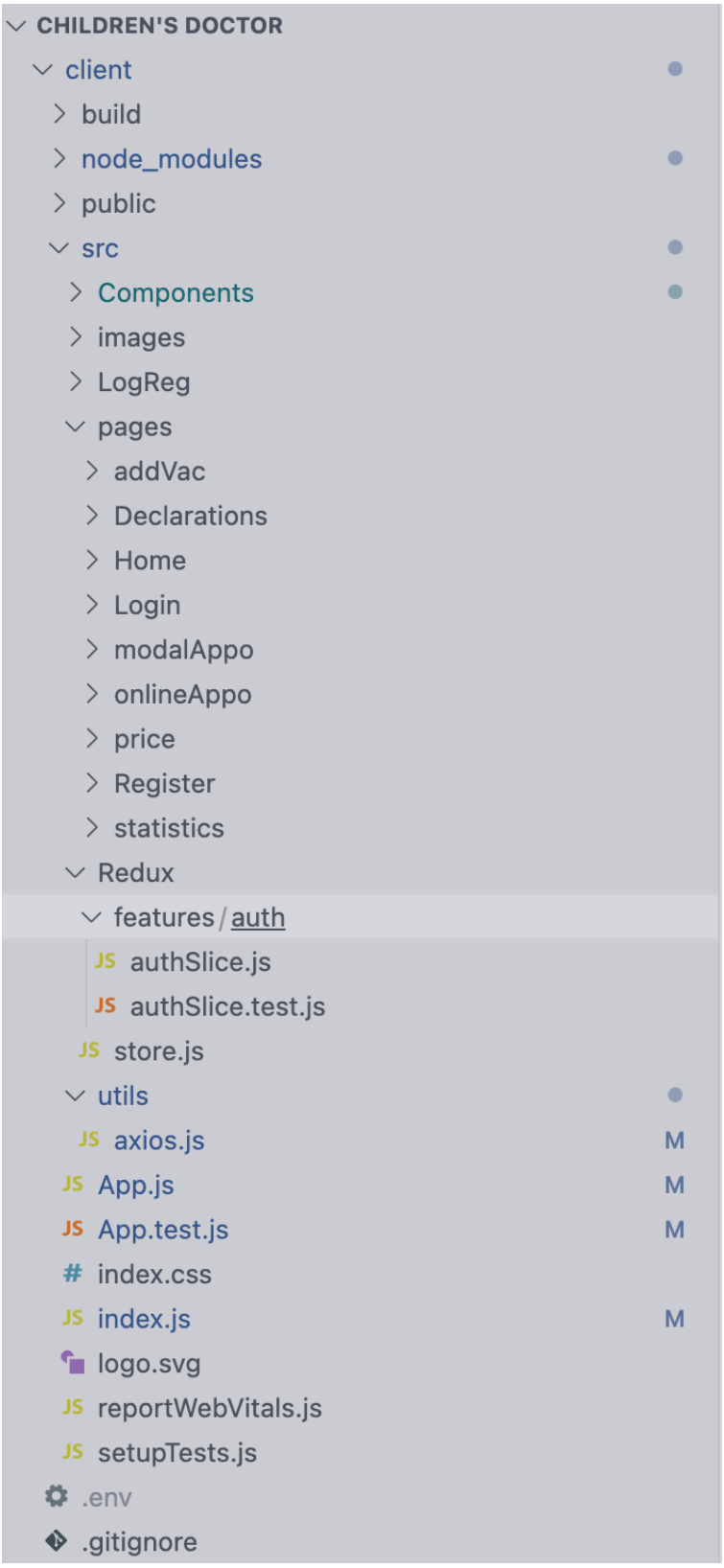
33 Зганич, Поліна Валеріївна. "Дизайн приміщень медіатеки." (2022).

34 Міллер Р. Змінюй простір, змінюй культуру: Як залучення робочого простору веде до трансформації та зростання / Р. Міллер, М. Кейсі, М. Кончар., 2018. – 288 с. – (Wiley). – (978-1118937815).

ДОДАТКИ

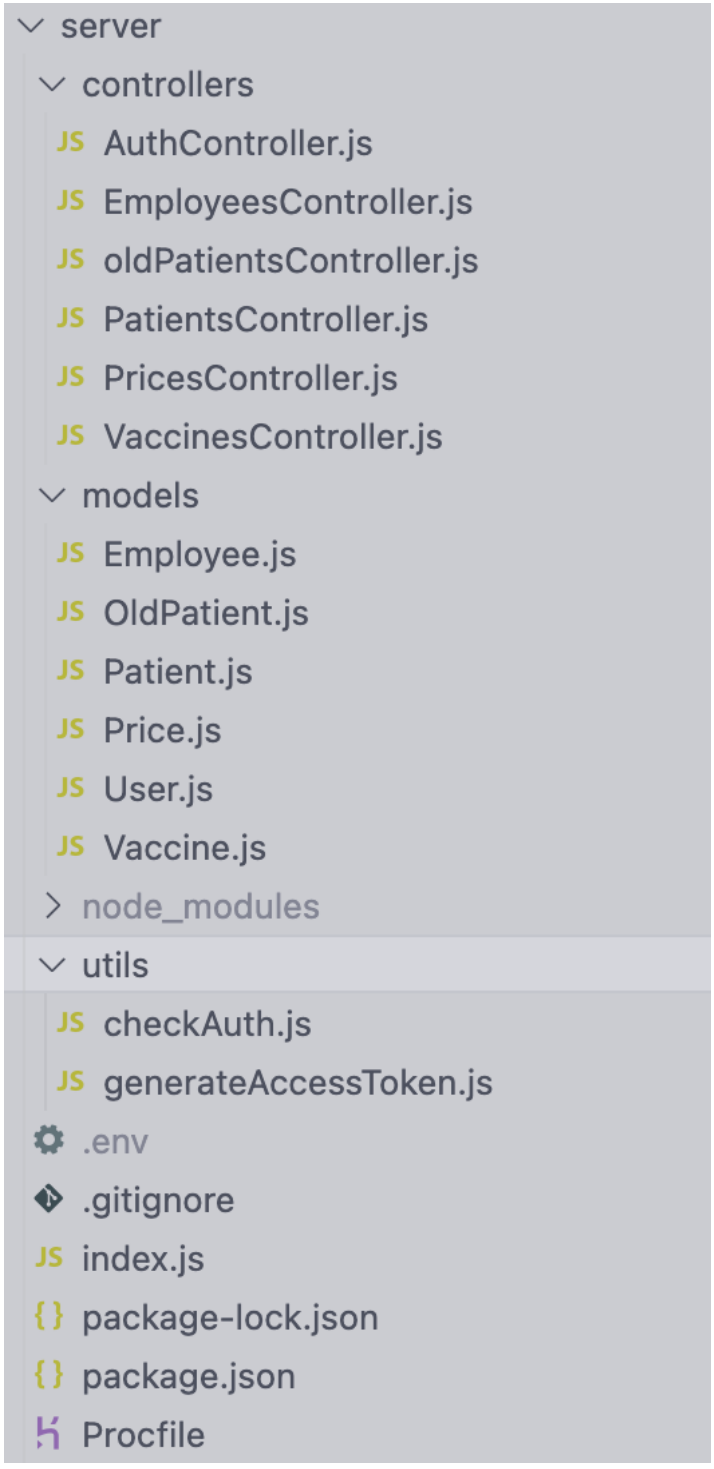
Додаток А

Структура каталогів клієнтської частини



Додаток Б

Структура каталогів серверної частини



Додаток В

Код функції реєстрації нового користувача

```
const registration = async (req, res) => {
     try {
          const errors = validationResult(req);
          if (!errors.isEmpty()) {
               return res.json({
          message: "Логін та пароль повинен бути не менше 4
символів", errors,
          });
     }

if (req.body.username && req.body.password) {
     const { username, password } = req.body;
     const candidate = await User.findOne({ username });
          if (candidate) {
               return res.status(200).json({
               message: "Користувач с таким ім'ям вже існує",
success: false,
          });
     }
const salt = 10;
const hashPass = bcrypt.hashSync(password, salt);
const user = new User({
     username: username,
     password: hashPass,
     role: "user",
});
console.log(user);
await user.save();
return res.status(200).json({
     status: true,
     message: "Користувач успішно створений",
     user: { username: user.username, id: user._id },
          });
     }
} catch (err) {
     return res
          .status(400)
          .json({ success: false, message: "Помилка при реєстрації"
});
     }
};
```

Додаток Д

Код функції логіну нового користувача

```
const login = async (req, res) => {
     try {
          const { username, password } = req.body;
          const user = await User.findOne({ username });

if (!user) {
     return res
          .status(200)
          .json({ success: false, message: "Неправильний логін" });
     }
const      isValidPasword      =      bcrypt.compareSync(password,
user.password);
if (!isValidPasword) {
     return res
          .status(200)
          .json({  success:  false,  message:  "Неправильний  пароль"
});
}

const token = jwt.sign({ id: user._id }, process.env.JWT_KEY, {
     expiresIn: "30d",
});
return res.status(200).json({
     success: true,
     message: "Ви успішно увійшли в систему",
     token,
     user:   {   username:   user.username,   id:   user._id,   role:
user.role },
     role: user.role,
     });
} catch (err) {
     res.json({ success: false, message: "Some error" });
}
};
```

Додаток Е

Код функція для перевірки авторизації користувача

```
const getMe = async (req, res) => {
     try {
          const user = await User.findById(req.userID);

if (!user) {
     return res.json({
          message: "Такий користувач не існує",
     });
}
const token = jwt.sign({ id: user._id }, process.env.JWT_KEY, {
     expiresIn: "30d",
});
return res.json({
     success: true,
     user:  {  username:  user.username,  id:  user._id,  role:
user.role },
     token,
     role: user.role,
     });
} catch (err) {
     res.status(400).json({ success: false });
}
};

module.exports = { login, registration, getMe };
```

Додаток Ж

JS-код функції checkAuth().

```
const jwt = require('jsonwebtoken')

const checkAuth = (req, res, next) => {
     const     token     =     (req.headers.authorization     ||
'').replace(/Bearer\s?/, '');

if(token) {
     try {
          const decoded = jwt.verify(token, process.env.JWT_KEY)

          req.userID = decoded.id;

          next();
     } catch (error) {
          res.status(403).json({
               message: "Користувача не знайдено"
          })
     }

} else {
     res.status(403).json({
          message: "Немає доступу"
          })
     }
}

module.exports = checkAuth
```

Додаток З

Код з файлу AddVac.test.js

```
import { render, screen, fireEvent } from "@testing-
library/react";
import { AddVac } from "./AddVac";
import axios from "axios";
import React from "react";

jest.mock("axios");

describe("VACS TEST", () => {
  let response;
  beforeEach(() => {
    response = {
      data: [
        {
          _id: "6384bd54abc2b04c1b9d72e9",
          title: "Інфлувак",
          price: 670,
          __v: 0,
        },
        {
          _id: "6384bd69abc2b04c1b9d72f2",
          title: "Ротатек",
          price: 1100,
          __v: 0,
        },
        {
          _id: "6384bd8aabc2b04c1b9d72f8",
          title: "Гардасил",
          price: 4800,
          __v: 0,
        },
      ],
    };
  });

  afterEach(() => {
    jest.clearAllMocks();
  });

  test("renders learn react link", async () => {
    axios.get.mockReturnValue(response);
    render(<AddVac />);
    const users = await screen.findAllByTestId("vacList");
    expect(users.length).toBe(3);
    expect(axios.get).toBeCalledTimes(1);
    screen.debug();
  });
});
```

Додаток И

Код з файлу Header.test.js

```
import {render, screen} from '@testing-library/react';
import userEvent from "@testing-library/user-event";
import React from "react";
import Header from "./Header";
import {renderWithRouter} from
"../../tests/helpers/renderWithRouter";

describe('USERS TEST', () => {
    test('test addVac link', async() => {
        render(renderWithRouter(<Header />));
        const usersLink = screen.getByTestId('addVac')
        userEvent.click(usersLink);
        expect(screen.getByTestId('users-
page')).toBeInTheDocument()
    });
    test('test declaration link', async() => {
        render(renderWithRouter(<Header />));
        const aboutLink = screen.getByTestId('declaration')
        userEvent.click(aboutLink);

expect(screen.getByTestId('declaration')).toBeInTheDocument()
    });
    test('test price link', async() => {
        render(renderWithRouter(<Header />, '/price'));
        const mainLink = screen.getByTestId('price')
        userEvent.click(mainLink);

expect(screen.getByTestId('price')).toBeInTheDocument()
    });
})
```