КВАЛІФІКАЦІЙНА РОБОТА

Група ІПЗс-2019
Пахольчук О.Р

2023

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

на правах рукопису

Пахольчук Олег Романович

УДК 004.031.4

Розробка веб-застосунку міської поліклініки для обліку пацієнтів засобами Ruby on Rails

Спеціальність 121 – «Інженерія програмного забезпечення»

Кваліфікаційна робота на здобуття кваліфікації бакалавра

Нормоконтроль

_____Стисло О.В.

(підпис, дата, розшифрування підпису)

Студент

_____Пахольчук О.Р.

(підпис, дата, розшифрування підпису)

Допускається до захисту

Завідувач кафедри

_____к.т.н., доц. Пашкевич О.П.

(підпис, дата, розшифрування підпису)

Керівник роботи

_____к.т.н., доц. Ващишак С.П.

(підпис, дата, розшифрування підпису)

Івано-Франківськ – 2023

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

Освітній ступінь: «бакалавр»

Спеціальність: 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

« ___ » _____ 2023 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Пахольчук Олег Романович

(прізвище, ім’я, по батькові)

1. Тема кваліфікаційної роботи

Розробка веб-застосунку міської поліклініки для обліку пацієнтів засобами Ruby on Rails

керівник роботи:

Ващишак Сергій Петрович, кандидат технічних наук

затверджена наказом вищого навчального закладу від « 11» листопада 2022 року

№ 155/1НВ

2. Термін подання студентом роботи 14.06.2023

3. Вихідні дані роботи: Мова програмування Ruby, HTML, CSS, JavaScript

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)

1. Опис та порівняння веб-сайтів аналогів

2. Проектування архітектури та моделі сайту

3. Програмна реалізація веб-сайту

5. Дата видачі завдання 10.10.2022

КОНСУЛЬТАНТИ РОЗДІЛІВ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Розділ	Консультант (прізвище, ініціали та посада)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Дослідження існуючих аналогів	24.03.2023	Виконано
2.	Проектування архітектури веб-сайту	26.03.2023	Виконано
3.	Розробка веб-сайту	08.05.2023	Виконано
4.	Формування висновків	18.05.2023	Виконано
5.	Оформлення пояснювальної записки	24.05.2023	Виконано
6.	Оформлення графічного матеріалу та підготовка до захисту роботи	01.06.2023	Виконано

Студент _____ Пахольчук О.Р.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Ващишак С.П.
(підпис) (прізвище та ініціали)

Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Сторінка	Опис графічного матеріалу	Сторінка	Опис графічного матеріалу
13	Головна сторінка сайту odkl.if.ua	27	Приклади швидкого написання з Tailwind CSS
14	Навігаційна панель	28	Будова User Story
15	Карта закладів медичної допомоги дітям	34	Логотип СКБД PostgeSQL
16	Головна сторінка сайту okl.if.ua	36	Таблиці у проекті
17	Спеціальні можливості	38	Структура сайту
18	Події БПР на сайтіokl.if.ua	41	Прототип головної сторінки
19	Головна сторінка сайту medics.ua	42	Прототип меню з кількома можливостями сайту
20	Сайтбар після реєстрації на medics.ua	43	Прототип сторінки із всіма лікарями
21	Показники пацієнтів на medics.ua	45	Структура файлів проекту
24	Логотип фреймворку Ruby on Rails	46	Головна сторінка при створенні нового проекту
25	Архітектура проекту по патерну MVC	52	Сторінка із всіма пацієнтами зареєстрованими на сайті

АНОТАЦІЯ

Кваліфікаційра робота присвячена для веб-сайту міської поліклініки, із засобами для обліку пацієнтів., шляхом використання Ruby on Rails.

В першому розділі проведено загально опис вимог, і аналіз здійснивши їх порівння з основними конкурентами. Виділивши їхні переваги та недоліки, порівнюючи їх наповнення та інструментів взаємодії з користувачем.

В другому розділі проведено аналіз здійснивши вибір платформи, методів, та мови програмування , написавши User Stories, що як розробнику дало мені краще зрозуміти потреби користувачів та створити продукт, що відповідає їх очікуванням. Розроблено макет архітектури бази даних, та проектування інтерфейсу веб-сайту.

В третьому розідлі описано процес реалізації веб-сайту, розглядаючи розробку основним частин Ruby on Rails, та технологій які він використовує.

КЛЮЧОВІ СЛОВА: RUBY, RUBY ON RAILS, POSTGESQL, САЙТ ПОЛІКЛІНІКИ.

SUMMARY

The qualification work is dedicated to the website of a city polyclinic and aims to develop tools for patient management using Ruby on Rails.

The first section provides a general description of the requirements and conducts an analysis by comparing them with the main competitors. It highlights their advantages and disadvantages, comparing their content and user interaction tools.

The second section analyzes the selection of platforms, methods, and programming languages by writing User Stories. This helped me, as a developer, better understand the needs of the users and create a product that meets their expectations. The database architecture and website interface design were developed.

The third section describes the process of implementing the website, focusing on the development of the main parts using Ruby on Rails and the technologies it utilizes.

KEYWORDS RUBY, RUBY ON RAILS, POSTGESQL, WEBSITE OF THE POLYCLINIC:

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД	–	бази даних
RoR	–	Ruby on Rails
UI	–	user interface
MVC	–	Model–view–controller
СКБД	–	Система управління базами даних
БПР	–	безперервний професійний розвиток для лікарів
DOM	–	Document Object Model

ВСТУП

Актуальність теми. В епоху цифровізації коли увесь світ розвивається в плані ІТ, все більше організацій та установ переходять до використання веб-застосунків для автоматизації своєї діяльності. Розробка веб-застосунку для міської поліклініки є важливим кроком у поліпшенні якості медичної допомоги та забезпечення комфорту для пацієнтів, особливо в такому прогресивному місту як Івано-Франківськ, в якому вже існують такі веб-сайти, але їх якість та можливості, залишають бажати кращого.

Окрім цього, облік пацієнтів є однією з найважливіших задач у медичній галузі що може стати важливим кроком вперед у розвитку медичної галузі в Івано-Франківську, тому розробка веб-застосунку для його автоматизації може значно полегшити роботу медичного персоналу та забезпечити більш точний та ефективний облік медичних даних пацієнтів. Насамперед, веб-застосунок міської поліклініки може включати такі функції, як реєстрація пацієнтів, збереження медичної інформації, відслідковування історії захворювань та проходження курсів лікування, надання доступу до результатів обстежень та аналізів, покращує процедуру лікування в поліклініці, навіть маючи історію в якій палаті знаходиться пацієнт, а також можливість записатися на прийом до лікаря в режимі онлайн.

Мета і завдання дослідження. Розробка веб-застосунку міської поліклініки для обліку пацієнтів та оптимізації всіх процесів в поліклініці.

Об’єкт роботи – процес оптимізації роботи з наданням медичних послуг в міськоій поліклініці.

Предмет роботи – інформаційна система у вигляді веб-сайту для підвищення якості обслуговування пацієнтів у міській поліклініці.

Завдання роботи. Відповідно до вибраної теми в роботі покладені такі задачі:

- пошук уже існуючих аналогів та їх аналіз;

- вибір мови програмування та технологій розробки;
- розроблення сучасного та зручного дизайну;
- розробка швикдого і безпечного веб-сайту.

Методи роботи. Для вирішення поставленого завдання були використані мова програмування Ruby, фреймворк Ruby on Rails, база даних PostgreSQL та серидовище розробки Visual Studio Code.

Практичне значення отриманих результатів. Результатом виконання кваліфікаційної роботи є створений на Ruby on Rails веб-сайт, призначенням якого є оптимізація всіх робочих процесів в медичних закладах типу міських поліклінік, з швидким записом на прийом, і системою моніторингу за пацієнтами.

Структура роботи. Розділи – 3. Загальний обсяг основної частини – 53 сторінок. Список використаних джерел – 20

РОЗДІЛ 1. ОПИСА ТА ПОРІВНЯННЯ ВЕБ-САЙТІВ АНАЛОГІВ

1.1 Загальний опис вимог до веб-сайту поліклініки

У сучасному світі, інтернет став невід'ємною складовою нашого життя. Люди користуються Інтернетом для отримання різної інформації та послуг, включаючи навіть медичні. Однак з великої кількості веб-застосунків поліклінік багато є недосконалими і мало з них є дійсно хорошими, не кажучи вже про те, що в деяких поліклініках вони відсутні взагалі. Переглянувши існуючі веб-сайти можна помітити, що інтерфейс, функціонал та дизайн цих веб-сайтів вже досить застарілий і не відповідає сучасним вимогам стосовно якості обслуговування пацієнтів як в області лікування, так і в областях документообігу та загальної організації виробничого процесу поліклініки . Тому основною метою кваліфікаційної роботи є розробка веб-застосунку для поліклініки з елементами моніторингу за пацієнтами, такими як їхня медична книга, з курсом їх лікування, або те в якій палаті вони зараз знаходяться, чи які процедури в них призначені за планом.

До сучасного веб-сайту поліклініки поставлено кілька вимог. Перш за все, що впадає в очі кожному користувачу, це UI, а саме правильно підібраний дизайн, який був би простий та зрозумілий, для всіх користувачів в незалежності від їхнього віку. Це особливо буде корисно пацієнтам, при спробі записати на прийом, або на якусь процедуру, а також медичному персоналу, що лікує пацієнтів та документує перебіг лікування. Дизайн повинен бути досить простим, але при цьому вміщати всю можливу для перегляду інформацію. Говорячи про дизайн важко не відмітити інтерфейс який повинен бути кросбраузерним, тобто має підтримуватись на більшості браузерах і пристроях, підлаштовуючись під їх розмірність екрану включно з мобільного адаптацією, що буде дуже корисним, для додаткового комфорту як для пацієнтів, так і для медичного персоналу, збільшуючи їхню ефективність.

Важливо враховувати, що веб-сайт поліклініки повинен бути максимально інформативним і актуальним. Інформація про послуги, лікарів, розклад роботи та контакти має бути обновлюваною і достовірною. Крім того, веб-сайт може мати розділ з частими запитаннями (FAQ), де надається відповіді на найпоширенішіпитання клієнтів.

Загальним призначенням веб-сайту поліклініки є забезпечення зручного та ефективного способу отримання інформації та послуг для клієнтів. Веб-сайт повинен створювати позитивне враження про поліклініку, викликати довіру та забезпечувати легкий доступ до необхідної інформації.

Крім того, веб-сайт поліклініки може мати розділи, присвячені корисній інформації для пацієнтів, наприклад, статті про захворювання, профілактику та загальні поради щодо здорового способу життя. Це допомагає підвищити свідомістьклієнтів про своє здоров'я та стимулює взаємодію з поліклінікою.

У веб-сайту поліклініки також можуть бути розділи, присвячені акціям та знижкам на медичні послуги, що можуть заохочувати клієнтів звертатися до поліклініки.

Останнім, але не менш важливим аспектом веб-сайту поліклініки є забезпечення безпеки та конфіденційності інформації. Веб-сайт повинен мати відповідні заходи захисту даних клієнтів, зокрема за допомогою шифрування та безпечних протоколів передачі даних.

1.2 Огляд та аналіз існуючих аналогів

У сучасному цифровому світі багато поліклінік в Україні визнають важливість присутності в Інтернеті та створюють веб-сайти для забезпечення зручного доступу до інформації про свої послуги та персонал. Однак, з власного досвіду можу сказати, що далеко не всі веб-сайти поліклінік є ефективними та відповідають вимогам сучасних клієнтів. Зі спільних недоліків можна виділити поганий дизайн, та наявність неважливої другорядної інформації, або навпаки, взагалі її відсутність.

Перед початком розробки було проаналізовано існуючі аналоги веб-сайтів поліклінік, де можна виділити наступні:

- *odkl.if.ua*;

- *okl.if.ua;*

- *medics.ua*.

В даному розділі ми пройдемся по кожному аналогу із списку, виділяючи їхні сильні та слабкі сторони.

1. Odkl.if.ua – це веб-сайт обласної дитячої клінічної лікарні [1], вигляд головної сторінки наведено на (рис. 1.1), де зібрана інформація згідно самого закладу, що розпочав свою роботу у 2012 році, але сам заклад був заснований у 1962 році.

Рисунок 1.1 – Головна сторінка сайту odkl.if.ua

Один з найпоширеніших недоліків стосовно поліклінічних сайтів - це застарілий дизайн. Застарілий дизайн може створювати враження, що поліклініка не дотримується сучасних тенденцій і може викликати сумніви щодо якості медичних послуг. Пацієнти можуть вважати, що якщо веб-сайт поліклініки необхідно оновити, то можливо, і самі медичні технології та методи, що використовуються, застарілі. На сайті також відсутня адаптивність,

що ускладнює користування телефонами, чи планшетах, що є важливим аспектом сучасних сайтів.

Серед недоліків нестача зручності, головна сторінка буквально складається з великої кількості інформації зайвої для користувача, яка збиваючи його з пантеликув пошуках потрібно інформації, але разом з цим серед сторінок веб-сайту відсутня базова інформація, про послуги, та процедури.

В хедері веб-сайту розташована навігаційна панель (рис.1.2) яка відкриває доступ до: новин, галереї, медичних закладів, графіку прийому, відгуків, для підлітків та контактів.

Рисунок 1.2 – Навігаційна панель

Рисунок 1.3 – Карта закладів медичної допомоги дітям

Серед цікавих особливостей хочеться виділити сторінку «медичні заклади» в навігаційній панелі, яка складається із ітеративної карти закладів які надають медичну допомогу дітям (рис.1.3), де навівшись на той чи інший район в Івано-Франківській області можна отримати інформацію про районний дитячий медичний заклад.

Наступним недоліком хочеться виділити обмежену функціональність, а саме відсутність можливості онлайн-запису на прийом до лікаря, відсутність онлайн-консультацій або можливості завантаження медичних документів. Один з недоліків, що полягає у відсутності можливості зареєструватись на сайті поліклініки. Це може бути проблемою для користувачів, які бажають отримати доступ до додаткових функцій або послуг, які доступні лише після реєстрації.

2. Okl.if.ua – це сайт обласної клінічної лікарні Івано-Франківської області який був заснований ще в 1949 році [2]. З допомогою архівного проекту в Інтернеті під назвою «wayback machine» [3], який дозволяє переглядати збережені копії веб-сторінок, вдалось дізнатись, що сам сайт бере свій початок ще у 2001 році, і з того часу вже неодноразово повністю мінявся, і на сьогоднішній день виглядає таким чином (рис.1.4).

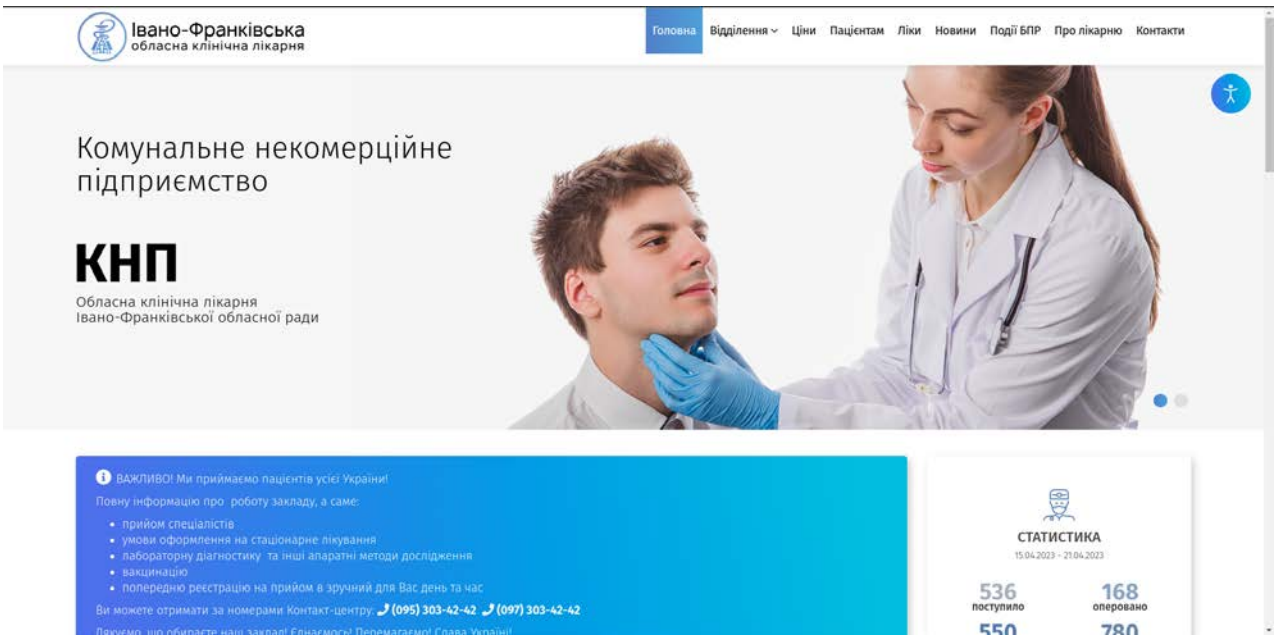

Рисунок 1.4 – Головна сторінка сайту okl.if.ua

Серед переваг над попереднім сайтом хочеться виділити хороший дизайн, який є сучасним, і досить простим на відмінну від попереднього який був забитий великою кількістю непотрібної інформації. Все дуже зрозуміло і зручно як для користувачів так і для медичного персоналу, у веб-сайту також присутня адаптивність під різні мобільні пристрої, планшети, чи що небуть з іншим розширенням екрану. Серед переваг дизайну на сайті доступне вікно з спеціальними можливостями, яке дозволяє міняти дизайн сайту, або навіть розміру тесту (рис.1.5).

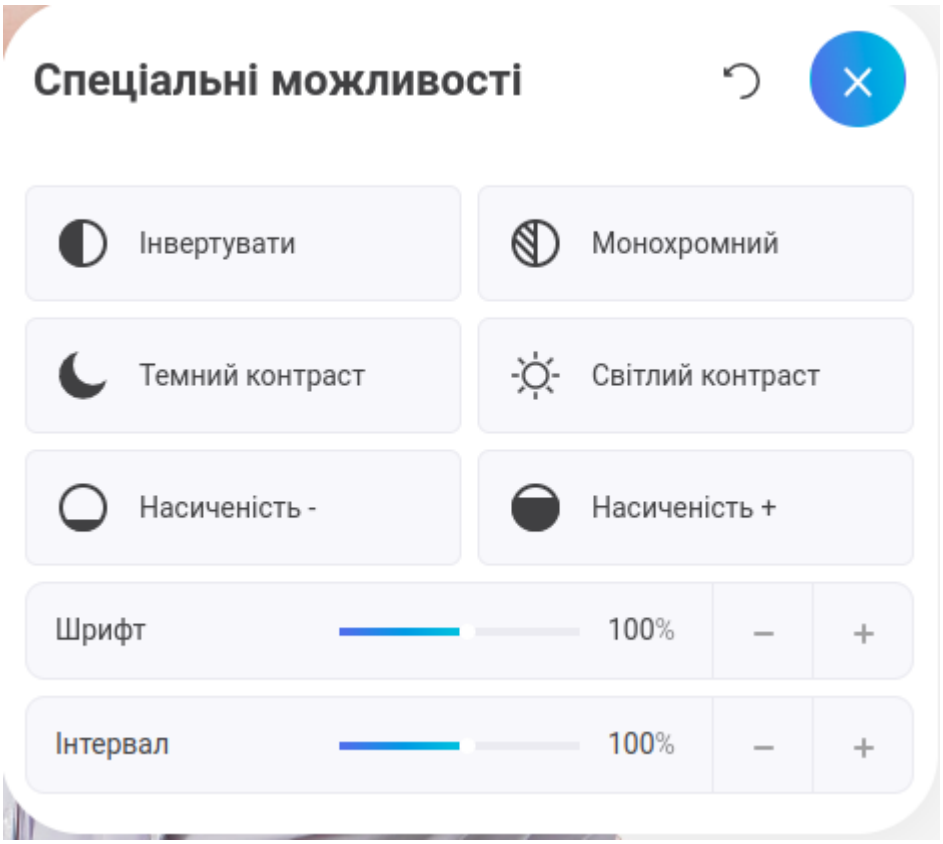

Рисунок 1.5 – Спеціальні можливості

Серед недоліків на сайті також відсутня можливість авторизації, що розбавляє великої кількості можливостей як звичайних користувачів, так і пацієнтів, що є великим недоліком. На сайті присутня сторінка для онлайн запису на прийом, але з недоліків по невідомих причинах вікно не є робочим. З чого випливає і інший недолік стосовно мови на якій написаний сайт, а саме PHP який має досить довгу історію розвитку, що може призводити до деяких

недоліків у самій мові. Вона була спочатку розроблена для роботи з неперервними запитами та обмеженими ресурсами, і це може впливати на її здатність працювати з більш сучасними веб-технологіями. І одним з найбільших недоліків є безпека, PHP має репутацію не надто безпечної мови програмування, оскільки на початку свого розвитку не було враховано безпеку

З переваг хочеться виділити те, що в порівнянні з попереднім сайтом присутня корисна інформація як для самих пацієнтів, так і для лікарів. Для пацієнтів на сайті доступну сторінки з всіма послугами, цінами для них, відділеннями,і сімейними лікарями з гарячою лінією до них. Серед цікавих особливостей якіможна було би використати для власного веб-сайту, для лікарів є дуже корисна сторінка з безперервним професійним розвитком та подіями для лікарів, завдяки якому лікарі мають можливість постійно вдосконалювати свої знання та вміння, а також використовуючи отримані бали для власної атестації, вся інформація доступна на сторінці «події БПР» (рис.1.6).

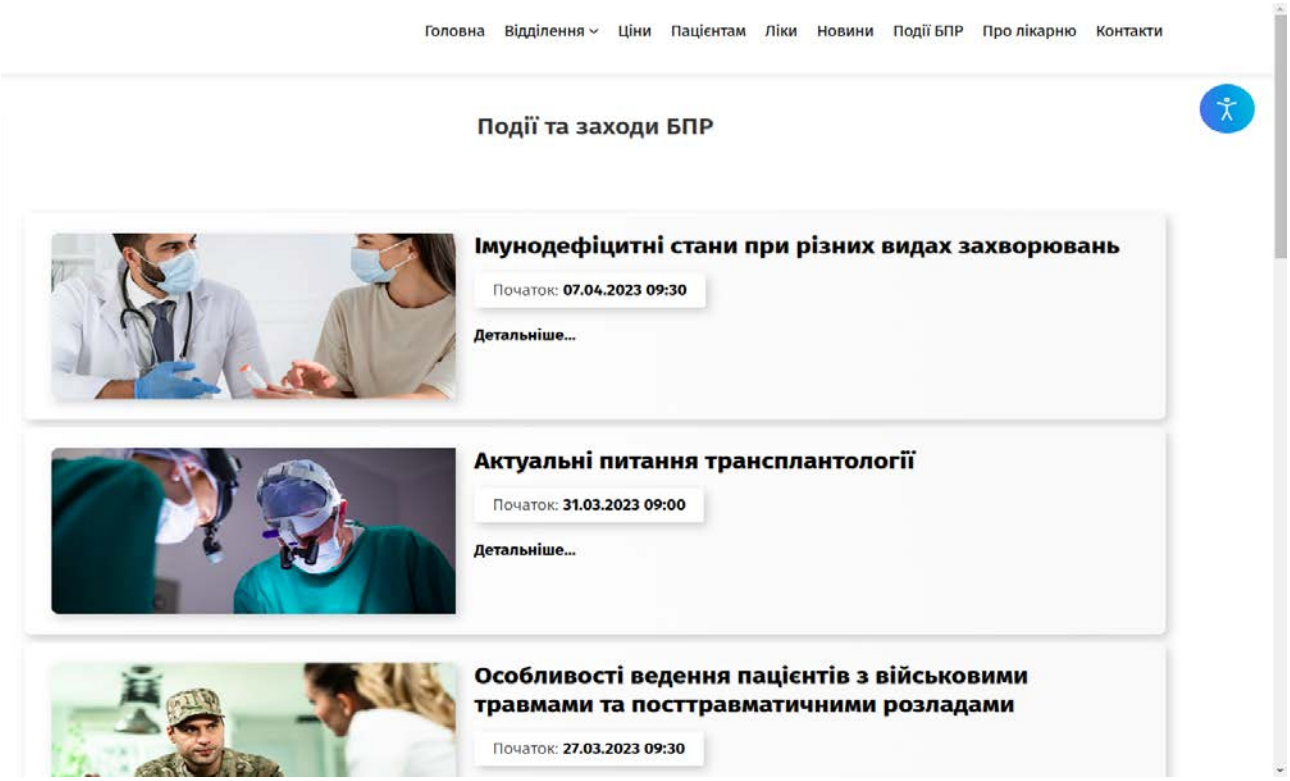

Рисунок 1.6 – Події БПР на сайті okl.if.ua

3. medics.ua – це інтернет-ресурс з допомогою якого ви можете зареєструватися на прийом до лікарів в будь яких областях України, взаємодіяти з ними та переглядати результати призначень у медичній картці, а також оцінки лікарів (рис.1.7). Одним із переваг інтернет-ресурс є кросбраузеним, тобто його спокійно можна відкрити на телефонах, планшетах, чи будь яких інших пристроїв з іншим розширенням екрану, але разом з цим він має мобільний додаток в Google Play, і App Store.

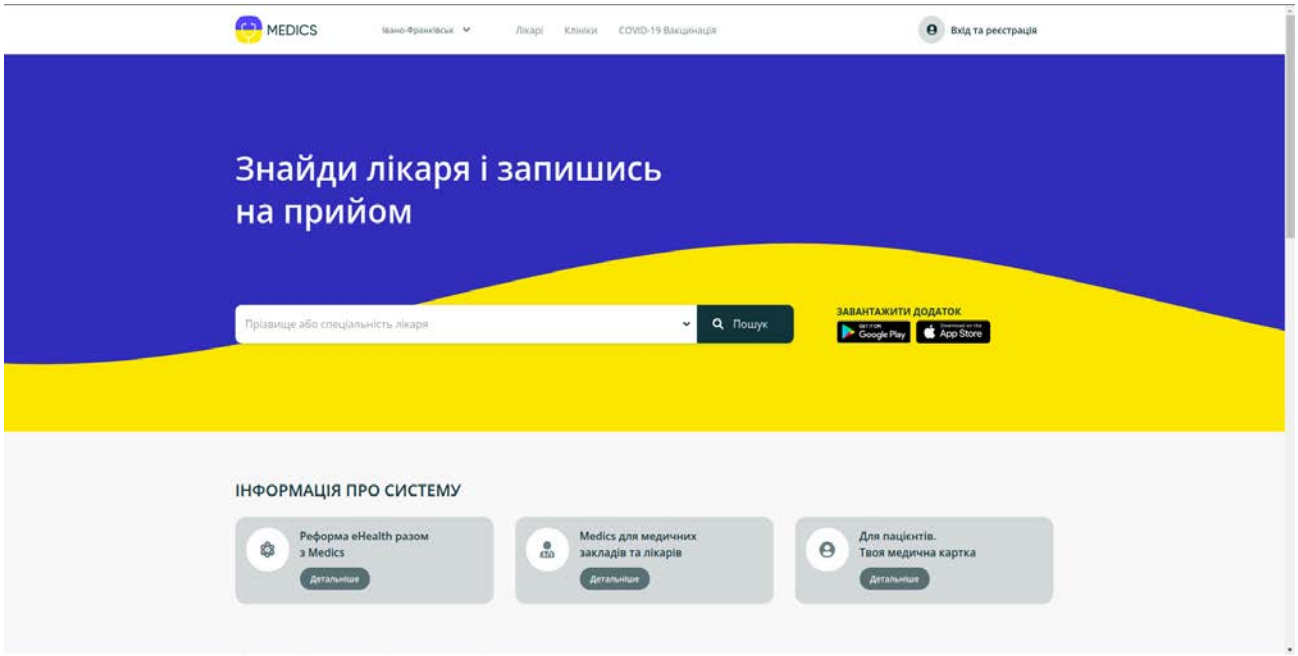

Рисунок 1.7 – Головна сторінка сайту medics.ua

З можливостей веб-сайту важливо відмітити що він надає можливість зареєструвати свій медичний формуляр шляхом заповнення декларації, і долучати персонал до медичного закладу, що є досить зручним, прогресивним і таким, що надає доступ до якісної та ефективної медицини. Таким чином, користуючись пошуком ми можем знайти будь якого лікаря, вибраної нами спеціальності в будь якій області України. Хоч це і є сильного перевагою веб-застосунку, але відкриває проблеми через відсутність великої кількості інформації, так як вибравши якусь поліклініку все, що ми можем зробити, це вибрати до кого на прийом ми хочем записатись із цього закладу, але відсутня інформація про самий заклад, послуги які він надає, ціни на ці самі послуги.

Серед переваг інтернет-ресурсу є дуже стабільна, і інтуїтивна навігація, на відмінну від деяких медичних сайтів які можуть мати заплутану або незручну структуру навігації, що ускладнює користувачам знаходження потрібної інформації. Інформація яку надає сайт є дуже актуальною як постійно оновлюється як стосовно самого сайту, так і медичних закладів які зареєструвались [4].

Веб-сайт має багато переваг. Він дозволяє створити медичну картку для кожного пацієнта, зберігати всі дані про здоров'я в одному місці і забезпечує безпеку інформації. Можна пройти обстеження, приймати ліки, знайти лікаря або медичну послугу. Функція сімейного доступу дозволяє підключити профілі членів сім'ї. Зворотній зв'язок, персональні нагадування та можливість оцінювання допомагають вибрати правильного лікаря. Більшість можливостей можна знайти в сайдбарі сайту (рис.1.8), який може здатися недостатнім, але насправді містить багато сторінок з додатковими можливостями.

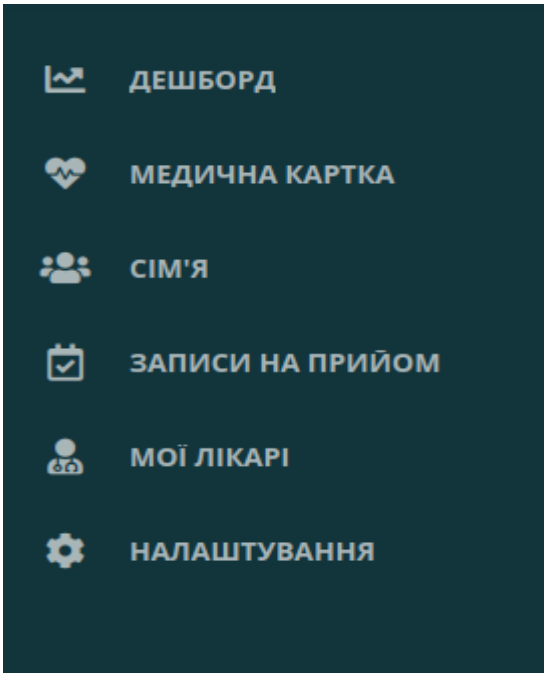

Рисунок 1.8 – Сайдбар після реєстрації на medics.ua

Серед цікавих особливостей веб-сайту хочеться відмітити сторінку з показниками (рис.1.9), що дозволить самому слідкувати за своїм здоров’ям, і

тримати лікаря в курсі самопочуття пацієнта, де лікар матиме можливість бачити навіть температуру пацієнта.

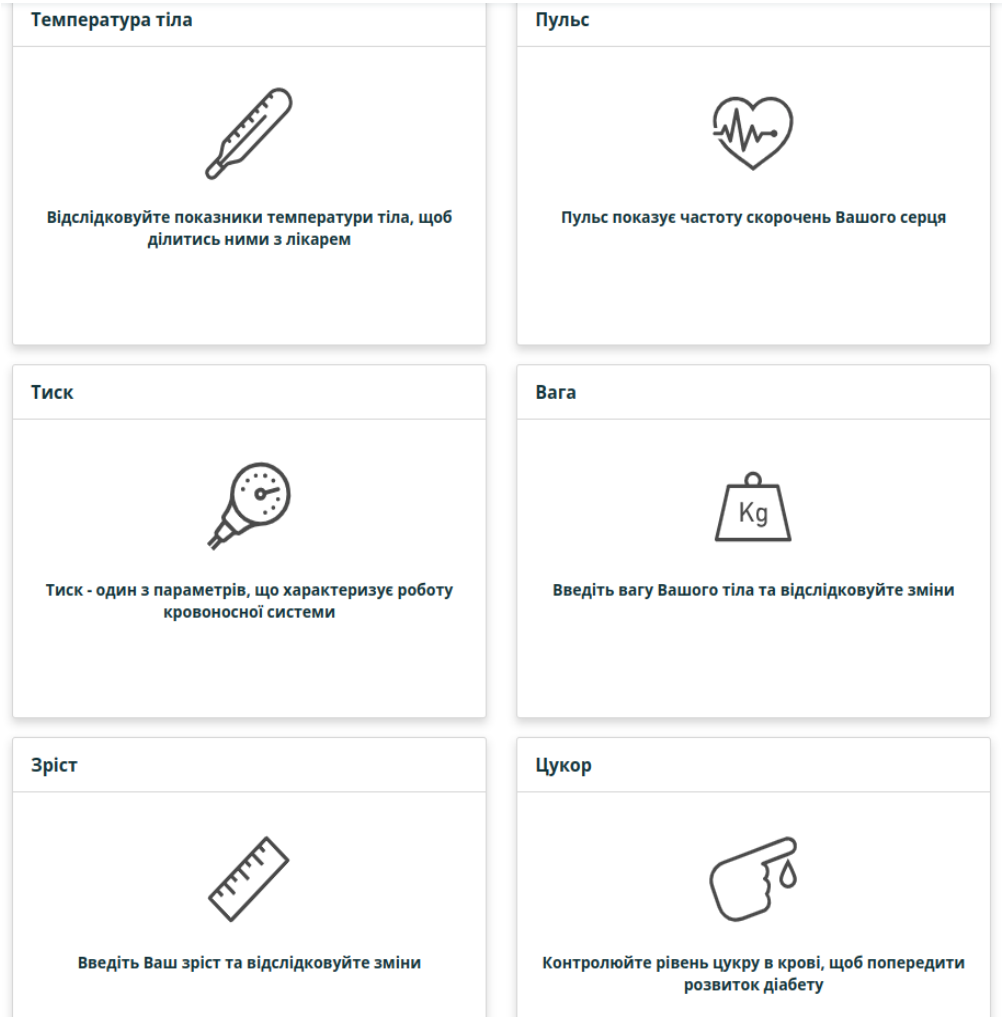

Рисунок 1.9 – Показники пацієнта на medics.ua

Тобто, можливості описаного веб-сайту є доволі потужними.

1.3 Постановка задачі

Згідно вищенаведеного, для покращення якості обслуговування пацієнтів і збільшення ефективності роботи поліклініки необхідно створити веб-сайт поліклініки, за допомогою якого можна було б здійснювати моніторинг пацієнтів, вести медичну справу пацієнта, записуватись на прийом до лікаря,

консультацію онлайн, завантаження аналізів, для оптимізації основних процесів роботи в поліклініці. Для цього необхідно вирішити такі задачі:

- провести аналіз переваг та недоліків існуючих сайтів-аналогів;
- вибрати технології з мовою програмування;
- розробити сучасний, ефективний, та зручний дизайн за стосунку;
- розробити фронтенд і бекенд веб-сайту з використанням технології RoR.

Висновки до розділу 1

В розділі були проведені дослідження та проаналізовані вже існуючі сайти-аналоги, за допомогою яких організовано роботу медичних закладів. Описані загальні можливості та характеристики сайтів медичних закладів. Описані їхні як недоліки так і переваги. В результаті проведених досліджень в подальшій роботі будуть використовуватись опис вищеперерахованого з даного розділу, для запобігання недоліків, та покращення переваг під час проектування системи.

РОЗДІЛ 2. ПРОЕКТУВАННЯ АРХІТЕКТУРИ ТА МОДЕЛІ САЙТУ

2.1 Вибір мови та технології архітектури

У сфері медичних послуг веб-сайт поліклініки виконує важливі функції, забезпечуючи доступ до інформації про медичні послуги, лікарів, графіків роботи, онлайн-запису на прийом та інших сервісів. Враховуючи велику кількість даних, щопотребують управління, а також потенційну необхідність взаємодії з іншими системами (наприклад, системою керування пацієнтами або медичними даними), проект реалізації сайту поліклініки вимагає глибокого розуміння бізнес-процесів і використання потужних інструментів розробки.

Аналізуючи існуючі аналоги веб-сайтів поліклінік, було помічено, що вони часто мають недоліки в розширюваності, продуктивності та безпеці. Деякі з них можуть бути складні у використанні та обслуговуванні, що може призводити до незадоволення користувачів та збитків для бізнесу поліклініки.

Для реалізації сайту поліклініки була обрана мова Ruby з фреймворком до неї Ruby on Rails (рис.2.1) [5], та з допомогою HTML 5, CSS 3, JS.

Основну і найбільшу частину було написано саме на Ruby on Rails. Перш за все, RoR забезпечує продуктивність і швидкість розробки завдяки своїм готовим рішенням для типових задач, таких як маршрутизація, бази даних та автентифікація користувачів. Це дозволяє прискорити процес розробки, скоротити обсяг написаного коду і забезпечити високу продуктивність команди розробників.

Крім того, Ruby on Rails має зрозумілий і простий синтаксис, який легко читається. Це сприяє швидкому освоєнню фреймворку новими розробниками і полегшує підтримку та розширення сайту [6].

Також варто відзначити велике співтовариство розробників Ruby on Rails, що має значний вплив на вибір мови. Існує велика кількість доступних

бібліотек, модулів та розширень, які допомагають будувати функціональність сайту поліклініки.

Крім того, наявність активного співтовариства спрощує отримання допомоги, порад та рішень проблем, що можуть виникнути під час розробки.

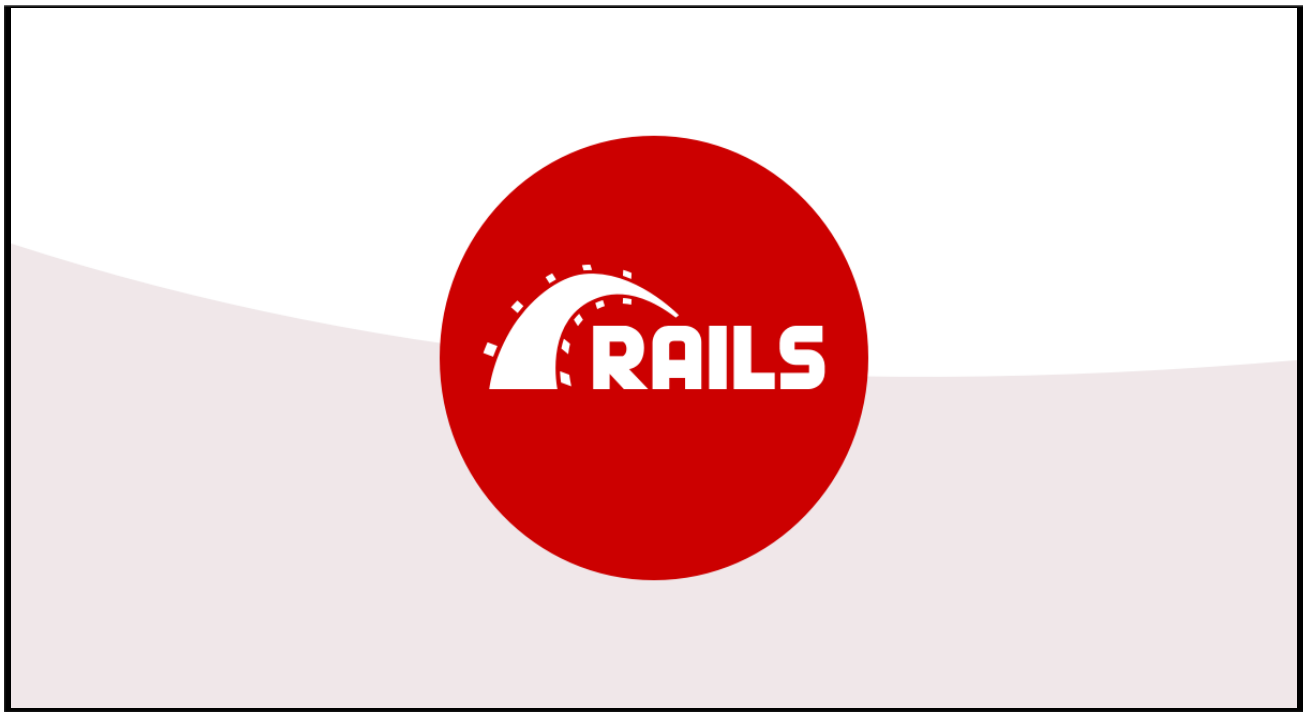

Рисунок 2.1 – Логотип фреймворку Ruby on Rails

Ruby on Rails також відомий своєю надійністю та безпекою. Він надає вбудовані механізми для захисту веб-додатків від SQL-ін'єкцій і міжсайтового скриптингу (XSS). Постійна підтримка та оновлення в рамках співтовариства розробників також гарантують безпеку та стабільність сайту поліклініки.

Крім цього, Ruby on Rails має вбудовані засоби для підтримки іншого важливого аспекту веб-додатків - масштабованості. Фреймворк надає зручні інструменти для розширення функціональності, оптимізації продуктивності та розподілення навантаження, що є особливо важливим для сайту поліклініки, який може мати значну кількість відвідувачів та великий обсяг даних.

Одним із важливих пунктів використання RoR є його архітектура, а саме на паттерні проектування Model-View-Controller (MVC) [7]. Цей підхід розбиває

додаток на три основні компоненти на (рис. 2.1), що сприяє кращій організації коду та полегшує його розуміння і підтримку в майбутньому.

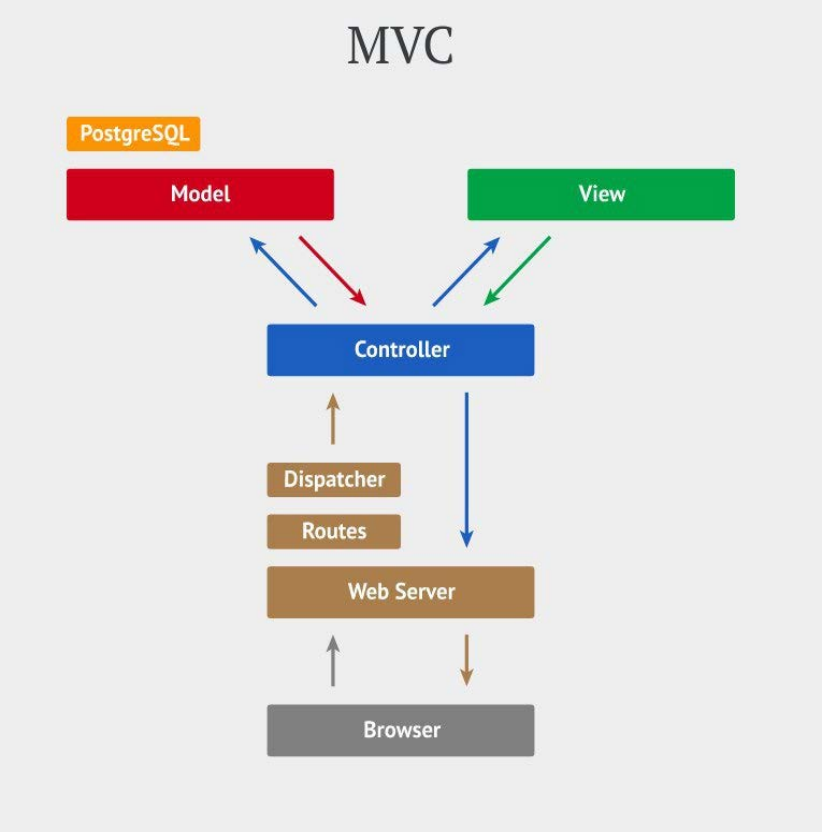

Рисунок 2.2 – Архітектура проекту по патерну MVC

Модель (Model) відповідає за управління даними додатку. Вона включає логіку доступу до бази даних, операції збереження, оновлення, видалення даних та виконання запитів до бази. Модель відображає структуру та взаємозв'язки даних у додатку, такі як пацієнти, лікарі, медичні записи тощо. Вона забезпечує безпеку та цілісність даних та може містити логіку бізнес-правил.

Представлення (View) відповідає за відображення даних користувачу. Його завдання - відобразити інформацію з моделі в зручному для користувача форматі. Представлення можуть бути HTML-шаблонами, які відображають дані, форми вводу для користувача або інші компоненти інтерфейсу. Вони відокремлюють представлення від бізнес-логіки, що дозволяє змінювати зовнішній вигляд без впливу на роботу моделі та контролера.

Контролер (Controller) відповідає за обробку запитів користувача та управління потоком даних у системі. Він приймає запити від користувача, виконує необхідну логіку та спілкується з моделлю та представленням. Контролер вирішує, які дані потрібно відобразити користувачу, які дії виконати при отриманні запиту та які дані передати в модель для збереження чи оновлення. Він реагує на події від користувача та координує взаємодію моделі та представлення.

Ця поділена на компоненти архітектура дозволяє розділити відповідальності між різними частинами системи. Вона сприяє модульності, розширюваності та повторному використанню коду. Зміни в одному компоненті не впливають на решту системи, що полегшує розробку, тестування та підтримку проекту. Крім того, цей підхід дозволяє розробникам працювати паралельно над різними частинами проекту.

MVC є популярним підходом для веб-розробки на Ruby on Rails через його простоту та ефективність. Він надає зручну структуру для розробки та підтримки веб-сайту поліклініки, дозволяючи розбити його на логічні компоненти та забезпечити гнучкість та розширюваність системи.

Для взаємодії з сайтом, використовується RESTful API (Representational State Transfer), що дозволяє здійснювати запити через HTTP-протокол, такі як GET, POST, PUT та DELETE, що забезпечує гнучкість та ефективність взаємодії з системою.

У розробці веб-сайту поліклініки на Ruby on Rails, для стилізації та дизайну інтерфейсу, буде використаний окрім CSS, один з популярних фреймворік до нього Tailwind CSS, який надає готові компоненти та класи для швидкої розробки і привабливого вигляду веб-сайту [8]. Завдяки своїй філософії "utility-first", Tailwind CSS дозволяє розробникам легко стилізувати елементи інтерфейсу, використовуючи класи замість написання власного CSS. Це полегшує роботу зі стилями, прискорює процес розробки та забезпечує єдність стилів на всьому сайті (рис.2.3).

Таким чином, використання Tailwind CSS у розробці сайту поліклініки на Ruby on Rails дозволяє швидко і зручно стилізувати інтерфейс, забезпечувати єдність стилів та пристосовувати дизайн до потреб проекту. Це сприяє швидкому розвитку та покращує візуальний вигляд веб-сайту.

Варто зазначити, що Ruby on Rails має добре розвинену систему тестування. Це дозволяє проводити автоматизоване тестування коду, забезпечуючи високу якість та надійність роботи сайту поліклініки. Тести допомагають виявляти та усувати помилки на ранніх етапах розробки, що сприяє підвищенню стабільності та ефективності сайту.

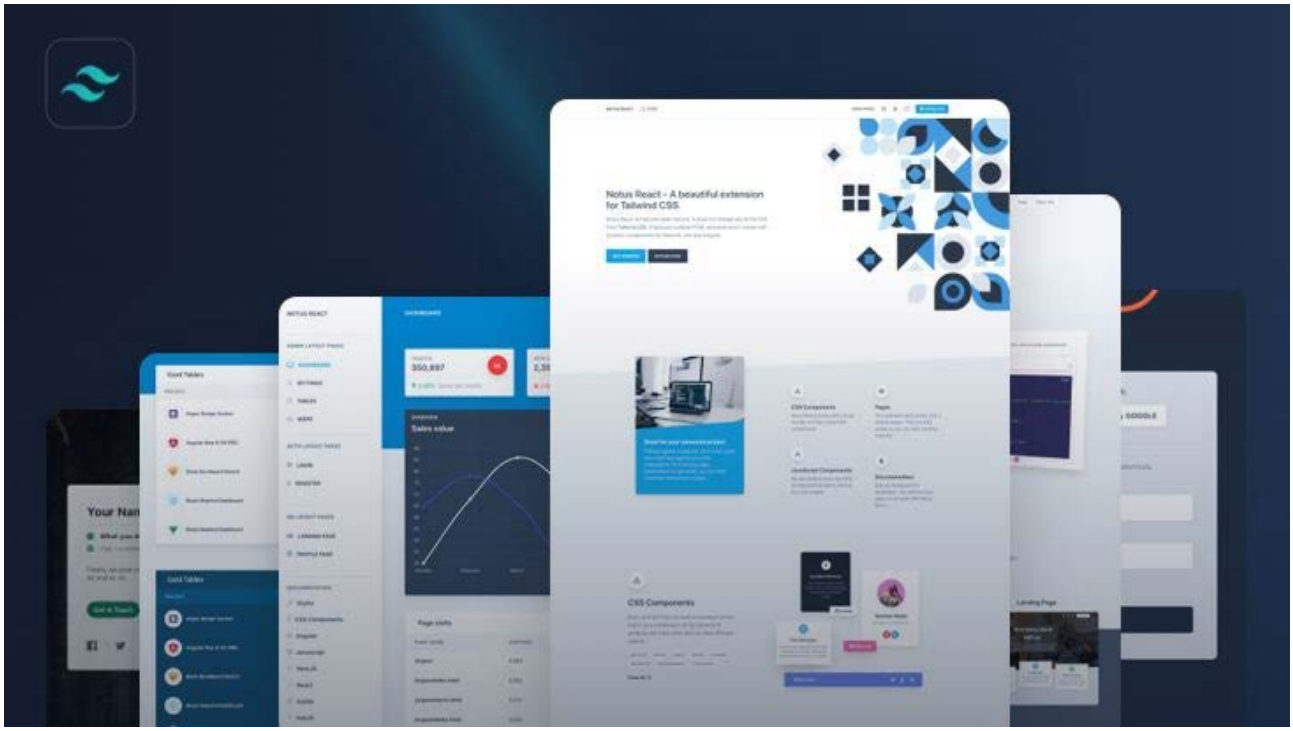

Рисунок 2.3 – Приклади швидкого написання з Tailwind CSS

2.2 Написання User Stories

У процесі розробки сайту поліклініки на Ruby on Rails я активно використовував підхід з написання User Stories [9]. Цей підхід, що базується на Agile-методології розробки програмного забезпечення, дозволяє мені, як розробнику, краще розуміти потреби користувачів та створювати продукт, що відповідає їх очікуванням.

Підхід з написання User Stories базується на коротких, простих описах функціональності системи з точки зору кінцевого користувача. Їх основна мета полягає у створенні чіткого розуміння того, як система повинна працювати та які результати повинні бути досягнуті для задоволення потреб користувачів. Кожна User Story містить короткий опис бажаної функціональності «What?», актора, якому ця функціональність потрібна «Who?“, та корисність, яку вона надає “Why?“ (рис.2.4).

Рисунок 2.4 – Будова User Story

Використання User Stories в процесі розробки сайту поліклініки дозволило мені приступити до реалізації функціоналу, який найкращим чином відповідає потребам користувачів. Кожна User Story стала основою для створення відповідних моделей, контролерів та представлень, які забезпечують необхідний функціонал сайту поліклініки.

Підхід з написання User Stories було важливим етапом у реалізації сайту поліклініки на Ruby on Rails. Він допоміг мені, як розробнику, краще зрозуміти потреби користувачів та створити продукт, що відповідає їх очікуванням.

Ось кілька User Stories які були написані при розробці сайту, для всіх користувачів:

1. Як новий пацієнт, хочу мати можливість зареєструватися на сайті поліклініки, щоб мати доступ до моїх медичних записів та записуватися на прийоми.

2. Як пацієнт, хочу мати можливість переглядати свої медичні записи, щоб бути в курсі свого стану здоров'я та результатів обстежень.

3. Як пацієнт, хочу мати можливість записатися на прийом до лікаря онлайн, щоб уникнути черг та забезпечити зручність у виборі часу.

4. Як пацієнт, хочу мати можливість скасувати запис на прийом, якщо виникла необхідність змінити дату або час, щоб звільнити час лікаря та надати можливість іншим пацієнтам записатися на цей час.

5. Як пацієнт, хочу отримувати сповіщення та нагадування про наближення мого прийому або необхідності проведення додаткових обстежень.

6. Як пацієнт, хочу мати можливість швидко знайти лікаря з певною спеціалізацією або іменем, щоб здійснити запис на прийом до певного фахівця.

7. Як пацієнт, хочу мати можливість переглянути список доступних медичних послуг, щоб визначити, які процедури або обстеження можна отримати в поліклініці.

8. Як пацієнт, хочу мати можливість замовити лікарські рецепти онлайн, щоб уникнути додаткових візитів до поліклініки та отримати необхідні медикаменти.

9. Як пацієнт, хочу мати можливість залишити оцінку та відгук про лікаря, щоб допомогти іншим користувачам при виборі медичного фахівця.

10. Як пацієнт, хочу мати можливість змінити свої персональні дані, такі як адреса, контактна інформація або страховий номер, щоб забезпечити актуальність моїх медичних записів.

11. Як лікар, хочу мати можливість переглянути свій розклад роботи на сайті поліклініки, щоб бути в курсі призначених прийомів та організувати свій робочий день.

12. Як лікар, хочу мати можливість переглядати та керувати списком моїх пацієнтів, записаних на прийом, щоб забезпечити організований та ефективний прийом.

13. Як лікар, хочу мати можливість переглядати та оновлювати електронну медичну книгу пацієнта, щоб мати актуальну інформацію про їх медичну історію та рекомендації.

14. Як пацієнт, хочу мати можливість поділитись доступом до своєї електної медичної книги, щоб мій лікар мав актуальну інформацію про мій стан здоров'я.

15. Як лікар, хочу мати можливість записувати результати обстежень та лабораторних досліджень пацієнтів у їх електронні медичні картки, щоб забезпечити централізоване зберігання та доступ до цієї інформації.

16. Як лікар, хочу мати можливість звернутися до інших лікарів на сайті поліклініки для консультації щодо складних медичних випадків або отримання допомоги при прийнятті рішень.

17. Як лікар, хочу мати можливість переглядати медичні директиви, протоколи та оновлення в галузі моєї спеціалізації, щоб залишатися інформованим про найновіші медичні стандарти та рекомендації.

18. Як лікар, хочу мати можливість організовувати лікарські зустрічі, семінари та навчальні заходи на сайті поліклініки, щоб сприяти професійному розвитку та обміну знаннями з колегами.

19. Як лікар, хочу мати можливість переписати пацієнта з однієї палати в іншу, щоб забезпечити пацієнта потрібною палатою з курсом лікування.

20. Як адміністратор, я хочу мати можливість створювати нові акаунти лікарів, щоб дати їм доступ до системи та дозволити їм керувати пацієнтами.

21. Як адміністратор, я хочу мати можливість додавати нових лікарів до різних медичних спеціалізацій, щоб пацієнти могли легко знайти та обрати потрібного лікаря.

22. Як адміністратор, я хочу мати можливість переглядати та змінювати інформацію про лікарів, таку як контактні дані, графік роботи та фахові навички, щоб забезпечити актуальну інформацію на сайті.

23. Як адміністратор, я хочу мати можливість створювати розклади прийому для кожного лікаря, щоб пацієнти могли записуватись на прийом онлайн.

24 Як адміністратор, я хочу мати можливість переглядати та змінювати інформацію про пацієнтів, таку як контактні дані, медичні записи та історія відвідувань, щоб забезпечити повну і актуальну базу даних.

25. Як адміністратор, я хочу мати можливість встановлювати привілеї доступу для різних користувачів, щоб забезпечити безпеку даних та обмежити доступ до конфіденційної інформації.

26. Як адміністратор, я хочу мати можливість генерувати звіти про роботу поліклініки, такі як звіти про надходження, витрати, кількість пацієнтів тощо, щоб мати контроль над фінансовою та статистичною інформацією.

27. Як адміністратор, я хочу мати можливість керувати лікарськими кабінетами та медичним обладнанням, щоб забезпечити їх ефективне використання та планування обслуговування.

28. Як адміністратор, я хочу мати можливість створювати та керувати списками послуг, що надаються поліклінікою, з можливістю зміни цін та опису послуг.

29. Як адміністратор, я хочу мати можливість керувати системою нагадувань пацієнтам про прийоми та інші медичні заходи, щоб забезпечити вчасну та точну інформацію.

30. Як адміністратор, я хочу мати можливість переглядати та керувати журналом лікарських записів, щоб забезпечити доступ до повної та актуальної інформації про лікування пацієнтів.

31. Як адміністратор, я хочу мати можливість створювати та керувати групами пацієнтів, наприклад, для проведення спеціалізованих операцій або медичних програм, щоб краще організувати та надавати послуги.

32. Як адміністратор, я хочу мати можливість керувати запланованими медичними заходами, такими як операції, обстеження та консультації, щоб забезпечити їх ефективне планування та ресурси.

33. Як адміністратор, я хочу мати можливість генерувати звіти про використання медичних ресурсів, наприклад, лікарських препаратів, обладнання та ліжок, щоб забезпечити контроль та оптимізацію їх використання.

34. Як адміністратор, я хочу мати можливість керувати системою замовлення медичних матеріалів та інвентарю, щоб забезпечити наявність необхідних ресурсів та запобігти дефіциту.

35. Як адміністратор, я хочу мати можливість встановлювати та керувати робочим графіком лікарні, включаючи відпустки та зміни, щоб забезпечити належне функціонування поліклініки.

36. Як адміністратор, я хочу мати можливість надсилати повідомлення та оголошення до лікарів та іншого персоналу, щоб ефективно спілкуватись та надавати важливу інформацію.

37. Як адміністратор, я хочу мати можливість переглядати та керувати системою підтримки, де користувачі можуть звертатись зі своїми запитами та проблемами, щоб забезпечити вчасне та якісне вирішення їх потреб.

Ці user stories допомагають створити функціональну та зручну адміністративну панель, що дозволяє адміністраторам ефективно керувати персоналом поліклініки та пацієнтами.

Завдяки можливостям створення та керування акаунтами лікарів, адміністратор може забезпечити доступ до системи та надати лікарям можливість керувати пацієнтами. Керування інформацією про лікарів, включаючи контактні дані, графіки роботи та фахові навички, дозволяє забезпечити актуальну інформацію на сайті. Створення розкладів прийому для лікарів сприяє зручному онлайн-запису пацієнтів на прийом. Керування інформацією про пацієнтів, їх контактними даними, медичними записами та історією відвідувань, дозволяє підтримувати повну та актуальну базу даних.

2.3 Архітектура бази даних

Архітектура бази даних, і вибір правильно бази даних є важливою складовою будь-якого веб-додатку особливо в реалізації сайту поліклініки. Одна з головних причин полягає у тому, що поліклініка зберігає значну кількість медичних даних про пацієнтів, призначення ліків, результати обстежень тощо.

Отже, база даних повинна бути здатною зберігати, керувати та надійно захищати ці дані.

При створенні нового проекту в RoR, за замовчування використовується SQLite. Однак, для реалізації сайту поліклініки на Ruby on Rails була обрана реляційна база даних PostgreSQL (рис.2.5) по ряду причин. PostgreSQL відома своєю надійністю, масштабованістю та підтримкою складних запитів і зв'язків між даними, яка широко використовується в індустрії [10]. Вона також надає розширення для географічних даних, що може бути корисним для розміщення даних про місцезнаходження поліклініки та інших об'єктів.

Хоча PostgreSQL є потужним вибором, одним з найпопулярніших аналогів PostgreSQL є MySQL. MySQL також є реляційною БД, яка має широку підтримку і велику спільноту розробників. Вона часто використовується у веб-розробці, включаючи проекти на Ruby on Rails. MySQL надає хороші швидкодіючі можливості та можливості масштабування, але вона може бути менш потужною у порівнянні з PostgreSQL в деяких аспектах. Іншим популярним вибором є MongoDB, яка відноситься до категорії NoSQL баз даних. MongoDB зберігає дані у форматі JSON-подібних документів, що дозволяє гнучко зберігати та опрацьовувати дані без потреби використовувати фіксовану схему. Вона зазвичай використовується для проектів, де потрібно ефективно зберігати та обробляти великі обсяги неструктурованої інформації. Іншим аналогом є SQLite, який є легким та вбудовуваним реляційним двигуном бази даних, який не вимагає окремого сервера для роботи. Вона зберігає всю базу даних у одному файлі, що полегшує розгортання та управління базою даних. SQLite підтримує мову SQL та має достатні функціональні можливості для багатьох веб-проектів. Однак, в порівнянні з PostgreSQL, SQLite має свої обмеження. Вона не підтримує деякі розширені можливості, такі як реплікація даних, масштабування та паралельне виконання запитів. Також, SQLite використовує менш потужні алгоритми оптимізації та має обмежену підтримку конкурентного доступу до бази даних.

Рисунок 2.5 – Логотип СКБД PostgreSQL

У кожної з цих альтернатив є свої переваги та недоліки, і вибір залежить від конкретних потреб проекту. У випадку сайту поліклініки на Ruby on Rails, PostgreSQL була обрана через свою надійність, підтримку RoR та розширені можливості для оптимізації, масштабування даних та розширення для географічних даних та популярність у спільноті розробників.

У Ruby on Rails для оновлення структури бази даних використовуються міграції, які представляють собою файлові скрипти, що описують зміни у структурі бази даних, такі як створення таблиць, додавання або видалення стовпців тощо. Міграції забезпечують контроль версій структури бази даних і дозволяють легко виконувати оновлення на живому сайті без необхідності вручного втручання [11]. Міграції автоматично відстежуються та зберігаються у вигляді файлів в папці db/migrate вашого проекту. Кожна міграція має унікальне ім'я та включає методи, які визначають необхідні зміни структури бази даних. Наприклад, для створення нової таблиці можна використовувати метод create_table, для додавання стовпця - метод add_column, а для видалення таблиці або стовпця - метод drop_table або remove_column.

Приклад готової міграції з користувачами створеного з домогою гему Devise [12], для швидкого написання авторизації, та реєстрації для користувачів:

```
class DeviseCreateUsers < ActiveRecord::Migration[6.1]
  def change
    create_table :users do |t|
      t.string :email,              null: false, default: ""
      t.string :encrypted_password, null: false, default: ""
      t.string   :reset_password_token
      t.datetime :reset_password_sent_at
```

```
      t.datetime :remember_created_at
      t.timestamps null: false
    end
    add_index :users, :email,                unique: true
    add_index :users, :reset_password_token, unique: true
  end
end
```

З допомогою використання гему Redis який використаний в проекті, він надає повний набір сховищ [13]. Асоціації є одним з ключових аспектів архітектури бази даних для сайту поліклініки на RoR. Вони дозволяють встановлювати зв'язки між різними моделями бази даних, що використовуються у додатку. Асоціації реалізуються за допомогою спеціальних методів та декларативних визначень в моделях. Вони дозволяють визначати тип зв'язку між моделями, такі як "один до одного", "один до багатьох" та "багато до багатьох". За допомогою асоціацій, ви можете здійснювати звернення до пов'язаних моделей, отримувати та змінювати пов'язані дані.

У Ruby on Rails взаємодія з базою даних реалізована за допомогою ActiveRecord, що є частиною фреймворку Rails. ActiveRecord надає зручний та експресивний спосіб взаємодії з базою даних, спрощуючи роботу з моделями і таблицями. Якщо створити модель у Rails, вона автоматично співставляється з таблицею у базі даних. ActiveRecord використовує конвенцію назв таблиць та стовпців, що дозволяє зменшити кількість конфігураційних налаштувань. Наприклад, якщо є модель "Patient", то ActiveRecord очікуватиме таблицю з назвою "patients", і можна виконувати різноманітні запити до бази даних, такі як вибірка даних, фільтрація, сортування, групування тощо. ActiveRecord автоматично генерує та виконує SQL-запити, забезпечуючи високу продуктивність та безпеку.

У проекті було реалізовано структуру БД у draw.io [14]. На (рис. 2.6) зображення частина робочої бази даних, яка досі ще розширяється новим функціоналом, але вже є робочою представляючи основні сутності, в роботі , і дозволяє записуватись на прийом, займатись моніторингом пацієнтів,

переглядати як звичайно новини, так і новини для БПР, надавати послуги, редагувати медичну книгу.

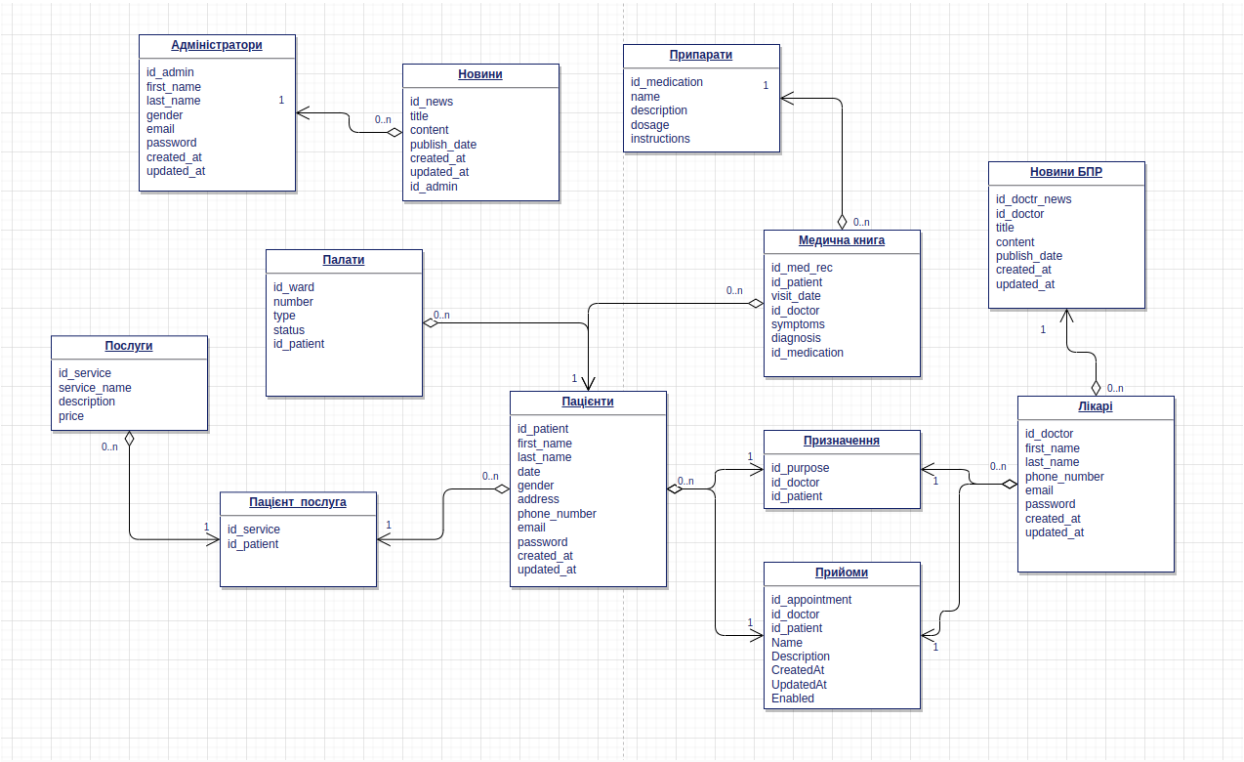

Рисунок 2.6 – Таблиці у проекті

В цілому, правильно спроектована архітектура бази даних в Ruby on Rails для сайту поліклініки допомагає забезпечити ефективне управління даними, зручну роботу з ними та підтримку функціональності, необхідної для ефективного функціонування поліклініки.

2.4 Проектування інтерфейсу

Проектування інтерфейсу є критично важливою частиною розробки програмного продукту або системи. Врахування потреб та очікувань користувачів, структурування інформації, вибір елементів керування та використання зручних графічних компонентів допомагають створити зручний, ефективний та задовольняючий інтерфейс для користувачів. Регулярне

тестування та вдосконалення інтерфейсу дозволяє покращувати його якість та відповідність потребам користувачів.

Загальна структура сайту із зв’язками і переходами між сторінками зображена на (рис. 2.7). З допомогою хедера на головній сторінці, в якому буде розміщуватись більшість посилань сайту можна буде одразу на потрібно сторінку.

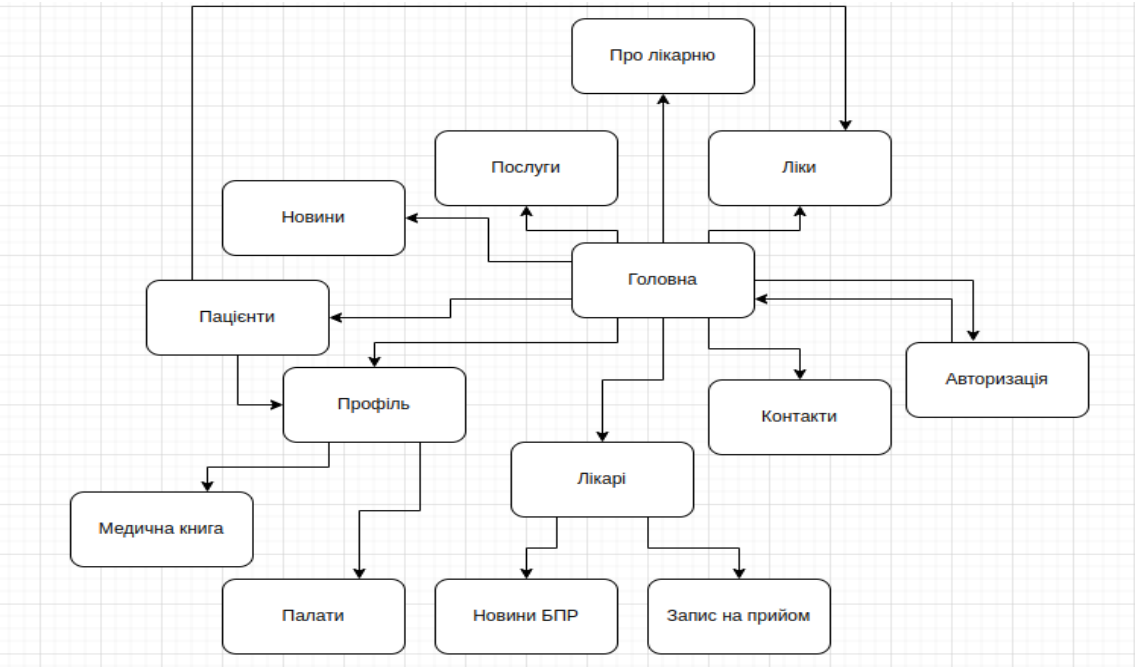

Рисунок 2.7 – Структура сайту

На сторінці лікарі, є список із всіх лікарів медичного закладу, з короткою інформацію про них, та їхнім рейтингом, де після вибраного лікаря, відкриється швидкий перехід запису на прийом, календарем з вільними датами та годинами. Але для користувачів з роллю лікар, буде доступна сторінка із їхніми пацієнтами, де будуть всі їхні дані в медичній книзі, з курсом лікування, рецептом, хворою, температурою пацієнта в різний час, і інші аналізи, разом з спеціальною можливість перевести пацієнта в якусь палату, якщо він лягає в лікарню. Як у пацієнтів так і у лікарів, є система сповіщення, про прийом, або нагадування про нього, у пацієнтів нагадування може працювати так само і на ліки. Після прийому пацієнта, і лікаря з’являється

можливість виписати рецепт пацієнту, так само і електронному форматі. Якщо запросити в адресній стоці неіснуючі сторінку, то веб-сайт буде видавати відповідне повідомлення про помилку.

Використовуючи аналіз із першого розділу можна ефективно розробити інтерфейс веб-сайту. Створення скетчів та прототипів є важливим етапом у процесі проектування користувацького інтерфейсу. Це дозволяє візуалізувати ідеї та концепції інтерфейсу та перевірити їх перед реалізацією.

Скетчі використовуються для швидкого наброскування концепцій та ідей інтерфейсу. Вони можуть бути створені вручну за допомогою паперу та олівця або за допомогою спеціальних програм для створення скетчів яких в інтернеті досить. Скетч дозволяє зосередитися на структурі, розміщенні елементів та логіці взаємодії без деталізації.

Прототипи є більш деталізованими варіантами інтерфейсу, які можуть бути інтерактивними. Вони можуть бути створені за допомогою спеціального програмного забезпечення для прототипування, які дозволяють створювати взаємодію з різними елементами інтерфейсу. Прототипи дозволяють випробувати функціональність інтерфейсу, перевірити його поток роботи та збортовувати фідбек від користувачів.

Створення скетчів та прототипів дозволяє зрозуміти, як буде виглядати та працювати інтерфейс перед розробкою його на реальному рівні. Це допомагає виявити проблеми та внести необхідні зміни в дизайн та функціональність, що економить час та ресурси у майбутньому.

Але у випадку розробки для проектування інтерфейсу буде використаний варфрейм який є важливим інструментом у проектуванні користувацького інтерфейсу(UI). Його використовують для візуалізації структури та розташування елементів у інтерфейсі, не звертаючи уваги на деталі дизайну, кольори та графічні ефекти.

У проектуванні інтерфейсу для сайту поліклініки варфрейми мають декілька цілей. Вони допомагають визначити структуру сайту, включаючи головне меню, підменю, розділи та розташування різних функцій та

інформації. Варфрейми також допомагають виявити потреби користувачів, візуалізуючи розташування кнопок для запису на прийом, пошуку лікарів, перегляду медичної інформації та інших важливих функцій.

Крім того, варфрейми можуть бути використані для узгодження з командою проекту та збору відгуків. Вони дозволяють швидко показати загальну структуру та функціональність інтерфейсу та отримати відгуки та пропозиції щодо його вдосконалення. Також вони можуть бути використані для проведення тестування з користувачами, щоб отримати відгуки щодо зручності навігації, розташування елементів та загального враження від інтерфейсу.

Застосування варфреймів у проектуванні користувацького інтерфейсу для сайту поліклініки допомагає забезпечити логічну та зручну структуру, відповідну потребам користувачів та бізнесу. Вони є важливим інструментом для створення ефективного та привабливого інтерфейсу для поліклінічного веб-сайту. Вони можуть бути створені вручну за допомогою паперу та олівця або за допомогою спеціалізованих програм для дизайну інтерфейсу. Вони можуть бути статичними, або інтерактивними, з можливістю навігації між сторінками та елементами.

На (рис. 2.8) я розробив прототип головної сторінки, яка надає візуальний інтерфейс для веб-додатку з допомогою Figma [15]. Моя мета була створити естетично привабливий та функціональний дизайн, який забезпечує зручну навігацію та залучає користувачів до взаємодії з додатком. У моєму прототипі головної сторінки я використав сучасний мінімалістичний стиль, з фокусом на чистоті форм та використанні яскравих контрастних кольорів. Я врахував принципи веб-дизайну, такі як візуальна ієрархія, баланс елементів та використання простору, щоб створити зручне та привабливе візуальне

Окрім цього, я врахував важливі елементи, які сприяють користувацькому досвіду, такі як проста навігація, використання іконок для ілюстрації функціональності та адаптивний дизайн, який забезпечує оптимальне відображення на різних пристроях.

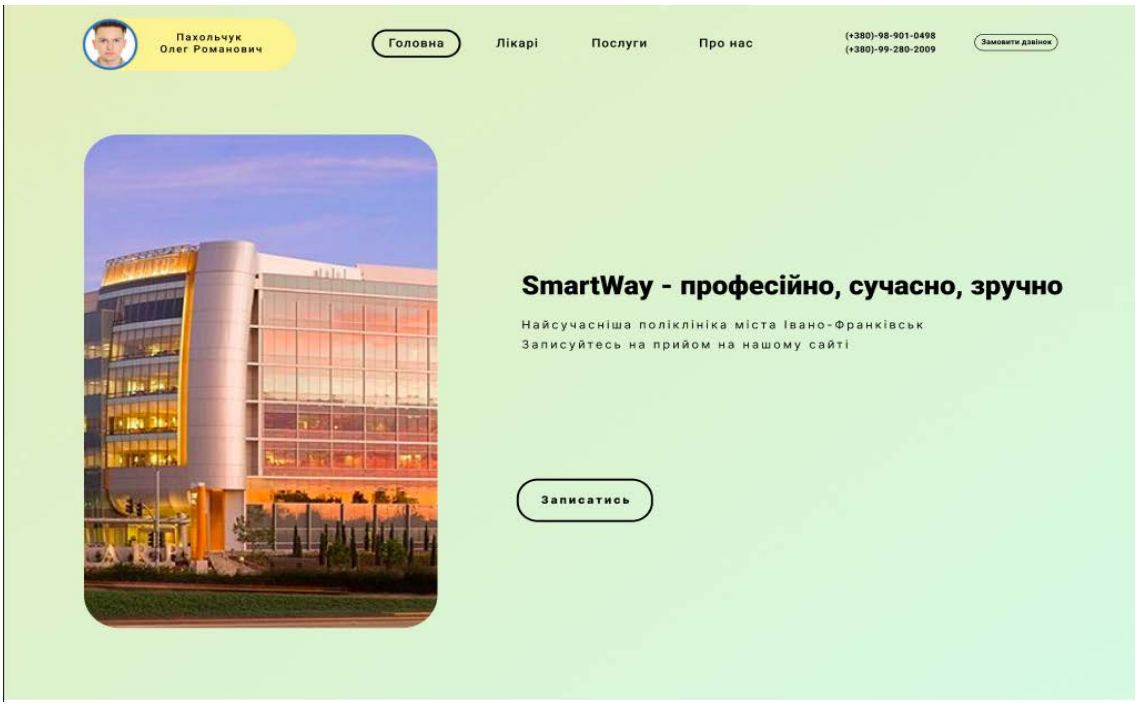

Рисунок 2.8 – Прототип головної сторінки

В результаті роботи над прототипом головної сторінки, я створив ефективне та привабливе веб-середовище, яке допоможе залучити користувачів та покращити їх взаємодію з веб-додатком. Який у свою чергу підлягатиме ще додатковому тестуванню зі сторони звичайних користувачів. Оскільки це не є вже готовий варіант.

На (рис. 2.9) зображений прототип меню на головній сторінці, де переховуються переваги, кількома можливостями функціоналу сайту. Прототип також враховує потреби різних користувачів та розширює можливості додатку. Я розмістив важливі функції та інформацію на доступних місцях, щоб користувачі могли легко знайти те, що їм потрібно. Крім того, я використав чіткі та зрозумілі написи, які допомагають користувачам швидко орієнтуватися та використовувати додаток без зайвих зусиль.

Усі вищезгадані рішення та деталі були обрані з урахуванням потреб користувачів та мети додатку. Прототип головної сторінки є першим кроком до створення зручного та привабливого веб-додатку, який здатний відповісти на потреби користувачів та забезпечити їм задоволення від використання.

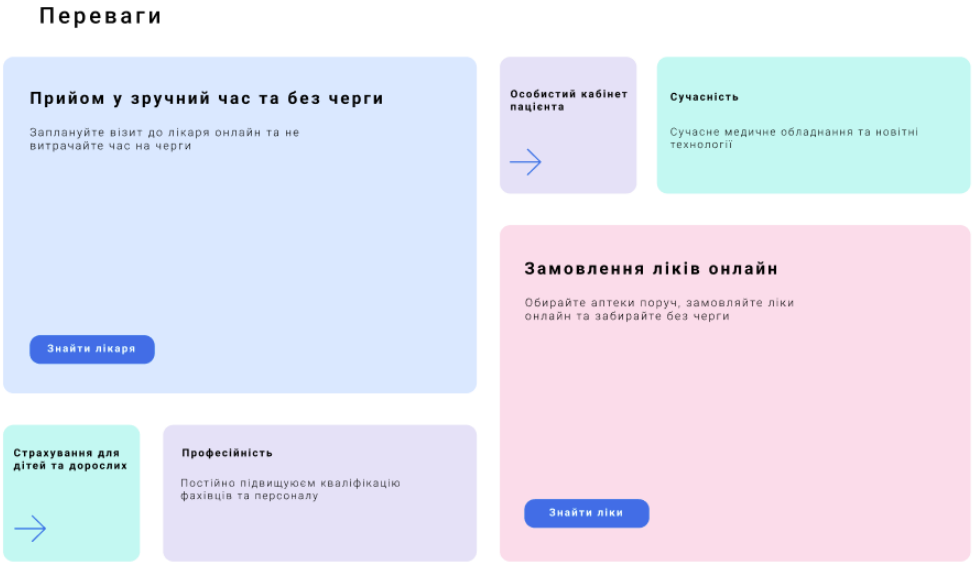

Рисунок 2.9 – Прототип меню з кількома можливостями веб-сайту

При розробці прототипу я активно використовував принципи респонсивного дизайну, забезпечуючи його адаптивність на різних пристроях. Це дозволить користувачам зручно використовувати додаток як на комп'ютерах, так і на мобільних пристроях, зберігаючи при цьому консистентність та зручну навігацію.

Окрім того, прототип використовує підходи до взаємодії, що покращують користувацький досвід. Наприклад, я використав елементи drag-and-drop для зручного переміщення елементів на сторінці та інтерактивні ефекти, які реагують на дії користувача, надаючи відчуття активності та взаємодії з інтерфейсом.

У прототипі я також врахував важливість доступності та використання універсальних дизайнерських рішень. Я забезпечив контрастність кольорів для полегшення читання та використання шрифтів з хорошою читабельністю. Крім того, я забезпечив можливість зміни розміру шрифту та інших параметрів, щоб користувачі з різними потребами могли налаштувати інтерфейс під свої вимоги.

На (рис 2.10) зображена сторінка із всіма лікарями в поліклініці з можливістю пошуку лікаря як за спеціалізаціє, так і за іменем. Чи навіть

банально за рейтингом, після чого відкриється інформація про самого лікаря з всіма його особистими даними, і можливістю записатись в нього на прийом.

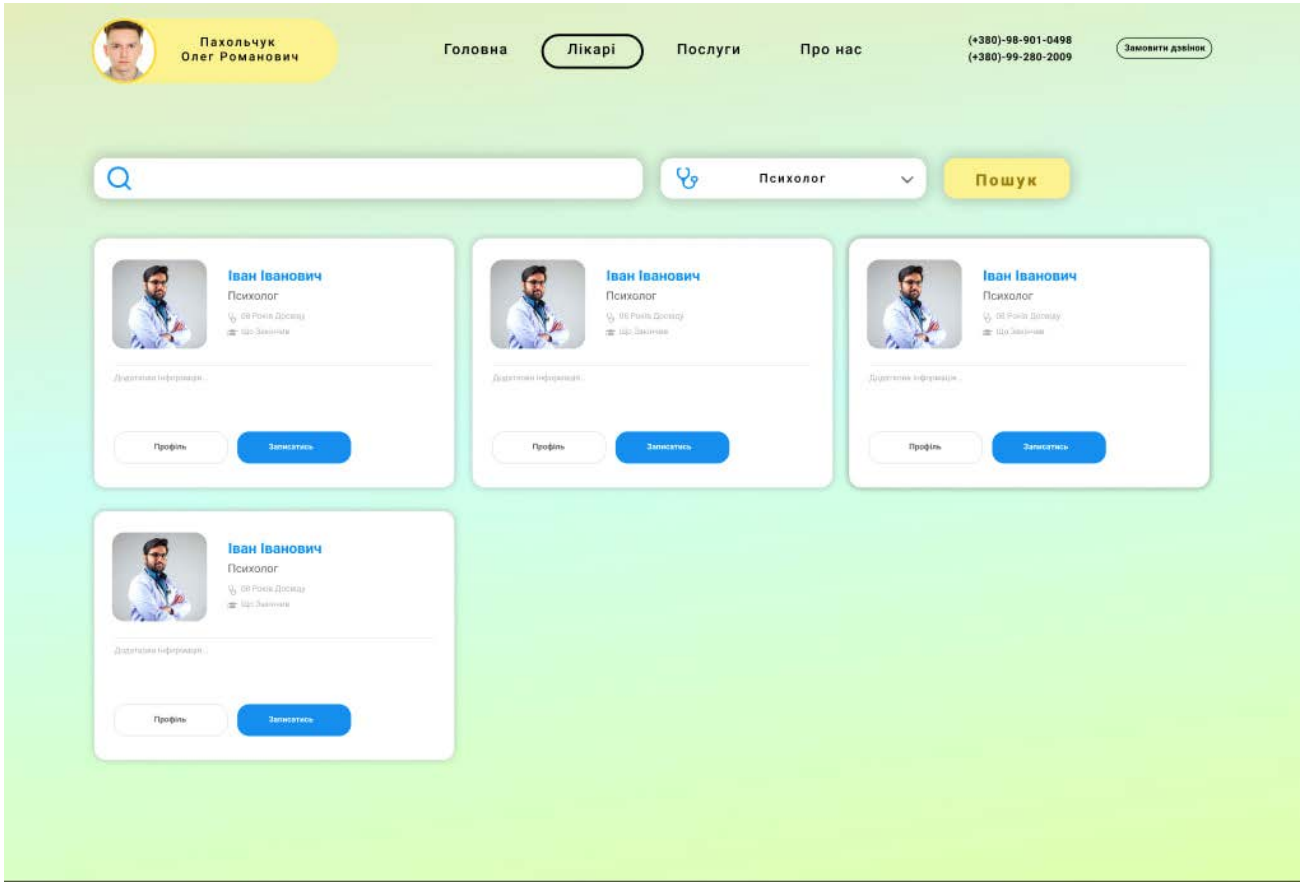

Рисунок 2.10 – Прототип сторінка із всіма лікарями

Система привілеїв доступу дозволяє адміністратору обмежувати доступ до конфіденційної інформації та забезпечувати безпеку даних. Генерація звітів про роботу поліклініки надає можливість контролювати фінансову та статистичну інформацію. Керування лікарськими кабінетами та медичним обладнанням допомагає забезпечити їх ефективне використання. Створення та керування списками послуг дозволяє адміністратору легко змінювати ціни та опис послуг.

Введення системи нагадувань пацієнтам про прийоми та медичні заходи сприяє вчасній та точній інформації. Керування групами пацієнтів дозволяє організувати спеціалізовані операції та медичні програми. Управління запланованими медичними заходами та ресурсами сприяє їх ефективному

плануванню. Контроль використання медичних ресурсів та генерація відповідних звітів допомагає управляти затратами та забезпечувати належну наявність ресурсів. Керування робочим графіком лікарів та надсилання повідомлень сприяє ефективній комунікації та обміну важливою інформацією. Усі ці функції та можливості дозволяють користувачам сайту поліклініки на RoR ефективно керувати медичним закладом, забезпечуючи високу якість надання послуг та задоволення потреб пацієнтів.

Але разом з цим дипломна робота досі розширюється функціоналом та покращення ефективності сайту поліклініки на Ruby on Rails. Розглянуті можливості відображають актуальні тенденції в сфері медичного програмного забезпечення та сприяють подальшому розвитку поліклінічного веб-додатку.

Висновки до розділу 2

Результатом розділу є розробка загальної архітектури веб-сайту, і додаткових знарядь які допоможуть при розробці. Для того щоб краще розуміти що потрібно розробити було написано в розділі 2.2 кілька User Stories. Створено та описано структуру бази даних та вказано яка бака використала. Змодельовано кілька прототипів дизайну головної веб-сайту. І було обґрунтовано вибір мови програмування.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-САЙТУ

б3.1 Розробка веб-сайту

Початок роботи з розробки сайту поліклініки на Ruby on Rails завдяки правильному вибору технології був досить легким. Я зайнявся вивченням основних концепцій та структури RoR, що дало мені глибокі уявлення про фреймворк. Перші кроки у розробці сайту вимагали від мене ретельного планування архітектури, створення моделей та міграцій бази даних. Крім того, я ознайомився з використанням HTML, CSS та JavaScript для створення привабливого та функціонального інтерфейсу користувача. Для розробки проекту був використаний Visual Studi Code. А початок роботи з RoR починається із створенням папки з проектом, в якій буде міститись дерево файлів проекту (рис.3.1).

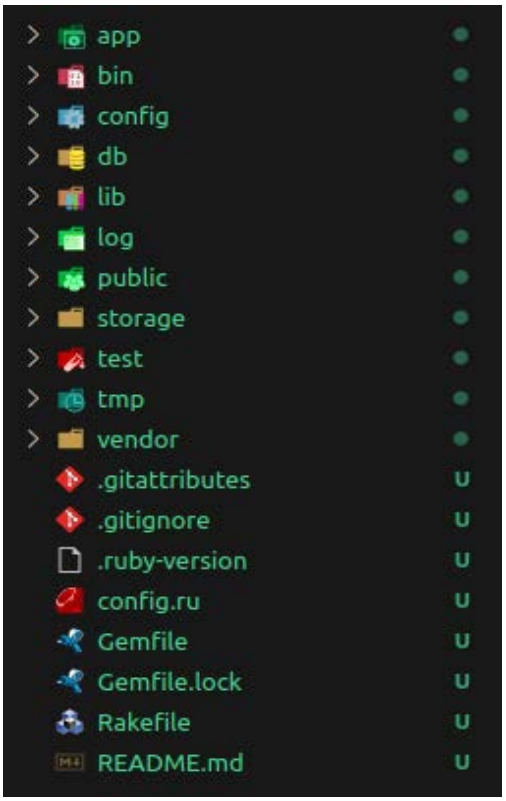

Рисунок 3.1 – Структура файлів проекту

Для цього в терміналі у потрібній локацій потрібно вписати команду **rails new hospital –database=postgresql**, де hospital назва проекту, а як було вказано вище для розробки проекту була використана PostgeSQL тому з допомогою префікса –database=posgresql ми вказуєм, що основною базою даних для проекту буде саме postgresql, де RoR автоматично включить всі необхідні геми для роботи з цією базою даних, написавши конфігураційний файл з налаштуванням. Після чого з допомогою команди **bundle** ми скачаєм усі необхідні залежності для нашого проекту, а саме з цією командою вона переглядає файл «Gemfile», так переконується, що всі вказані геми встановлені, залежності задоволені та версії сумісні. Після цих процедур з допомогою команди **rails serve** можна запусти локальний сервер проекту, де перейшовши по локальному посиланні, ми перейдем на головну сторінку веб-сайту (рис. 3.2) [16].

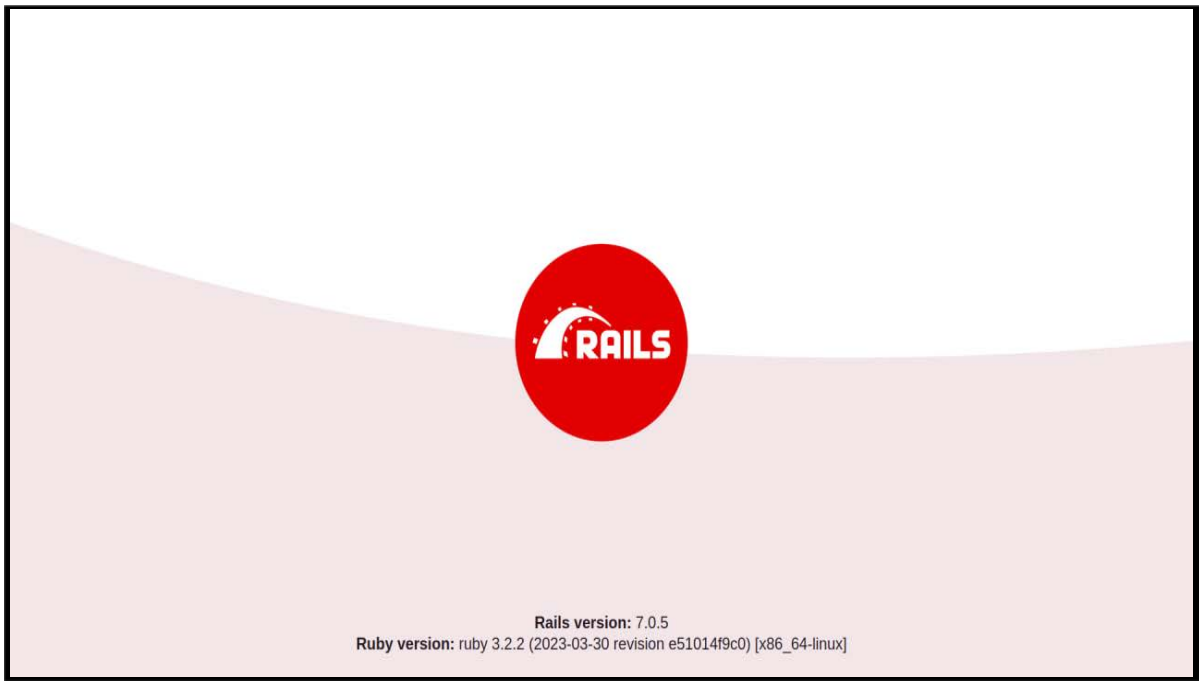

Рисунок 3.2 – Головна сторінка при створенні нового проекту

Як було проговорено в попередніх розділах rails використовує MVC-патерн, де моделі і представлення уже було розглянуто, але що стосується контролера він відповідає за обробку HTTP-запитів, взаємодію з моделями та представленням даних на веб-сторінках. Метою контролера є забезпечення

функціональності, пов'язаної з управлінням пацієнтами, лікарями та записами на прийоми в поліклініці.

Контролер повинен мати можливості додавання, редагування та видалення пацієнтів та лікарів, відображення списку пацієнтів та лікарів, а також їхніх основних даних, запис пацієнта на прийом до лікаря, відображення списку записів на прийоми та їхніх деталей, а також пошук пацієнтів та лікарів за різними критеріями. І розміщується він в папці «app/controllers». Контролер також співпрацює з представленнями, які можуть бути написані за допомогою HTML, CSS [17,18] та використовувати динамічні шаблони, такі як ERB (Embedded Ruby). Для забезпечення безпеки та обмеження доступу, контролер у веб-сайті поліклініки буде використовувати механізми автентифікації та авторизації, наприклад, гем Devise та Pundit.

Приклад розробленого контролера для сторінки з пацієнтами для сайту поліклініки:

```
class PostsController < ApplicationController
  before_action :set_patient, only: [:show, :edit, :update, :destroy]
  def index
    @posts = Patient.all
  end
```

Action index відповідає HTTP-запиту GET і використовується для відображення списку пацієнтів, лікарів або записів на прийоми. Він отримує необхідні дані з моделі та передає їх до відповідного представлення для відображення.

```
  def show
  end
```

Action show також відповідає HTTP-запиту GET і використовується для відображення деталей певного пацієнта, лікаря або запису на прийом. Він отримує ідентифікатор об'єкта з параметрів запиту, знаходить відповідний запис у моделі і передає його до відповідного представлення.

```
def new
  @post = Patient.new
end
```

Action new відповідає HTTP-запиту GET і використовується для відображення форми для створення нового пацієнта, лікаря або запису на прийом. Він просто відображає пусту форму, яку користувач може заповнити.

```
def create
  @patient = Patient.new(patient_params)
  if @patient .save
    redirect_to @patient, notice: 'Пост був успішно створений.'
  else
    render :new
  end
end
```

Action create відповідає HTTP-запиту POST і використовується для збереження нового пацієнта, лікаря або запису на прийом. Він отримує дані з форми, перевіряє їх на правильність та створює новий запис у відповідній моделі.

```
def edit
end
```

Action edit відповідає HTTP-запиту GET і використовується для відображення форми для редагування пацієнта, лікаря або запису на прийом. Він отримує ідентифікатор об'єкта з параметрів запиту, знаходить відповідний запис у моделі і передає його до відповідного представлення з заповненими полями форми.

```
def update
  if @patient.update(post_params)
    redirect_to @patient, notice: 'Пост був успішно оновлений.'
  else
    render :edit
```

```
  end
end
```

Action update відповідає HTTP-запиту PATCH або PUT і використовується для збереження змін у пацієнта, лікаря або запису на прийом. Він отримує дані з форми, перевіряє їх на правильність та оновлює відповідний запис у моделі.

```
def destroy
  @patient .destroy
  redirect_to posts_url, notice: 'Пост був успішно видалений.'
end
```

Action destroy відповідає HTTP-запиту DELETE і використовується для видалення пацієнта, лікаря або запису на прийом. Він отримує ідентифікатор об'єкта з параметрів запиту, знаходить відповідний запис у моделі та видаляє його.

```
  private
  def set_ patient
    @patient = Patient.find(params[:id])
  end
  def patient_params
    params.require(:patient).permit(:name, :surname)
  end
end
```

І два приватних методи один з яких синтаксичний цукор, що дозволяє з допомогою цього коду, писати його кілька разів один і той самий код, а другий є сильним параметром, певним захистом, де ми вказуємо які саме параметри можна міняти у пацієнтів в проекті.

Ці actions можна додатково налаштувати для виконання додаткових дій, які відповідають потребам конкретного сайту поліклініки. Крім того, можна визначити власні додаткові actions, які реалізують специфічну функціональність, яка може бути потрібна в контексті сайту поліклініки.

Як згадувалось в попередніх розділах присутній не менш важливий елемент архітектури RoR, що відповідає за управління даними, включаючи логіку, яка має доступ до бази даних, а саме модель. Як було загадано раніше з допомогою моделі ми звертаємся до бази даних, через що можем виконувати базові операції з даними, такі як їхнє зберігання, оновлення, видалення, та перегляд. Окрім того, вони відповідають за валідацію даних.

Таким чином маючи контролер для користувачів ми можем створити для нього модель, з допомогою команди **rails generate model User**. Окрім створеної моделі команда створює міграцію, без якої наш проект не працюватиме, і яка слугує табличкою в базі даних, і виглядає наступним чином:

```
class DeviseCreateUsers < ActiveRecord::Migration[6.1]
  def change
    create_table :users do |t|
      t.string :name,              null: false
      t.string :surname,               null: false
      t.string :email,             null: false, default: ""
      t.string :encrypted_password, null: false, default: ""
      t.string   :reset_password_token
      t.datetime :reset_password_sent_at
      t.datetime :remember_created_at
      t.string :gender,              null: false
      t.string :status,              null: false
      t.date :date_of_born,                null: false
      t.string :phone,               null: false
      t.string :adress,              null: false
      t.timestamps null: false
    end
    add_index :users, :email,                unique: true
    add_index :users, :reset_password_token, unique: true
  end
end
```

Після того як ми прописали усі необхідні поля в таблиці для користувачів можем використати команду **rails db:migrate** з допомогою якої виконується міграція бази даних. Коли вносяться якісь зміни у структуру бази даних в RoR, з

допомогою тієї команди RoR перевіряє, які міграції вже були виконані в базі даних, і запускає тільки ті міграції, які ще не були застосовані, або для зміни вже існуючих табличок.

В той час модель для користувачів виглядає таким чином:

```
class Users < ApplicationRecord
  enum role: {
    patient: «patient»,
    doctor: «doctor»,
    admin: «admin»,
    personnel: «personnell»
  }
  validates :name, :surname, :email, presence: true
  validates :gender, :status, :date_of_born, presence: true
  validates :phone, :adress, presence: true
end
```

Таким чином ми отримає базову логіку з користувачами, де справа лишається за візуальним відображенням функціоналу, та реалізацією іншого функціоналу проекту. З допомогою гему «enum» було створено ролі для користувачів, з допомогою якого можна визначати список можливих значень для атрибута моделі і надати зручний спосіб роботи з цими значення. А з допомогою validate, RoR перевіє дані перед їхнім зберіганням, враховуючи що ми вписали presence:true, він перевіряє саме чи вони присутні.

З допомогою використання enum це допомогло розробити зручний інтерфейс для сторінки з всіма користувачами (рис.3.3), яка буде доступна, лікарям, адміністраторам, та медичному персоналу, не включаючи пацієнтів, і дозволило зручно розробити додаткові можливості для різних ролей користувачів. На цій сторінці зображений список усіх користувачів, з різними даними про кожного з користувачів, включаючи пацієнтів, буде виводити і їхню медичну книгу, лікарі матимуть можливість переглядати як повністю усіх пацієнтів, так і з допомогою фільтрації тільки свої пацієнтів, для адміністраторів відкриватимуться додаткові можливості з редагуванням, чи видаленням пацієнта з тих чи інших причин. Для зручності були додані можливості фільтрації та

сортування списку, як за алфавітним порядком, так і за всіма іншими параметрами які тільки відслідковуються на сайті.

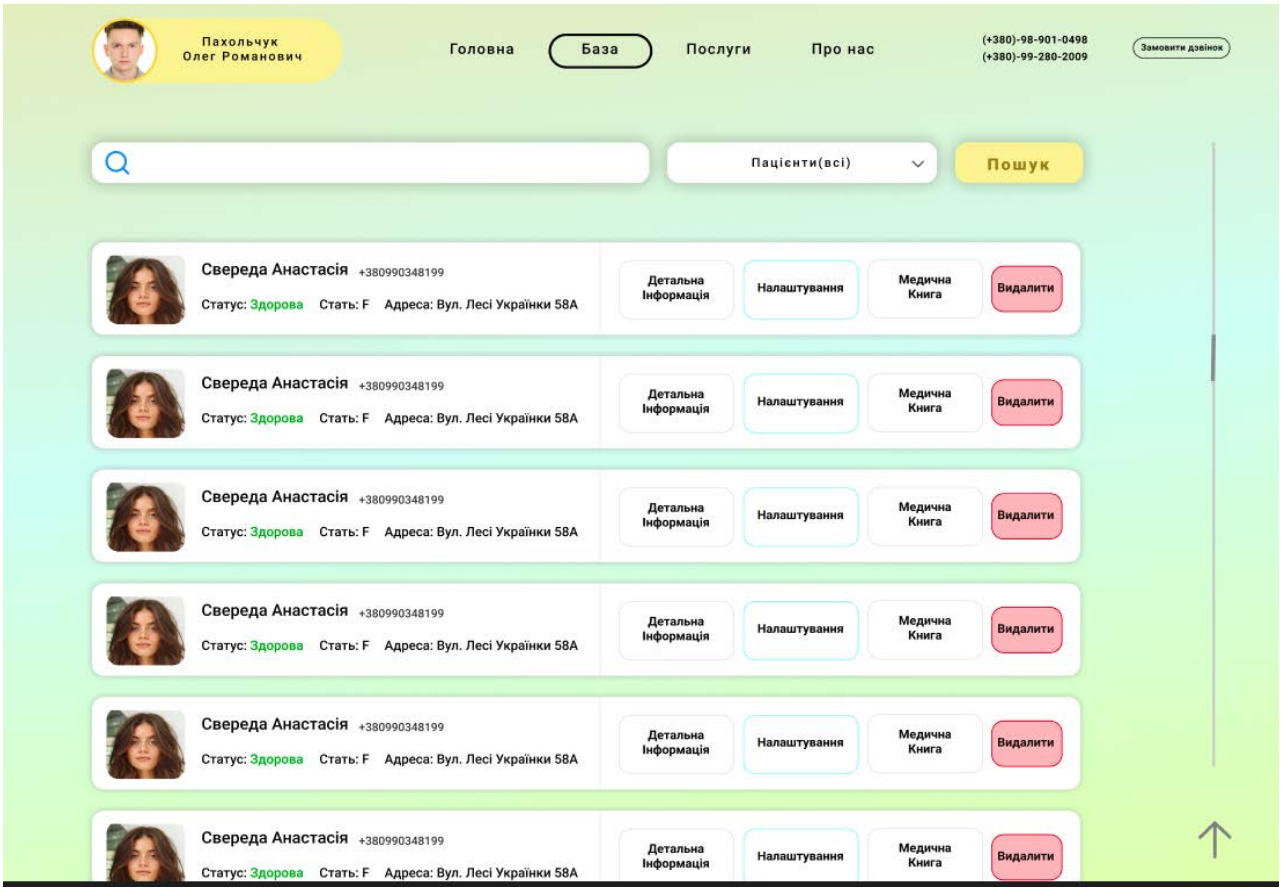

Рисунок 3.2 – Сторінка з всіма пацієнтами зареєстрованими на сайті

Однією із ключових технологій використаній при розробці веб-сайту був JavaScript [19] з допомогою якого можна створювати різні динамічні елементи. В реалізації веб-сайту був використаний JavaScript фреймоворк Stimulus [20], для реалізації інтерактивності на клієнтській стороні. Stimulus є вбудованою технологією, яка використовується за замовчуванням в RoR 7, і саме ця версія була використана в реалізації кваліфікаційної роботи. Він дозволяє вбудовувати динамічну функціональність в сторінки RoR. Де основною ідеєю є те, що можна визначати контролери, які відповідають елементам DOM на сторінці, і які можуть реагувати на події, та змінювати стан сторінки, виконуючи певні дії або змінюючи вигляд.

За допомогою Stimulus, я додав динамічні функціональність до окремих елементів на веб-сторінці, таких як форми, кнопки або списки. Я використовував контролери Stimulus для керування цими елементами та обробки подій, що виникають при їх взаємодії. Завдяки Stimulus, користувачі могли взаємодіяти зі сторінкою без необхідності перезавантаження всієї сторінки, що покращило їхнє взаємодію з веб-додатком. Таким чинному були реалізовані модальні вікна.

Висновки до розділу 3

Результатом розділу є детальний опис процесів на початку розробки веб-сайту. Розроблено базовий контролер для користувачів, з приватними методами, як для спрощення самого коду, так і для захисту, створена модель для користувачів з створеним списком допустимих значень для атрибутів моделі, і валідацією даних перед збереженням. Розроблена табличка в базі даних з користувачами з всіма необхідними полями для різних категорій користувачів. Де в результаті, було розроблено повноцінну частину веб-сайту з всіма користувачами в базі даних.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи створено веб-сайт для оптимізації процесів роботи в поліклініці засобами Ruby on Rails, за допомогою якого можна записуватись на прийом уникаючи черг, виписувати рецепти онлайн пацієнтам, отримуванням сповіщення як про прийом ліків, так і якихось аналізів, займатись моніторингом пацієнтів, починаючи від їхнього стану здоров’я, закінчуючи палатою в які вони лежать. Були проведені дослідження та проаналізовані вже існуючі сайти-аналоги, за допомогою яких організовано роботу медичних закладів.

При виборі інструментів для розробки сайту було враховано їхню ефективність, як хороші так і погані сторони. Розроблену архітектуру веб-сайту, і додаткових технологій які допоможуть при розробці. Додатково написані User Stories дозволили краще розуміти потреби користувачів потрібно розробити. Створено структуру бази даних з допомого PostgeSQL.

В результаті, було створено повноцінний веб-сайт, що виконує усі поставлені задачі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Обласна дитяча клінічна лікарня. веб-сайт. URL: http://www.odkl.if.ua/ (дата звернення: 25.03.2023)

2. Обласна клінічна лікарня веб-сайт URL: https://www.okl.if.ua/ (дата звернення: 26.03.2023)

3. Веб-сайт Wayback Machine веб-сайт URL: https://web.archive.org/ (дата звернення: 27.03.2023)

4. Медична соціальна платформа веб-сайт URL: https://medics.ua/ (дата звернення: 28.03.2023)

5. Ruby on Rails Wikipedia.com веб-сайт URL: https://en.wikipedia.org/wiki/Ruby_on_Rails (дата звернення: 30.03.2023)

6. Документація Ruby on Rails веб-сайт URL: https://guides.rubyonrails.org/ (дата звернення: 8.04.2023)

7. MVC Wikipedia.com веб-сайт URL: https://en.wikipedia.org/wiki/Model-View-Controller (дата звернення: 8.04.2023)

8. Документація Tailwind CSS веб-сайт URL: https://tailwindcss.com/docs/installation (дата звернення: 8.04.2023)

9. User Stories Wikipedia.com веб-сайт URL: https://en.wikipedia.org/wiki/User_story (дата звернення: 12.04.2023)

10. Документація PostgreSQL веб-сайт URL: https://www.postgresql.org/docs/ (дата звернення: 13.04.2023)

11. Active record pattern Wikipedia.com веб-сайт URL: https://en.wikipedia.org/wiki/Active_record_pattern (дата звернення: 15.04.2023)

12. Документація Devise веб-сайт URL: https://github.com/heartcombo/devise (дата звернення: 15.04.2023)

13. Документація Redis веб-сайт URL: https://redis.io/docs/ (дата

звернення: 17.04.2023)

14. Веб-сайт draw.io веб-сайт URL: https://app.diagrams.net/ (дата звернення: 18.04.2023)

15. Веб-сайт Figma веб-сайт URL: https://www.figma.com/ (дата звернення: 19.04.2023)

16. Rack Wikipedia.com веб-сайт URL: https://en.wikipedia.org/wiki/Rack_(web_server_interface) (дата звернення: 25.03.2023)

17. Документація HTML веб-сайт URL: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 2.05.2023)

18. Документація CSS веб-сайт URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 2.05.2023)

19. Документація JavaScript веб-сайт URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 3.05.2023)

20. Документація Stimulus веб-сайт URL: https://stimulus.hotwired.dev/handbook/introduction (дата звернення: 4.05.2023)