КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

122 «Комп'ютерні науки»
(шифр і назва спеціальності)

«Прикладне програмування»
(назва освітньої програми)

Кваліфікаційна робота бакалавра

на тему: «Медична інформаційна система для поліклініки та амбулаторій»

Виконав _____
(Підпис)
Панасюк Олександр Ігорович
(прізвище, ім'я, по батькові)

Керівник Краснощок Віктор Миколайович
(прізвище, ім'я, по батькові)

_____
(Резолюція «До захисту»)

Попередній захист:

_____
(Висновок: "До захисту в екзаменаційній комісії ")
Завідувач кафедри _____ Плескач В.Л.
(Підпис) (Прізвище, ініціали)
(Дата)

Київ – 2021

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

№ п/п	Назва етапів бакалаврської роботи	Термін виконання етапів бакалаврської роботи	Відмітка про виконання
1.	Вибір теми та наукового керівника бакалаврської роботи	26.10.2020	Виконано
2.	Видача завдання бакалаврської роботи	23.11.2020	**Заява**
3.	Настановча групова співбесіда з бакалаврської роботи	01.12.2020	Початок роботи з темою
4.	Затвердження плану бакалаврської роботи	18.02.2021	Виконано
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	25.02.2021	Виконано
6.	Підготовка і подання науковому керівнику першого варіанту I розділу роботи	05.03.2021	Виконано
7.	Підготовка і подання науковому керівнику першого варіанту II розділу роботи	09.04.2021	Виконано
8.	Підготовка і подання науковому керівнику першого варіанту III розділу роботи	07.05.2021	Виконано
9.	Подання роботи у першому варіанті	11.05.2021	Виконано
10.	Оформлення пояснювальної записки бакалаврської роботи	12.05.2021	Виконано
11.	Подання бакалаврської роботи на попередній захист	**24.05.2021**	Виконано
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	28.05.2021	Виконано
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	11.06.2021	Виконано
14.	Захист бакалаврської роботи	23.06.2021	Виконано

Здобувач вищої освіти _____

(підпис)

Керівник _____

(підпис)

ВІДОМІСТЬ ДИПЛОМНОЇ РОБОТИ

Зміст пояснювальної записки (перелік питань під час дослідження)

Складові частини дипломної роботи	Обсяг, арк. 63
Титульний аркуш	1 ст.
Календарний план виконання кваліфікаційної роботи бакалавра	1 ст.
Відомість дипломної роботи	1 ст.
Анотація	1 ст.
Анотація (іноземною мовою-англійською)	1 ст.
Зміст	2 ст.
Вступ	4 ст.
1. Загальносистемні питання. Аналіз розв’язуваної задачі й огляд наявних результатів. Постановка задачі на проектування	17 ст.
2. Проектні і технічні рішення. Види забезпечення	19 ст.
3. Реалізація і опис роботи системи	10 ст.
Висновки	1 ст.
Перелік посилань	2 ст.
Додатки	3 ст.

				ДП ХХХХ 00.000.00		
	ПІБ	Підп.	Дата			
Розробн.				Відомість дипломної роботи	Лист	Листів
Керівн.						
Н/контр.	Макаренко С.А.					
Зав.каф.	Плескач В.Л.					

АНОТАЦІЯ

Кваліфікаційна робота: 63 с., 21 рис., 1 табл., 21 джерел, 1 дод.

Метою роботи є**:**

- Дослідити теоретичні основи побудови програмної системи «Медична інформаційна система для поліклініки та амбулаторій» і провести аналіз необхідних технічних рішень;
- Спроектувати і реалізувати програмну систему «Медична інформаційна система для поліклініки та амбулаторій»

Об’єкт дослідження: процеси пов’язані з веденням прийому у медичному закладі первинної ланки.

Предмет дослідження: програмно-технічні, організаційні засади, принципи, концептуальні підходи до побудови медичної інформаційної системи.

Методи дослідження. В роботі використовувалися наступні групи методів збору даних:

- Оцінка зручності і відповідності вимогам існуючих систем
- Пошук і вивчення програмно-технічних засобів для створення сучасних веб-застосунків.

Наукова новизна одержаних результатів:

- Виявлено особливості сучасних прикладних медичних інформаційних систем.
- Описано основні процеси пов'язані з ведення прийому в Українських закладах первинної ланки.

Ключові слова: MIC, медична інформаційна система, ASP.NET Core, ReactJS.

ABSTRACT

Qualification work: 63 pages, 21 figs., 1 table, 21 sources, 1 appendix.

The purpose of the work is:

- Investigate the theoretical foundations of the software system "Medical information system for polyclinic and outpatient" and analyze the necessary technical solutions
- Design and implement a software system "Medical information system for polyclinic and outpatient"

Object of research: processes related to the reception in the primary care facility.

Subject of research: software and hardware, organizational principles conceptual approaches to building a medical information system.

Research methods. The following groups of data collection methods were used in the work:

- Assessment of convenience and features of existing systems
- Search and study of information technologies for creating modern web applications.

Scientific novelty of the obtained results:

- Features of modern applied medical information systems are revealed.
- Described the main processes related to the reception in Ukrainian primary care facilities.

Keywords: MIS, medical information system, ASP.NET Core, ReactJS.

ЗМІСТ

ВСТУП

Розвиток сучасного суспільства відбувається під впливом глобалізації інформаційних процесів. Провідним проявом цих процесів є формування нового інформаційного простору, пов'язаного з появою мережі Інтернет. Формується нова культура в якій ключовими словами є інформація, інформатизація, Інтернет і віртуалізація. Поява нового інформаційного простору посприяла інтеграції інформаційних технологій у більшість соціальних, економічних і бізнес процесів. Загалом одним з основних проявів такої інтеграції є виникнення прикладних інформаційних систем, які покликані спростити життєдіяльність людей. В сучасній державі прикладні інформаційні системи створені майже для всіх сфер суспільного життя.

Охорона здоров’я є однією з найважливіших галузей діяльності держави. В сучасному інформаційному просторі охорона здоров’я є однією з найпоширеніших сфер впровадження провідних інформаційних технологій. В епоху розвинених інформаційних технологій державі необхідно забезпечити лікарів зручним і швидким засобом створення, зберігання і передачі медичної інформації. Для цього було створено медичні інформаційні системи.

Актуальність дослідження.

Прикладні інформаційні системи з’явились зразу після виникнення персональних комп’ютерів, з того часу роль прикладних інформаційних систем в житті людей постійно збільшується. Медичні інформаційні систем існують вже давно, вони є невід’ємною частиною взаємодії між сучасним пацієнтом і лікарем. Також медичні інформаційні системи забезпечують зручні способи зберігання і обробки медичної інформації та створення звітів. В 2018 році в Україні розпочалась реформа системи охорони здоров'я, одним з основних законів реформи, був закон про обов’язкове подання електронних медичних даних до державної бази даних. На основі цих даних «Національна службу здоров'я

України» регулює фінансування медичних закладів. Таким чином якщо раніше медичні інформаційні системи забезпечували лише прискорення і комфорт роботи, то тепер вони є обов'язковою частиною процесу отримання державного фінансування.

Мета дослідження:

- Дослідити теоретичні основи побудови програмної системи «Медична інформаційна система для поліклініки та амбулаторій»;
- Провести аналіз необхідних технічних рішень для зручної побудови системи
- Спроектувати і реалізувати програмну систему «Медична інформаційна система для поліклініки та амбулаторій»

Завдання дослідження. Створення медичної інформаційної системи за допомогою технологій C# 8.0, ASP.Net Core 5.0, HTML, CSS, ReactJS.

Об'єкт дослідження: процеси пов'язані з веденням прийому у медичному закладі первинної ланки.

Предмет дослідження: програмно-технічні, організаційні засади, принципи, концептуальні підходи до побудови медичної інформаційної системи.

Методи дослідження. Вибір методів дослідження продиктований його цілями, предметом дослідження і характером основних досліджуваних змінних. В роботі використовувалися наступні групи методів збору даних:

- Аналіз медичних процесів на основі документації державного компоненту Ehealth
- Оцінка зручності і відповідності вимогам існуючих систем
- Пошук і вивчення програмно-технічних засобів для створення сучасних веб-застосунків.

Наукова новизна одержаних результатів полягає в наступному:

- Виявлено особливості сучасних прикладних медичних інформаційних систем.
- Описано основні процеси пов'язані з ведення прийому в Українських закладах первинної ланки.

Теоретична значимість дослідження полягає в наступному: розглянуто можливості медичних інформаційних як основного засобу зберігання медичних даних.

Практичне значення одержаних результатів: у ході розробки системи було відпрацьовано процес ведення медичного прийому за стандартом ICPC2 для закладів первинної ланки.

Апробація результатів дипломної роботи. Результати роботи були представлені та отримали позитивну оцінку на конференціях:

- «Наукові розробки молоді на сучасному етапі», XVII Всеукраїнська наукова конференція молодих вчених та студентів, Київ, 26-27 квітня 2018 року, Київський національний університет технологій та дизайну;
- «Прикладні системи та технології в інформаційному суспільстві», II Міжнародна науково-практична конференція, Київ, 1 жовтня 2018 р., Київський національний університет імені Тараса Шевченка;
- «Мехатронні системи: інновації та інжиніринг», Міжнародна науково-практична конференція, Київ, 15 червня 2018 р. Київський національний університет технологій та дизайну;
- «Актуальні питання енергозбереження як вимога безпеки життєдіяльності» м. Київ, 4–5 червня 2019р., Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».

- «Прикладні системи та технології в інформаційному суспільстві», IV Міжнародна науково-практична конференція, Київ, 30 вересня 2020 р., Київський національний університет імені Тараса Шевченка;

Публікації на тему дипломної роботи. Основні результати дипломної роботи опубліковано у 7 наукових працях, з яких 2 статті у фахових виданнях України, розділ у колективній монографії, 5 тези доповідей на всеукраїнських та міжнародних конференціях.

Структура роботи. Дипломна робота складається зі вступу, трьох розділів, поділених на підрозділи, висновків, додатків та списку використаних джерел з 30 найменувань.

РОЗДІЛ 1
ЗАГАЛЬНОСИСТЕМНІ ПИТАННЯ. АНАЛІЗ РОЗВ'ЯЗУВАНОЇ ЗАДАЧІ Й ОГЛЯД НАЯВНИХ РЕЗУЛЬТАТІВ. ПОСТАНОВКА ЗАДАЧІ НА ПРОЕКТУВАННЯ

1.1 Інформаційні системи яка основна складова сучасного інформаційного простору

Системою називають будь-який об'єкт, який одночасно розглядається і як єдине ціле, і як об'єднана в інтересах досягнення поставлених цілей сукупність різнорідних елементів. Системи значно відрізняються між собою як по складу, так і по головним цілям[1, 2, 3].

Поняття системи дуже поширене в галузі інформаційних технологій. Найчастіше системою називають сукупності технічних, програмних і програмно-технічних засобів. Апаратну частину комп'ютера можна назвати системою. Сукупність комп'ютерних програм метою яких є розв'язання конкретних прикладних задач, а також методи і процедури автоматизації документообігу і управління розрахунками. Одним з розповсюджених варіантів використання поняття система є інформаційні системи. Мета розробки і спосіб застосування є зрозумілими з назви. Основним завданням інформаційної системи є забезпечення збору, обробки, зберігання, видачі інформації, яка може бути використана на благо людини.

Інформаційна система - взаємозв'язана сукупність засобів, методів і персоналу, використовуваних для зберігання, обробки та видачі інформації для задоволення потреб користувача.

В сучасних інформаційних системах використовують багато технічних засобів, однак в більшості випадків основним є персональний комп'ютер або сервер з підключеними до нього клієнтськими пристроями такими як: персональні комп'ютери, планшети або смартфони. На великих підприємствах або у крупних організаціях, у складі основного технічного комплексу можуть

знаходитись мейнфрейми або суперкомп’ютери, також нещодавно популярним стало використання хмарних сервісів для розгортання і впровадження високонавантажених інформаційних систем. Не слід забувати що в більшості навіть сучасні інформаційні системи не можуть функціонувати без участі персоналу, а також споживачів для яких призначені результати обробки і видачі інформації в системі.

1.2 Огляд і аналіз програмних систем, принципів і засобів створення медичної інформаційної системи

Медична інформаційна система (МІС) — інформаційна система для автоматизації лікувальної установи[1, 3].

МІС класифікують за різними ознаками.

1. Для автоматизації процесів збору і обробки інформації можна виділити автоматизовані і автоматичні медичні інформаційні системи. В автоматизованих системах людина виконує частину операцій по збору й обробці інформації. В автоматичних системах людина не приймає участі у процесах, тобто процеси збору і обробки медичних даних повністю забезпечуються системою.
2. Залежно від типу інформаційної бази, МІС поділяються на системи, що оперують даними, та системи, що оперують знаннями. Системи другого типу - це експертні системи. Їхнє функціонування істотно спирається на знання, отримані від експертів, а результати функціонування близькі результатам аналітичної діяльності експертів.
3. Залежно від виду розв'язуваних задач, МІС можна розділити на такі групи:
    - інформаційно-довідкові — системи автоматизованого пошуку, вимірювальні системи;
    - інформаційно-логічні — діагностичні системи; системи прогнозу; системи моніторингу;

- керуючі або автоматизовані системи управління.

Інформаційно-логічна система призначена для перетворення інформації таким чином, щоб можна було одержати нову інформацію, відсутню в інформаційному масиві.

У системах управління реалізується принципово нова функція — прийняття керуючих рішень.

Найбільш широке поширення в медичних установах одержали інформаційно-пошукові системи (ІПС), які у залежності від характеру інформації поділяються на фактографічні і документальні системи[3].

Фактографічні ІПС містять інформаційні масиви фактичних даних. Аналогами таких систем виступають «паперові» довідники, каталоги, технічні паспорти. У комп'ютерних ІПС фактичні дані звичайно зберігаються в базах даних (БД) і являють собою таблиці, у колонках яких вказано назви різних характеристик об'єктів, а в рядках дані опису (значення характеристик) цих об'єктів.

Документальні ІПС оперують з інформацією у вигляді документів. Прикладами таких систем можуть бути бібліографічна картотека, картотека з історіями хвороби, інші картотеки. Виконуючи пошук, документальна ІПС надає або номера необхідних документів, або список заголовків, або адреси зберігання шуканих документів. При цьому оцінку інформації, що знаходиться в знайдених документах, робить людина.

Керуючі системи реалізують збір інформації про об'єкт управління, обробку інформації, передачу даних в орган управління, формування керуючого рішення.

МІС або автоматизовані системи медичної документації класифікують на основі принципу ієрархії, які відповідають за це особливій структурі охорони здоров`я, що має кілька рівнів, існують:

- МІС, що мають початковий (базовий) рівень, завданням яких є певна підтримка роботи лікарів різної спеціалізації за допомогою ПК; такі системи можуть поліпшити якість діагностичної та профілактичної роботи, а особливо це стосується умов ведення великого прийому пацієнтів при нестачі часу лікарів. Тут можуть бути системи, призначені для пошуку необхідної інформації і даних медичного призначення за запитом лікуючого лікаря; Можуть бути вирішені такі завдання як:
    - системи, які призначені для того, щоб діагностувати різного роду патології, прогнозувати майбутній стан і вказувати рекомендації для лікування;
    - системи, призначені для підтримки і автоматизації діагностики та лікувального процесу, які здійснюються при контакті з пацієнтом;
    - автоматизовані робочі місця лікарів, для того, щоб повністю автоматизувати весь робочий процес лікуючого лікаря будь-якої спеціалізації та забезпечити підтримку інформацією для прийняття лікарем рішень діагностичного і тактичного характеру.
- МІС, що мають рівень лікувально-профілактичного закладу. Видаються такими типами систем:
    - інформаційні технології центрів консультації, призначених для підтримки функцій певних підрозділів і забезпечення інформацією медичного персоналу при консультації ними пацієнтів, прийнятті рішень та діагностуванні;
    - сховище інформації мед. служб, які несуть в собі дані про число і якості складу медичного персоналу конкретної установи, населення, яке обслуговуються ними, основні дані про статистику, характеристики районів та іншу важливу інформацію;

- спеціалізовані регістри, які містять дані про певний контингент людей, заснованому на формалізації історії хвороби та медичній карті пацієнта;
- спеціалізовані сервіси, за допомогою яких проводиться огляд в якості профілактики до огляду лікарем і виявлення особливих груп ризику і пацієнтів, яким необхідна допомога лікаря;
- ІС науково-дослідних інститутів і медичних вищих навчальних закладів, які визначаються рішенням такі завдань як: інформатизація процесу технології навчання, науково-дослідних робіт, діяльність управління).

● МІС територіального рівня характеризуються такими пунктами:

- інформаційні системи територіального рівня охорони здоров'я;
- інформаційні системи завдань технологій в галузі медицини, які забезпечують інформаційну підтримку діяльності мед. персоналу і спеціалізованих служб;
- телекомунікаційні мережі медичного призначення, які створюють єдиний інформаційний простір в регіоні.
- рівень держави, які призначається для забезпечення інформаційної підтримки охорони здоров'я на рівні держави.

Сучасні медичні інформаційні системи піддаються взаємодії з неймовірно великою кількістю всіляких даних. Від ефективності використання цих даних лікарями, їх керівниками, керуючими підрозділами залежить якість послуг, що надаються в сфері охорони здоров'я. Необхідність використання потоку даних дуже великих масштабів, які постійно збільшуються через такі фактори як діагностика, статистика, управлінські рішення, терапія і так далі, на сьогоднішній

день підтверджує необхідність створення автоматизованих інформаційних систем в медичних установах.

Деякий час назад система вітчизняної охорони здоров'я не мала практично ніяких яскравих ознак автоматизації. Такий документообіг як: медичні амбулаторні карти, виписки, аналізи, висновки, звіти, облік пацієнтів та інше виконувався повністю вручну самими співробітниками медичних установ на папері. У свою чергу це сильно уповільнювало процес обслуговування пацієнтів, відбувалися помилки, витрачалась велика кількість часу на заповнення медичних карт і складання звітів, що в сукупності сильно впливало на якість охорони здоров'я в цілому. Все це ускладнювало роботу контролюючих органів. Крім того, ефективність використання медичних інформаційних систем в найближчому майбутньому буде обумовлена здоров'ям населення країни.

Беручи до уваги цей факт, велика кількість лікувально-профілактичних установ активно застосовують у своїй діяльності комплексні медичні інформаційні системи «див. рис. 1.1». Вони є спеціальним продуктом, який може дозволити здійснювати управлінську діяльність медичного закладу на абсолютно іншому, більш високому рівні. Перевагою комплексних медичних інформаційних систем є поєднання найважливіших функцій. Так для багатьох закладів недостатньо системи яка може виконувати лише обмежене коло завдань. В сучасних реаліях медичному закладу водночас потрібно автоматично формувати звітність для подання в державні установи, проводити операційні процеси в автоматичному режимі, зберігати медичні дані про свої пацієнтів. Таким чином комплексні системи надають більшу гнучкість у використанні.

Рисунок 1.1 - Основні функції медичної інформаційної системи

Відмінною рисою медичних інформаційних систем можна назвати перехід від локальної роботи з медичною інформацією до комплексної системи, в якій, вся необхідна інформація, що проходить через медичну установу, може бути доступна з єдиного інформаційного середовища. Крім того, з'являється реалізація технології, при якій немає ніякої необхідності використовувати ресурс паперу, але можливість отримання, при необхідності, паперового варіанту також зберігається. Якість надаваних лікарями послуг стає помітно вище при використанні сучасних технологій МІС. Також зростають показники оптимізації управління різними медичними відділеннями в установі [1, 7].

Настав час, коли на заміну паперових документів лікувально-профілактичних установ приходять сучасні інформаційні технології, які спрямовані на здійснення внутрішніх функцій і рішення управлінських завдань в

галузі медицини. процес інформатизації охорони здоров'я швидко набирає обертів, також збільшується кількість масштабних проектів.

В 2018 році в Україні почав діяти законопроект про реформу системи охорони здоров'я в Україні. Основною метою реформи є забезпечення всім громадянам України рівного доступу до якісних медичних послуг та перебудови систему охорони здоров'я так, щоб у її центрі був пацієнт.

В рамках впровадження реформи було створено «Національну службу здоров'я України» (НСЗУ)

Згідно з Законом, її основними функціями є:

- впровадження державної політики у галузі державних фінансових гарантій медичного обслуговування населення за програмою медичних гарантій;
- моніторинг, аналіз і прогнозування потреб населення України у медичних послугах та ліках;
- виконання функцій замовника медичних послуг та лікарських засобів за програмою медичних гарантій;
- розроблення проекту програми медичних гарантій, внесення пропозицій щодо тарифів;
- укладення, зміна та припинення договорів про медичне обслуговування населення та договорів про реімбурсацію;
- перевірка дотримання надавачами медичних послуг вимог, встановлених порядком використання коштів програми медичних гарантій і договорами про медичне обслуговування населення;
- забезпечення функціонування eHealth — системи охорони здоров'я.

Електронна система охорони здоров'я eHealth — інформаційно-телекомунікаційна система, що забезпечує автоматизацію ведення обліку

медичних послуг та управління медичною інформацією в електронному вигляді, до складу якої входять центральна база даних та електронні медичні інформаційні системи, між якими забезпечено автоматичний обмін даними через відкритий програмний інтерфейс (API).

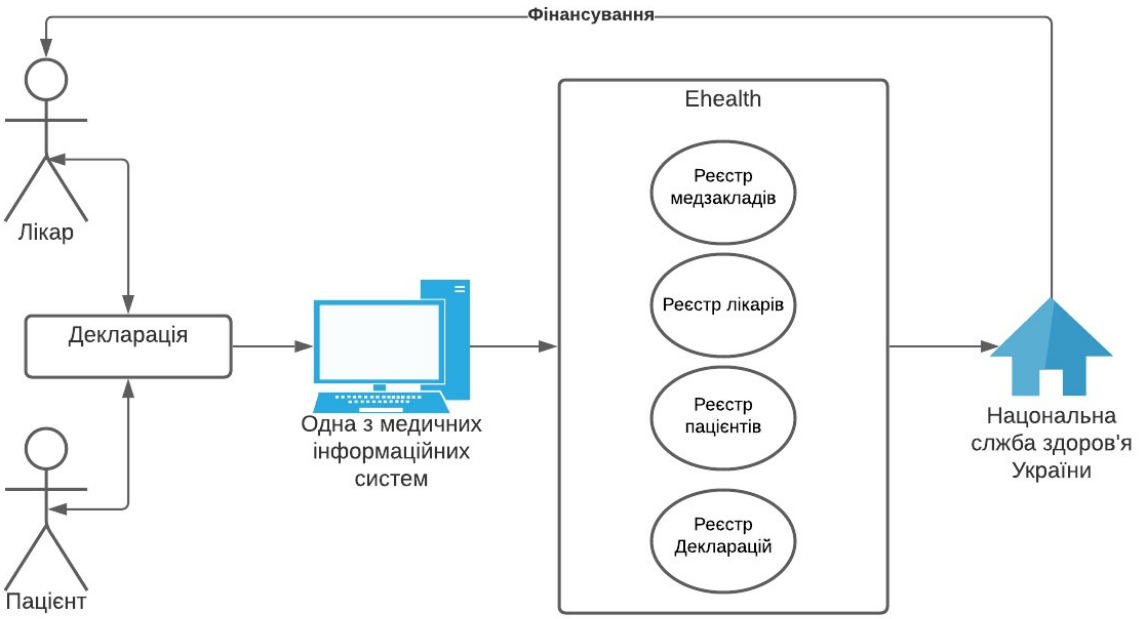


Рисунок 1.2 – Реформа первинної ланки системи охорони здоров’я

1.3 Огляд існуючих медичних інформаційних систем

У даному підрозділі будуть розглянуті медичні інформаційні системи для проведення аналізу над існуючими рішеннями, знаходження їх переваг і недоліків та створення висновку, що підкреслить найважливіші рішення впроваджених систем та виявить можливості, які додатково необхідно реалізувати в новій системі, розроблюваній в рамках кваліфікаційної роботи.

Після створення державного центрального компоненту eHealth з’явилось багато медичних інформаційних систем. На даний момент більше 20.

З найбільш популярних і досконалих слід зазначити:

- Helsi
- Doctor Eleks
- Emcimed
- ПБЧ
- MedStar
- Medics

Система Helsi — медична інформаційна система, що є однією з найбільших в Україні та покриває велику долю ринку. В переліку партнерів компанії Helsi знаходяться лікарі, пацієнти, медичні заклади з усієї України, фармацевтичні компанії та страхові компанії.

За офіційними даними Національної служби здоров'я України через систему Helsi укладено 15,6 мільйона декларацій між лікарями та пацієнтами, що становить приблизно 44% всього обсягу укладених декларацій.

За даними самої компанії: через систему HELSI кожного місяця записується до лікаря більше ніж 2 мільйони користувачів. Продуктом користується більш ніж 16 тисяч лікарів по всій Україні. Система працює в більш ніж 1400 медичних закладах. З HELSI співпрацюють більше 530 аптечних мереж, які за період співпраці погасили більше 6 мільйонів рецептів.

Система HELSI розроблена у форматі веб-застосунку. Веб застосунок складається з трьох частин:

- helsi.me
- helsi.pro
- reform.helsi.me

helsi.me – це частина системи призначена для пацієнтів. Вона знаходиться у відкритому доступі «див. рис. 1.1». Основними функціями порталу helsi.me є перегляд каталогу лікарів, перегляд каталогу медичних установ, можливість

попереднього заповнення декларації, яке дає змогу ввести всю необхідну для заключення інформацію на сайті і лише прийти до лікаря щоб підтвердити особу і отримати паперовий екземпляр, також для зареєстрованих користувачів доступна можливість запису на прийом до лікаря з зручним процесом вибору часу. Для користувачів у яких укладена декларація з сімейним лікарем в особистому кабінеті доступна інформація про історію їх візитів і взаємодій з лікарем, також вони можуть переглянути виписані їм направлення і в якому закладі ці направлення проходять обробку і погашення. Не менш важливим є доступ до результатів діагностичних досліджень і процедур.

У комунальних закладів є доступ до програми доступні ліки, тому пацієнти можуть отримувати безкоштовні ліки за унікальними ідентифікованими рецептами які доступні їм особистому кабінет.

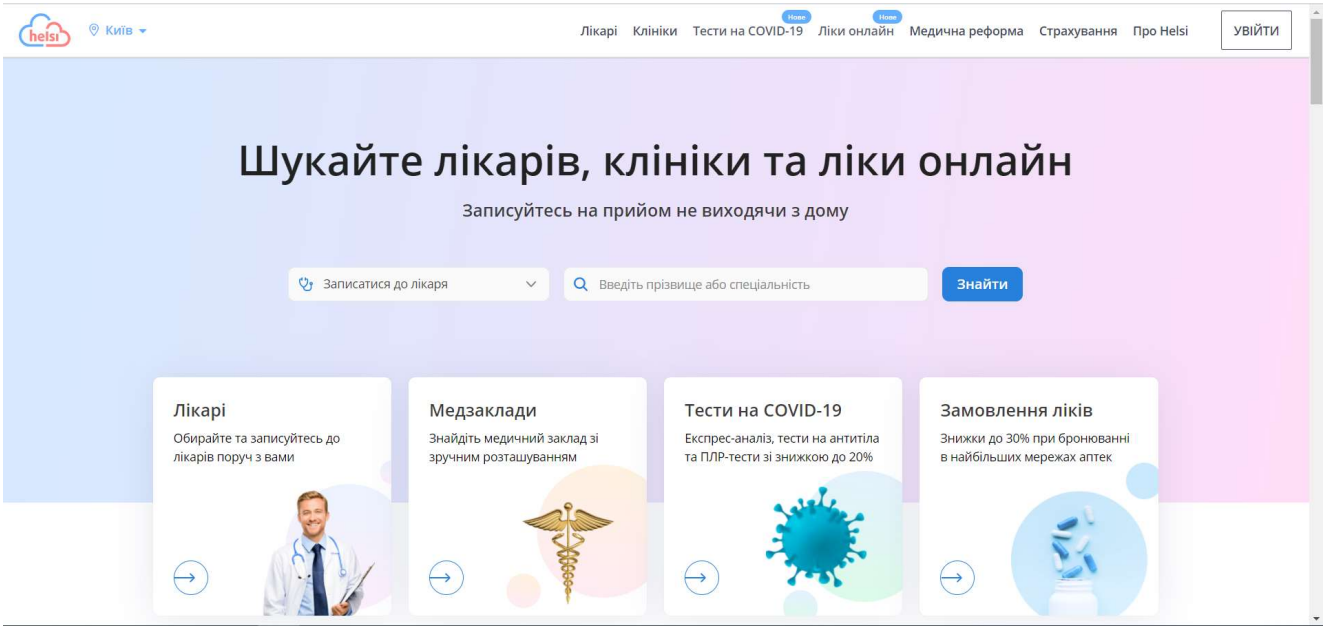


Рисунок 1.3 – Головна сторінка helsi.me

Каталог лікарів є дуже гнучким, в ньому є маса фільтрів для вибору потрібного пацієнту лікаря за типами установи де працює лікар, можливістю

онлайн прийому, географічним районом міста, а також не менш важливою є можливість фільтрація за вартістю прийому «див. рис. 1.4».

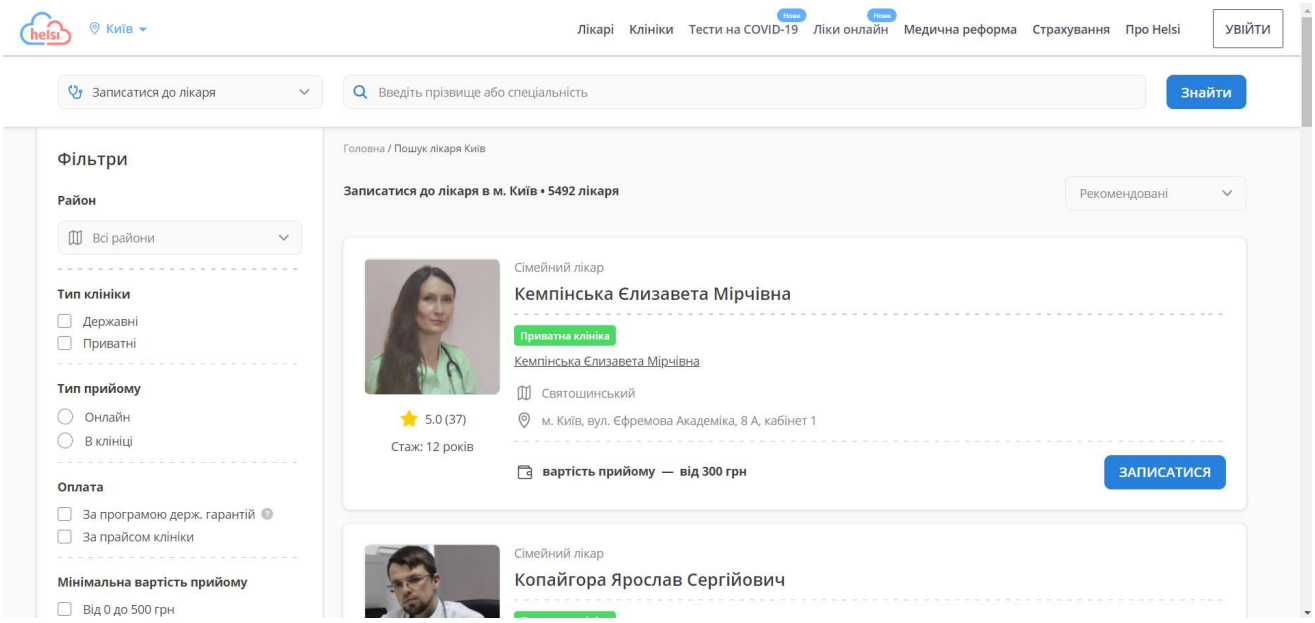


Рисунок 1.4 – Каталог лікарів helsi.me

helsi.pro – це частина системи призначена для лікарів і персоналу медичного закладу. Вона є закритою, тобто доступ через інтернет є можливим, лише для виділених ip-адрес медичних установ які уклали договір з компанією. Основними функціями системи helsi.pro є адміністрування медичного закладу, наприклад створення підрозділів, реєстрація і налаштування профілів лікарів, отримання звітності. Також це основна платформа для ведення прийому лікарем. Доступне ведення прийому в форматі eHealth, з кодуванням діагнозів через систему ICPC2 або МКХ-11.

reform.helsi.me – це частина системи призначена для адміністрації мед закладу. Через цю систему здійснюється інтеграція закладів з центральним компонентом eHealth, а саме реєстрація мед. установи, реєстрація підрозділів

мед. установи та укладення договору з Національною службою здоров'я України «див. рис. 1.5».

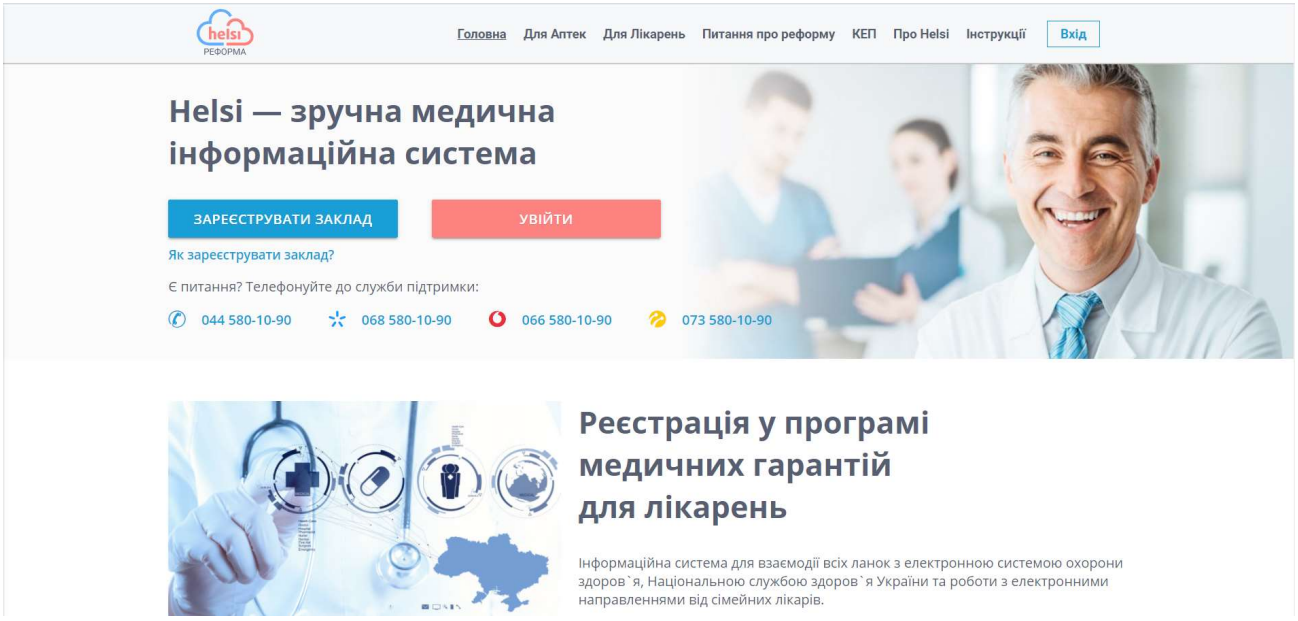


Рисунок 1.5 – Головна сторінка reform.helsi.me

Основними недоліками системи Helsi є:

- Низька швидкість роботи підсистеми helsi.pro.
- Не зрозумілий інтерфейс для лікарів які не мали досвіду роботи з медичними інформаційними системами
- Занадто громіздка для малих установ і ФОПів
- Інтерфейс у трьох підсистемах різний, тому перехід з reform.helsi.me на helsi.pro викликає у лікарів труднощі.

Doctor Eleks - це друга по розповсюдженості медична інформаційна система. Вона з'явилась ще у 2006 році і є однією з перших великих медичних інформаційних систем на території України. Основними функційними модулями системи Doctor Eleks є:

- Електронна медична картка для централізованого збереження інформації про пацієнта

- Розклад для ефективного планування та управління потоком пацієнтів
- Звіти та фінанси для оперативного доступу до аналітичних та статистичних даних
- Інтеграція з діагностичним обладнанням для легкого зберігання та швидкої обробки медичних зображень та відео
- Лабораторія для автоматизації збору, обробки та аналізу отриманих даних
- Послуги та їх оплата для безпечного управління грошовими потоками та контролю фінансів

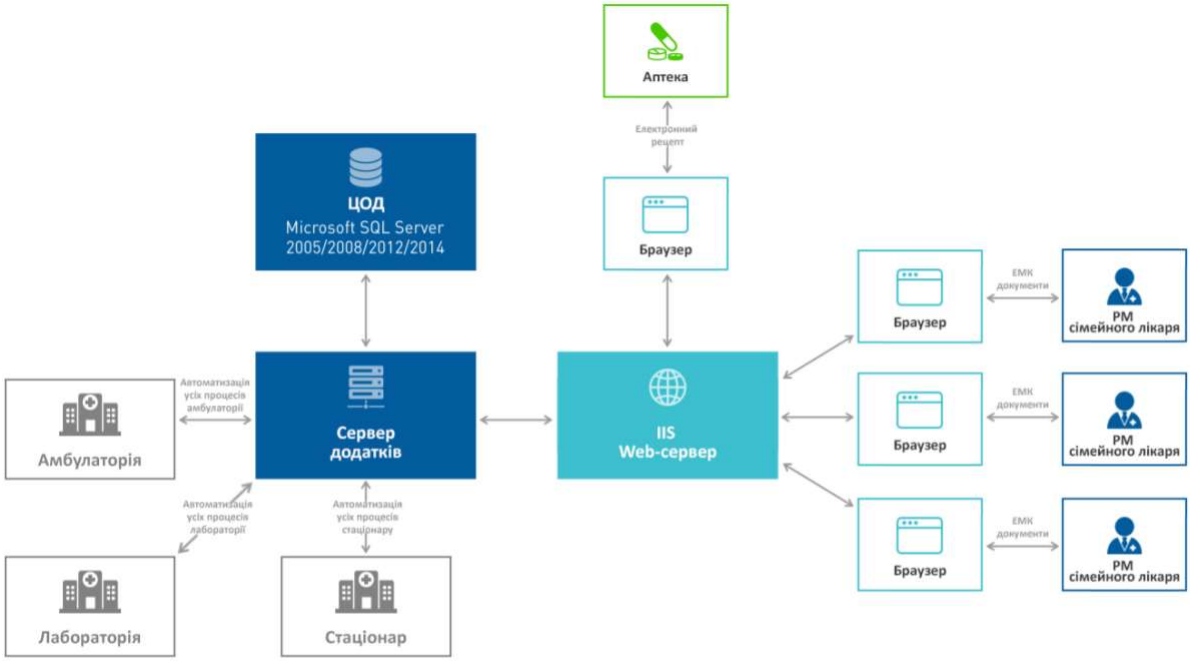


Рисунок 1.6 – Архітектура системи Doctor Eleks

Основними недоліками системи Doctor Eleks є:

- Основний модуль системи для автоматизація робочого місця лікаря розповсюджується у форматі десктопного застосунку під операційну

систему Windows, що робить неможливим його використання на інших платформах

- Десктопний застосунок має застарілий дизайн, незвичний і не зрозумілий для пересічного користувача
- Вартість впровадження системи є досить високою 400 доларів США на одне робоче місце на два роки без урахування витрат на устаткування, сервери та технічну підтримку.

Основним функціоналом медичних інформаційних систем є:

- укладення декларацій між лікарем і пацієнтом
- виписка і погашення рецептів на лікарські засоби включені до програми «доступні ліки»
- проведення прийомів за стандартами ICPC2 та МКХ-10
- виписка і погашення направлень у стаціонар

Всі перераховані вище медичні інформаційні системи реалізовано у форматі веб-застосунку. Вартість використання варіюється від 50 до 400 гривень за робоче місце.

1.4 Обґрунтування мети рішення поставленої задачі

Зробивши висновки з проведеного аналізу наявних медичних інформаційних систем, їх переваг та недоліків, а також враховуючи існуючі вимоги НСЗУ можна зробити наступні висновки:

- всі системи розповсюджуються на платній основі;
- не мають модульної архітектури і тому не можуть бути налаштовані під вимоги конкретного закладу;
- через громіздкість не можуть швидко реагувати на нові вимоги НСЗУ і eHealth.

В ході кваліфікаційної роботи буде реалізовано основу для модульної системи яка дозволить підстроювати функціонал під потреби кожного закладу. Архітектура веб-застосунку дозволить швидко додавати новий функціонал і використовуватиме найсучасніші технології для веб-розробки.

1.5 Постановка задачі. Технічне завдання на розробку

Основною задачею кваліфікаційної роботи є розробка веб-застосунку, що матиме функціональний модуль для проведення лікарських прийомів за стандартом ICPC2 для медичних закладів первинної ланки. Для забезпечення функціоналу буде додано можливість реєстрації користувачів різних ролей, створення медичного закладу, збір статистичних даних по прийомах, запис на прийом пацієнтів і ведення прийому. Для виконання поставленої задачі необхідно вирішити наступні задачі:

- проаналізувати програмно-технічні рішення конкурентів;
- визначити недоліки в реалізації функціоналу у конкурентів та усунути їх у кваліфікаційній роботі;
- створити діаграму роботи з готовим функціоналом веб-застосунку;
- розробити архітектуру застосунку з розділенням на front-end та back-end для простішого подальшого додавання і зміни функціоналу;
- знайти актуальний довідник кодів ICPC2
- спроектувати базу даних
- розробити back-end у форматі REST Api
- створити простий і зрозумілий дизайн;
- розробити front-end за допомогою ReactJS

- створити responsive-design для забезпечення можливості роботи з екранами різної роздільної здатності і різними пристроями;
- перевірити наявність всього функціоналу у веб-застосунок;
- виконати тестування як back-end частини, так і front-end частини веб-застосунку кваліфікаційної роботи.

ВИСНОВКИ

У першому розділі були розглянуті ключові поняття із побудови прикладних інформаційних систем, розглянуті медичні інформаційні системи як один з видів прикладних інформаційних систем, а також проаналізовані дві найбільш розповсюджені в Україні медичні інформаційні системи та виділено їх недоліки.

За допомогою теоретичних відомостей у даному розділі були сформовані основні вимоги до побудови сучасної медичної інформаційної системи, яка призначена для українських медичних закладів охорони здоров’я первинної ланки.

РОЗДІЛ 2
ПРОЕКТНІ І ТЕХНІЧНІ РІШЕННЯ. ВИДИ ЗАБЕЗПЕЧЕННЯ

2.1 Інформаційне забезпечення медичної інформаційної системи

Щоб реалізувати функціональну частину автоматизованої інформаційної системи потрібно мати відповідні ресурси, створені забезпечувальною частиною. До складу забезпечувальної частини АІС входять окремі забезпечувальні підсистеми, об’єднуючи певний вид ресурсів та умови їх організації. Набір ресурсів, необхідних для функціонування АІС, абсолютно типовий, тому вони є практично однаковими для різних АІС

ІЗ (інформаційне забезпечення)- це сукупність методів і засобів створення єдиного інформаційного фонду, систематизація та уніфікація показників і документів, розробка засобів формалізованого опису даних. Інформаційне забезпечення є найважливішим елементом автоматизованої інформаційної системи, що наповнює управлінські задачі конкретним змістом. Від якості розробленого інформаційного забезпечення значною мірою залежить достовірність і ефективність прийнятих управлінських рішень.

ТЗ (технічне забезпечення) - сукупність технічних засобів, що забезпечують функціонування автоматизованої інформаційної системи, і містить пристрої, за допомогою яких виконуються типові операції опрацювання даних. При цьому, опрацювання даних може здійснюватися як поза ЕОМ (периферійні технічні засоби збору, реєстрації, первинного опрацювання інформації, засоби телекомунікації і зв'язку), так і на ЕОМ різних класів. Технічне забезпечення є одним із найважливіших компонентів забезпечувальної частини, які створюють ресурси АІС. Від прогресивності застосовуваних технічних засобів значною мірою залежить рівень автоматизації функцій керування.

Технічні засоби можна поділити на п'ять груп:

1. Засоби збору та реєстрації інформації: пристрої підготовки даних, засоби збору інформації. Ці засоби призначені для приведення інформації у зручний для дистанційної передачі і подальшої обробки вигляд

2. Засоби передачі інформації. Ця група засобів призначена для передачі інформації у просторі. Це ті ж самі пристрої, що перераховані вище, за наявності інтернет з’єднання

3. Засоби зберігання інформації. При наявності інтернет з’єднання більшість даних переноситься на віддалений сервер, щоб не було можливості перевантаження слабких пристроїв. При відсутності з’єднання інформація зберігається на пристрої до появи інтернету.

4. Засоби обробки інформації (засоби обчислювальної техніки з відповідним програмним забезпеченням) складають основу комплексу технічних засобів. Вони призначені для перетворення вихідних даних у результативну інформацію, необхідну для прийняття управлінських рішень. Головні характеристики даної групи - швидкодія й обсяг пам'яті. Виділяють: універсальні - для багатьох користувачів і персональні; спеціалізовані - сервери та робочі станції.

5. Засоби видачі інформації: принтери, монітори - призначені для перетворення інформації у вигляд, зручний для сприйняття користувачем.

ПЗ (програмне забезпечення) - це сукупність програм для ефективної організації обчислювального процесу в автоматизованих інформаційних системах керування.

СПЗ (системне програмне забезпечення) - це сукупність програм, які управляють процесом обробки інформації в обчислювальних системах і забезпечують робоче середовище для функціонування прикладних програм. Базові програмні засоби служать для автоматизації взаємодії людини і

комп'ютера, організації типових процедур опрацювання даних, контролю і діагностики функціонування технічних засобів системи опрацювання даних. До їх складу входять: операційні системи, сервісні програми (оболонки, утиліти, антивірусні засоби), програми технічного обслуговування.

Проаналізувавши вимоги до медичної інформаційної системи, а також наявне у лікарів і пацієнтів ТЗ можна стверджувати що найбільш зручною платформою буде веб-застосунок.

Архітектурою веб-застосунку обрано клієнт-серверну архітектуру

Модель клієнт-сервер - це розподілена структура додатків, яка розділяє завдання або робочі навантаження між постачальниками ресурсу або послуги, що називаються серверами, і запитувачами послуг, що називаються клієнтами. Часто клієнти та сервери спілкуються через комп'ютерну мережу на окремому апаратному забезпеченні, але і клієнт, і сервер можуть знаходитись в одній системі. Хост сервера запускає одну або декілька серверних програм, які діляться своїми ресурсами з клієнтами. Клієнт не ділиться жодним із своїх ресурсів, але він вимагає інформацію від сервера. Таким чином, клієнти ініціюють сеанси зв'язку із серверами, які очікують на вхідні запити[1].

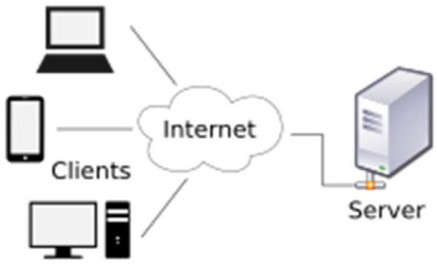


Рисунок 2.1 - Схема роботи клієнт-серверної архітектури.

В якості серверної частини виступає бекенд з архітектурою REST API.

REST розшифровується як Representational State Transfer, а API - Application Program Interface(програмний інтерфейс додатку). REST - це програмний архітектурний стиль, який визначає набір правил, які будуть використовуватися для створення веб-служб. Веб-сервіси, що відповідають архітектурному стилю REST, відомі як RESTful веб-сервіси. Це дозволяє запитувати системи доступу та керувати веб-ресурсами, використовуючи єдиний і заздалегідь визначений набір правил. Взаємодія в системах на базі REST відбувається через протокол передачі гіпертексту в Інтернеті (HTTP).

Система REST складається з:

- клієнт, який запитує ресурси.
- сервер, який має ресурси.
- Важливо створити API REST відповідно до галузевих стандартів, що призводить до полегшення розробки та збільшення кількості клієнтів.

Архітектурні обмеження API RESTful, які обов’язкові для будь-якої веб-служби перераховані нижче:

- Модель клієнт-сервер
- Відсутність стану
- Кешування
- Одноманітність інтерфейсу - Наявність уніфікованого інтерфейсу є фундаментальним вимогою дизайну REST-сервісів. Уніфіковані інтерфейси дозволяють кожному з сервісів розвиватися незалежно.

До уніфікованим інтерфейсів пред'являються наступні чотири обмежувальних умови:

1. Ідентифікація ресурсів - всі ресурси ідентифікуються в запитах, наприклад, з використанням URI в інтернет-системах. Ресурси концептуально відокремлені від уявлень, які повертаються клієнтам. Наприклад, сервер може надсилати дані з бази даних у вигляді HTML, XML або JSON, жоден з яких не є типом зберігання всередині сервера.

2. Маніпуляція ресурсами через представлення - якщо клієнт зберігає уявлення ресурсу, включаючи метадані - він володіє достатньою інформацією для модифікації або видалення ресурсу.

3. «Самодостатні» повідомлення - кожне повідомлення містить достатньо інформації, щоб зрозуміти, яким чином його обробляти. Наприклад, обробник повідомлення (parser), необхідний для отримання даних, може бути зазначений у списку MIME-типів.

4. Гіпермедіа як засіб зміни стану програми (HATEOAS) - клієнти змінюють стан системи тільки через дії, які динамічно визначені в гіпермедіа на сервері (наприклад, гіперпосилання в гіпертексті). Виключаючи прості точки входу в додаток, клієнт не може припустити, що доступна якась операція над якимось ресурсом, якщо не отримав інформацію про це в попередніх запитах до сервера.

- Багатошарова система - Клієнт зазвичай не здатний точно визначити, взаємодіє він безпосередньо з сервером або ж з проміжним вузлом, в зв'язку з ієрархічною структурою мереж (маючи на увазі, що така структура утворює шари). Застосування проміжних серверів здатне підвищити масштабованість за рахунок балансування навантаження і розподіленого кешування. Проміжні вузли також можуть підкорятися політиці безпеки з метою забезпечення конфіденційності інформації.

- Код на вимогу - REST може дозволити розширити функціональність клієнта за рахунок завантаження коду з сервера у вигляді аплетів або сценаріїв. Філдінг стверджує, що додаткове обмеження дозволяє спроектувати архітектуру, підтримуючу бажану функціональність в загальному випадку, але, можливо, за винятком деяких контекстів

Єдиним необов'язковим обмеженням архітектури REST є код на вимогу. Якщо додаток порушує будь-які інші обмеження, його не можна чітко називати RESTful.

Для створення клієнтської частини було обрано архітектуру односторінковий додаток (single-page application, SPA).

Односторінковий додаток (SPA) - це веб-додаток або веб-сайт, який взаємодіє з веб-браузером, динамічно переписуючи поточну веб-сторінку новими даними з веб-сервера, замість стандартного способу, тобто завантаження цілих нових сторінок. Мета - швидші переходи, які дозволяють веб-сайту відчувати себе як рідний додаток.

У SPA всі необхідні HTML, JavaScript та CSS-код отримують браузер із завантаженням однієї сторінки, або відповідні ресурси динамічно завантажуються та додаються на сторінку за необхідності, як правило, у відповідь на дії користувача. Сторінка не завантажується в будь-який момент процесу, а також не передає керування на іншу сторінку, хоча хеш місцеположення або API історії HTML5 можна використовувати для забезпечення сприйняття та навігації окремих логічних сторінок у програмі.

SPA повністю завантажується при початковому завантаженні сторінки, а потім частини сторінок замінюються або оновлюються новими фрагментами сторінок, завантаженими з сервера на вимогу. Щоб уникнути зайвого завантаження невикористаних функцій, SPA часто поступово завантажуватиме

більше функцій, коли вони стануть необхідними, або невеликі фрагменти сторінки, або повно-екранні модулі.

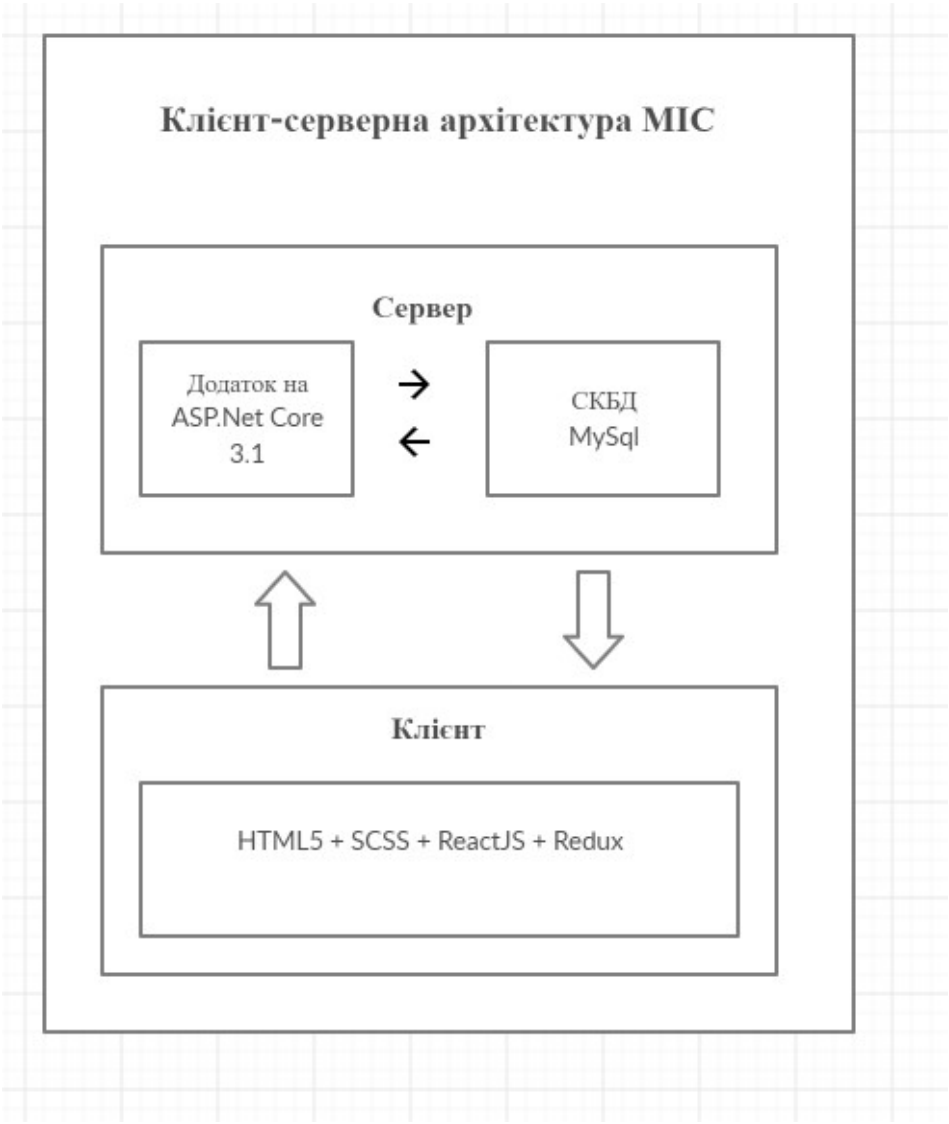


Рисунок 2.2 - Схема роботи клієнт-серверної архітектури у медичній інформаційній системі

Таким чином існує аналогія між "станами" в SPA та "сторінками" на традиційному веб-сайті. Оскільки "навігація за станом" на одній сторінці аналогічна навігації по сторінках, теоретично будь-який веб-сайт на основі

сторінки може бути перетворений на односторінковий, замінивши на одній сторінці лише змінені частини.

2.2 Стек використаних технологій

2.2.1 Серверна частина

Для реалізації серверного застосунку з RESTful архітектурою було обрано мову **C# 8.0** і фреймворк **ASP.Net Core 5.0**. У якості бази даних обрано СУБД MySql.

C # - об'єктно-орієнтована мова програмування. C # відноситься до сім'ї мов з C-подібним синтаксисом, з них його синтаксис найбільш близький до C ++ і Java. Мова має статичну типізацію, підтримує поліморфізм, перевантаження операторів (в тому числі операторів явного і неявного приведення типу), делегати, атрибути, події, властивості, узагальнені типи і методи, ітератори, анонімні функції з підтримкою замикань, LINQ, виключення, коментарі у форматі XML[4].

ASP.NET Core - вільно розповсюджений крос-платформенний фреймворк для створення веб-додатків з відкритим вихідним кодом. Дана платформа розроблена компанією Майкрософт і має більшу швидкодію порівняно з ASP.NET. Має модульну структуру та сумісний з такими операційними системами як Windows, Linux та macOS.

Не зважаючи на те, що це новий фреймворк, побудований на новому веб-стеці, він має високу сумісність з концепціями з ASP.NET. Додатки ASP.NET Core підтримують паралельне управління версіями, в яких є різні програми, які працюють на одночасних комп'ютерах, і можуть орієнтуватися на різні версії ASP.NET Core. Це було неможливо в попередній версії ASP.NET.

У якості **СУБД** було обрано **MySQL**, його було порівняно з **MongoDB**.

MySQL – вільна система управління базами даних, яка є власністю компанії Sun Mycrosystem. СКБД розповсюджується по ліцензії GNU General Public License та під власною комерційною ліцензією. За умовами GPL, якщо якась програма вміщує вихідні коди MySQL, то вона також повинна розповсюджуватися за ліцензією GPL. Останнє може не відповідати планам розробників, які не бажають відкривати вихідні коди своїх програм. Для таких випадків передбачена комерційна ліцензія, яка, крім того, забезпечує якісну сервісну підтримку. Крім того компанія розробляє різноманітну функціональність на замовлення ліцензійних користувачів. Завдяки одній із таких розробок ще в самих перших версіях з'явився механізм реплікації.

MySQL переважно використовується для нескладних інформаційних систем. У більшості випадків MySQL використовується як сервер, до якого звертаються локальні та віддалені клієнти, хоча до дистрибутиву також входить бібліотека внутрішнього сервера, яка дозволяє включати MySQL в автономні програми. MySQL підтримує багато типів таблиць, вона є мультиплатформенною СУБД. Завдяки широким можливостям, а також великій швидкодії MySQL часто використовується в якості СУБД, що забезпечує роботу динамічного Web-сайту в мережі Інтернет.

Порівняння реляційних і не реляційних баз даних

Мова

Уявіть собі місто - нехай він називається Місто А, де всі говорять на одній мові. Всі справи ведуться на ньому, він використовується в будь-якій формі комунікації - в цілому це єдиний засіб взаємодії і взаєморозуміння для мешканців міста. Зміна мови в кожній із сфер діяльності зіб'є всіх з пантелику.

Тепер уявіть Місто Б, де всі мешканці розмовляють різними мовами. Вони абсолютно по-різному взаємодіють з навколишнім світом, і для них не існує «універсального» кошти спілкування[7, 8].

Ці два приклади наочно демонструють відмінності між реляційними та нереляціоннимі базами даних, і за цими відмінностями ховаються ключові особливості обох СУБД.

- Реляційні бази даних використовують структурований мову запитів (Structured Query Language, SQL) для визначення і обробки даних. З одного боку, це відкриває великі можливості для розробки: SQL один з найбільш гнучких і поширених мов запитів, так що його вибір дозволяє мінімізувати ряд ризиків, і буде особливо до речі, якщо має бути робота з комплексними запитами. З іншого боку, в SQL є ряд обмежень. Побудова запитів на цій мові зобов'язує визначати структуру даних і, як у випадку з Містом А, подальша зміна структури даних може бути згубним для всієї системи.Нереляційні бази даних, в свою чергу, пропонують динамічну структуру даних, які можуть зберігатися кількома способами: орієнтовано по колонках, документо-орієнтовано, в вигляді графів або на основі пар «ключ-значення». Така гнучкість означає наступне:
- Ви можете створювати документи, не ставлячи їх структуру заздалегідь;
- Кожен документ може мати власну структурою;
- У кожної бази даних може бути власний синтаксис;
- Ви можете додавати поля прямо під час роботи з даними.

Масштабованість

У більшості випадків SQL бази даних вертикально масштабовані, тобто ви можете збільшувати навантаження на окремо взятий сервер, нарощуючи потужність центральних процесорів, обсяги ОЗУ або системи зберігання даних. А NoSQL бази даних горизонтально масштабованих. Це означає, що ви можете збільшувати трафік, розподіляючи його або додаючи більше серверів до вашої

СУБД. Все одно, що додавати більше поверхів до вашого будинку, або додавати більше будівель на вулицю. У другому випадку, система може стати куди більші і потужніші, роблячи вибір NoSQL бази даних віддається перевага для великих або постійно мінливих структур даних.

Структура

У реляційних СУБД дані представлені у вигляді таблиць, в той час як в нереляційних - у вигляді документів, пар «ключ-значення», графів або wide-column сховищ.

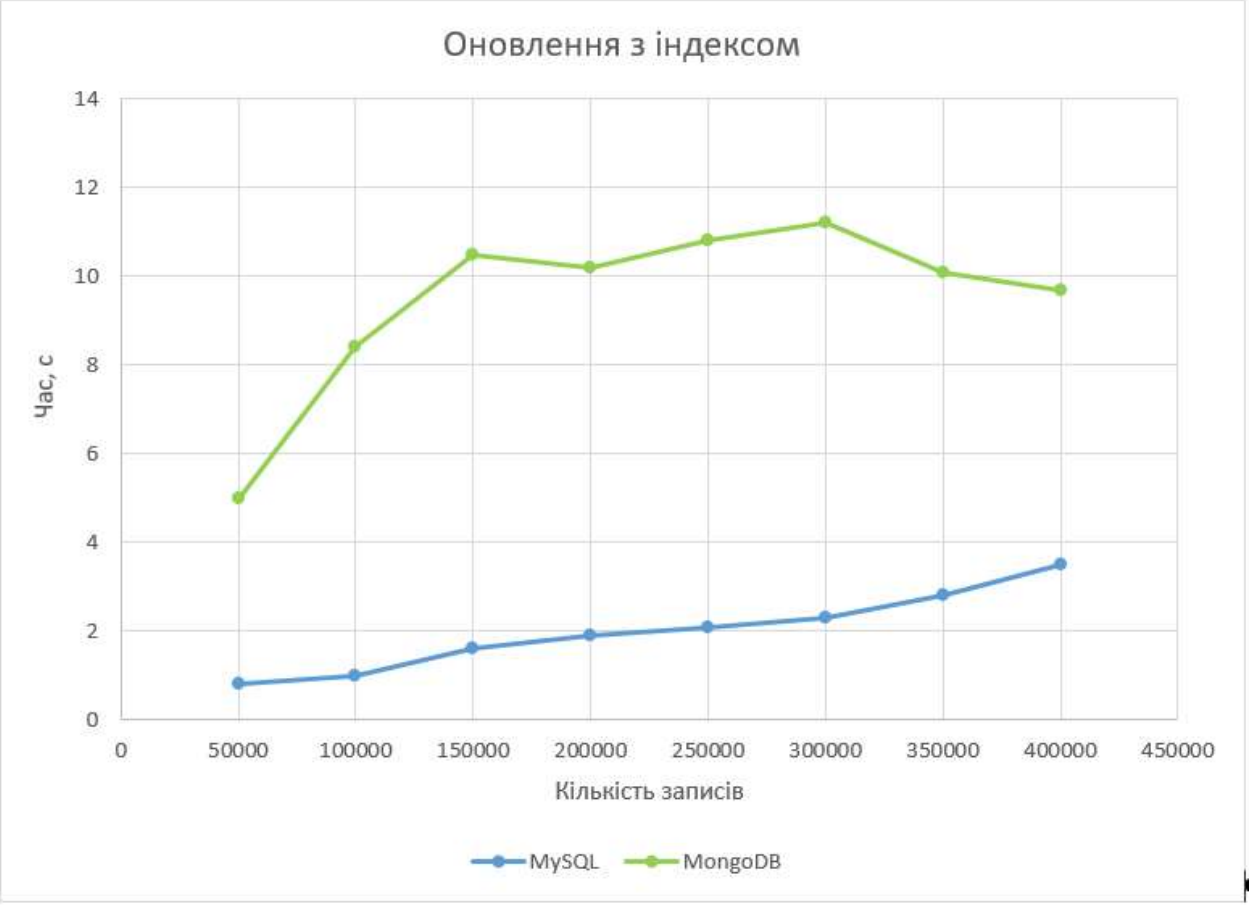


Рисунок 2.3 - Порівняння швидкості оновлення даних з індексом у СУБД MySQL і MongoDB

Це робить SQL бази даних кращим вибором для додатків, які передбачають транзакції з декількома записами - як, наприклад, система облікових записів - або для застарілих систем, які були побудовані для реляційних структур.

З діаграми на рис. 2.3 видно, що не реляційні бази даних, на прикладі MongoDB, значно повільніші при оновленні даних з індексом навіть на невеликих кількостях.

В основному в веб-додатках найпоширенішою операцією є читання даних. Тобто слід порівняти швидкість вибірки даних.

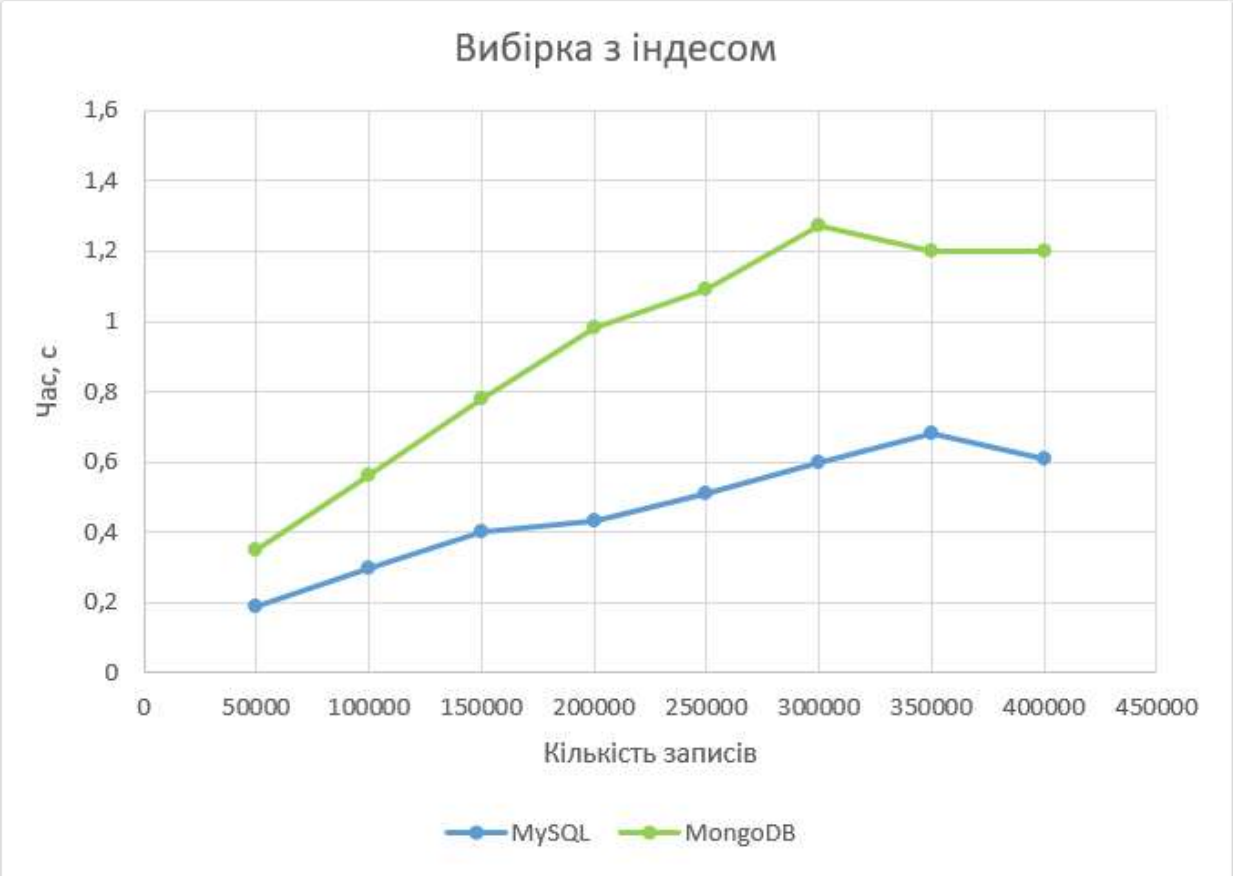


Рисунок 2.4 - Порівняння швидкості вибірки даних з індексом у СУБД MySQL і MongoDB

Отже, виходячи з результатів наведених вище порівнянь було обрано СУБД MySQL через швидкодію, велику розповсюдженість, широку підтримку і високу сумісність з об’єктно-орієнтованими мовами програмування.

2.2.2 Клієнтська частина

Для розробки клієнтської частини використовувались HTML5 SASS, JavaScript, React та Redux.

HTML (Hyper Text Markup Language) − це стандартна мова, яка призначена для створення гіпертекстових документів для середовища web сайтів. HTML документи можливо переглядати на різних пристроях та браузерів. Створений HTML документ браузер може інтерпретувати для виділення різних елементів та первинної їх обробки. Саме використання HTML дозволяє виконувати форматування документів для їх показу з використанням шрифтів, ліній та інших графічних елементів на будь−якому програмному забезпеченні та для перегляду web−сторінок.

Для виконання стилізування HTML документів використовуються каскадні таблиці стилів (**CSS**) − це технологія описування зовнішнього вигляду документа, який написаний мовою гіпертекстової розмітки HTML.

CSS використовується авторами веб−сторінок для вказування кольорів, шрифтів, розташування блоків та інших аспектів представлення документа. Основною метою CSS є розділення вмісту (написаного на HTML) та подання документа (написаного на CSS). Цей поділ документа збільшує доступність документа, надає велику гнучкість та можливість виконувати управління його показу, а також зменшити складність та повторюваність в структурі самого документа.

Sass (Syntactically Awesome Stylesheets) - модуль, що входить в Haml. Sass - це метамова на основі CSS, призначена для збільшення рівня абстракції CSS коду та спрощення файлів каскадних таблиць стилів.

Мова Sass має два синтаксису:

sass - відрізняється відсутністю фігурних дужок, в ньому вкладені елементи реалізовані за допомогою відступів;

SCSS (Sassy CSS) - використовує фігурні дужки, як і сам CSS.

Для створення інтерфейсу користувача буде використана технологія **ReactJS** − це бібліотека JavaScript, яка використовується для створення користувацьких інтерфейсів. Вона була створена компанією Facebook, а перший реліз самої бібліотеки був в березні 2013 року[5].

З самого початку React був призначений для веб−сайтів, проте пізніше з'явилась платформа React Native, яка була призначена для створення додатків під мобільні пристрої.

React представляється ідеальним інструментом для створення масштабних веб−сайтів, особливо в тих ситуаціях, коли веб−сайт має лише одну сторінку (landing page).

Вся структура веб−сторінки може бути представлена завдяки допомозі DOM − організації HTML елементів, якими ми можемо маніпулювати, змінювати, видаляти або додавати нові. Для самої взаємодії з DOM застосовується мова JavaScript. Однак коли ми намагаємося маніпулювати HTML−елементами, то можемо зменшити продуктивність веб−сторінки, особливо коли змінюємо велику кількість елементів. Через те, що операції над елементами можуть зайняти деякий час, це неминуче позначиться на призначеному для користувача досвіді. Саме для розв'язання проблеми продуктивності ми використовуємо концепцію віртуального DOM. Віртуальний DOM − це легковажна копія звичайного DOM. Тому React працює виключно з віртуальним DOM, а не звичайним.

Якщо нам необхідно змінити елементи веб−сторінки, то зміни спочатку задаються в віртуальний DOM. Після цього новий стан віртуального DOM

зрівнюється з поточним станом. Якщо ці стани відрізняються, то React виконує мінімальну кількість маніпуляцій, які необхідні для показу реального DOM до нового стану і виконує їх.

В результаті така схема взаємодії з елементами веб−сторінки працює набагато швидше та ефективніше, ніж якби ми працювали з JavaScript з DOM безпосередньо.

Redux - бібліотека для JavaScript з відкритим вихідним кодом, призначена для управління станом додаток. Частіше використовується в зв'язці з React або Angular для розробки клієнтської частини. Містить ряд інструментів, що дозволяють значно спростити передачу даних сховища через контекст.

2.3 Функціональні можливості медичної інформаційної системи

Для вирішення завдань проектованої системи було проаналізовано декілька якісних наявних додатків з операціями та можливостями ведення сімейного бюджету та виділено функціонал, який найкраще підходить для проектованої системи, не ускладнює роботу користувачу та не зменшує продуктивність всього веб−сайту.

Таблиця 2.1 - Функціонал веб−додатку «Медична інформаційна система»

Назва функціоналу	Опис функціоналу
Авторизація користувача	Для можливості ідентифікації користувачів та збереженням їх даних на сервері кожен користувач повинен пройти авторизації та підтвердити свої реєстраційні дані.

Продовження таблиці 2.1

Реєстрація	Функція яка дозволяє створювати облікові записи для кожного користувача з розділенням по ролям.
Створення медичного закладу	Функція створення медичного закладу дозволяє керівнику зареєструвати свій заклад.
Додавання лікарів до медичного закладу	Функція додавання лікарів до медичного закладу дозволяє керівнику зареєструвати свій заклад.
Редагування медичного закладу	Функція редагування гаманця дозволяє проводити зміни в назві адресі та інших властивостях медичного закладу.
Пошук лікарів	Функція пошуку лікарів дозволяє пацієнтам знаходити лікарів за прізвищем, а також фільтрувати результати.
Назва функціоналу	Опис функціоналу
Запис на прийом до лікаря	Пацієнт може записатись на прийом до лікаря

Продовження таблиці 2.1

Скасування прийому	Пацієнт або лікар можуть скасувати запис на прийом
Ведення прийому	Лікар може провести прийом з пацієнтом який був записаний до нього. В рамках ведення прийому лікар може додавати причини, діагнози, послуги з кодуванням ICPC2

2.4 Схема бази даних

База даних − це сукупність структур, які призначені для зберігання великих обсягів інформації та програмних модулів, що здійснюють управління даними, їх вибірку, сортування та інші подібні дії. Для виконання кваліфікаційної роботи було створено декілька таблиць:

- Користувачі – таблиця для зберігання облікових записів користувачів, загальних даних про них (ПІБ, стать, дата народження, телефон)
- Організації - таблиця для зберігання інформації про організації, їх адресу і код ЄДРПОУ
- Прийоми – таблиця для зберігання інформації про прийоми зі зв’язком з таблицею користувачів
- Довідник ICPC2 – таблиця для зберігання довідника ICPC2 кодів

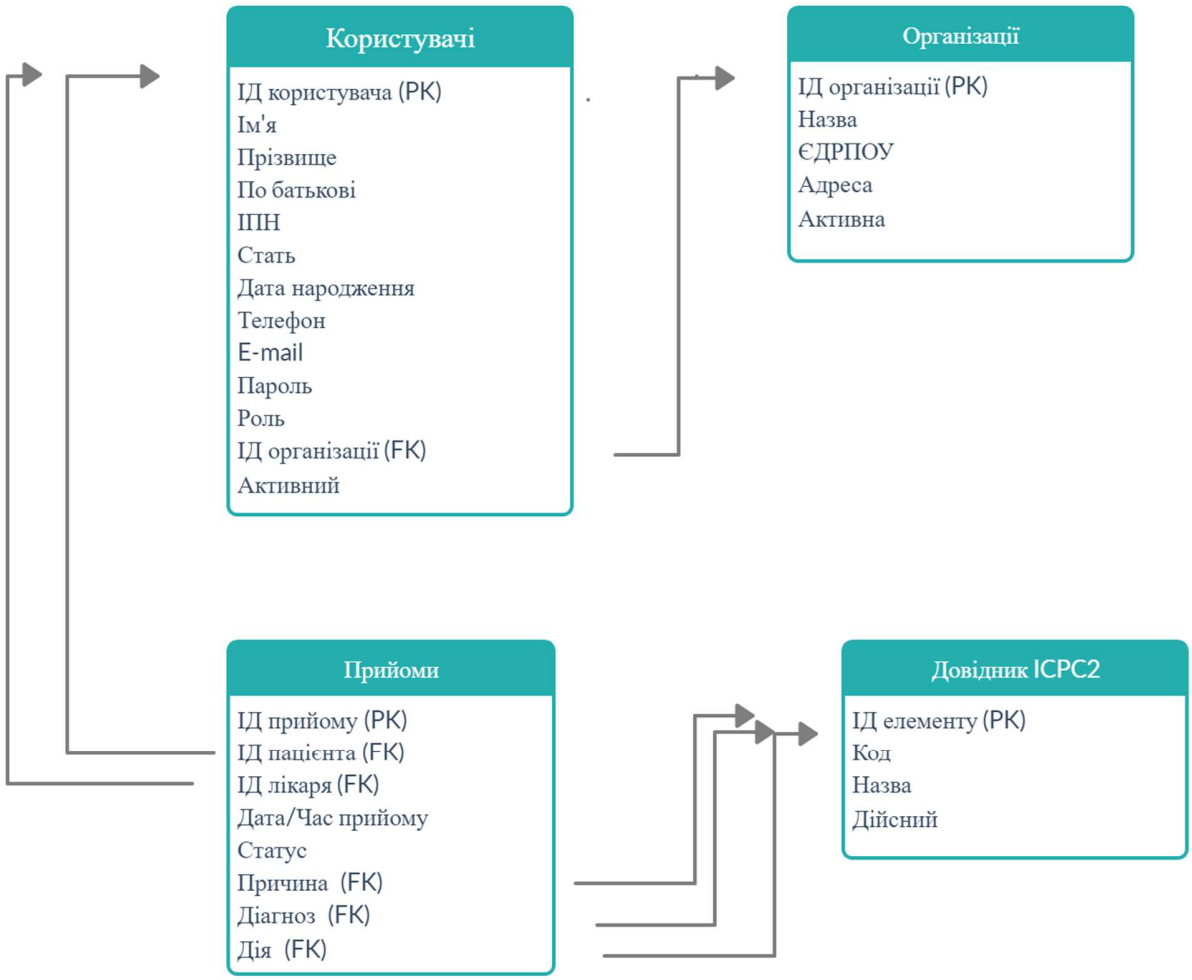


Рисунок 2.5 - Схема бази даних

Вся інформація складається з набору однотипних записів розташованих один за одним. Дані в цих таблицях можливо додавати, видаляти або змінювати. Також в кожній таблиці є набір іменованих полів, які зберігають корисну інформацію починаючи від імені користувача та закінчуючи довідником ICPC2 кодів. Детальну схему створеної бази даних можливо побачити на «див. рис. 2.5»

ВИСНОВКИ

У другому розділі були розглянуті ключові поняття із побудови сучасних веб-застосунків. Були проаналізовані особливості архітектури веб-застосунків, основні поняття протоколів взаємодії у клієнт-серверній архітектурі.

Для створення веб-застосунку прикладної медичної інформаційної системи було обрано сучасні технології. У якості інструменту для розробки серверної частини було обрано фреймворк ASP.NET Core і мову програмування C#. В ході вибору СУБД було розглянуто реляційні і документ-орієнтовані бази даних на прикладі двох розповсюджених СУБД MySQL та MongoDB. Порівняння і тестування швидкодії показало, що реляційні СУБД більше підходять для використання у високонавантаженій медичній інформаційній системі. Для реалізації клієнтської частини веб-застосунку було обрано бібліотеку ReactJS

РОЗДІЛ 3
РЕАЛІЗАЦІЯ І ОПИС РОБОТИ СИСТЕМИ

3.1 Особливості реалізації системи

В цьому підрозділі будуть розглянуті деякі архітектурно-програмні рішення прийняті під час розробки системи.

3.1.1 Патерн проектування фасад

У серверній частині веб-застосунку для реалізації механізму генерації і отримання доступних часових проміжків для запису на прийом до лікаря було застосовано один з популярних патернів проектування – фасад. За допомогою патерну фасад можна істотно спростити доступ до системи класів або методів. Це допомагає розділити шари логіки застосунку. Тобто фасад надає простий інтерфейс взаємодії з деякою підсистемою. У випадку підсистеми генерації проміжків часу для запису фасад дозволяє приховати всю реалізацію від шару взаємодії (API) тим самим розділяючи рівні API і бізнес-логіки, що в подальшому спростить підтримку, розширення і рефакторинг системи.

```
namespace HealthyCountry.Services
{
    3 usages  1 inheritor  Oleksandr Panasiuk  1 exposing API
    public interface IAppointmentFacade
    {
        17 usages  1 implementation  Oleksandr Panasiuk
        Task<List<Appointment>> GetAppointmentsAsync(string doctorId, DateTime dateFrom);
    }
}
```

Рисунок 3.1 – Інтерфейс фасаду отримання проміжків часу для запису

Рівень API отримує доступ до підсистеми генерації і отримання доступних часових проміжків для запису на прийом до лікаря через метод GetAppointmentsAsync інтерфейсу IAppointmentFacade

```
public class AppointmentService : IAppointmentFacade
{
    private readonly ApplicationDbContext _dbContext;

    1 usage  Oleksandr Panasiuk
    public AppointmentService(ApplicationDbContext dbContext)
    {
        _dbContext = dbContext;
    }

    0+17 usages  Oleksandr Panasiuk
    public Task<List<Appointment>> GetAppointmentsAsync(string doctorId, DateTime dateFrom)
    {
        var numberOfSlotsToCreate :int = GetNumberOfSlotsToCreate(doctorId, dateFrom);
        if (numberOfSlotsToCreate != 0)
        {
            CreateAppointmentSlots(doctorId, dateFrom, numberOfSlotsToCreate);
        }

        CreateSlotsForCancelledAppointments(doctorId, dateFrom);
        return _dbContext.Appointments.AsNoTracking()
            .Where(x :Appointment => x.EmployeeId == doctorId && x.DateTime.Date >= dateFrom.Date && x.Status!=AppointmentStatuses.CANCELED)
            .OrderBy(x :Appointment => x.DateTime).ToListAsync(); // Task<List<...>>
    }
```

Рисунок 3.2 – Клас і метод що реалізують інтерфейс фасаду

Клас AppointmentService реалізує інтерфейс фасаду, надаючи рівню API лише одну просту точку входу до підсистеми бізнес-логіки «див. додаток А».

3.1.2 Універсальний компонент вводу

У клієнтській частині веб-застосунку для спрощення роботи з формами вводу було розроблено універсальний компонент вводу. Основним функціоналом компоненту є створення відповідного HTML елементу форми в залежності від вхідних параметрів.

```
import React from "react";
import DatePicker from "react-datepicker";
import Select from "react-select";
import AsyncSelect from "react-select/async";
import "./_input.scss";
import "react-datepicker/dist/react-datepicker.css";
const colourStyles = {···
};
const Input = (props) => {
  let inputElement = null;
  const inputClasses = ["inputElement"];
  console.log(props.elementType + " " + props.value);
  if (props.className) inputClasses.push(props.className);
  if (props.invalid && props.shouldValidate && props.touched) {
    inputClasses.push("invalid");
  }

  switch (props.elementType) {
    case "input":
      inputElement = (
        <input
          className={inputClasses.join(" ")}
          {...props.elementConfig}
          value={props.value}
          onChange={props.changed}
        />
      );
      break;
    case "textarea":
      inputElement = (
        <textarea
          className={inputClasses.join(" ")}
          {...props.elementConfig}
          value={props.value}
          onChange={props.changed}
        />
      );
      break;
    case "select":
      inputElement = (
        <Select
          options={props.elementConfig.options}
          onChange={props.changed}
          value={props.value}
          styles={colourStyles}
          placeholder={props.elementConfig.placeholder}
        />
      );
      break;
```

Рисунок 3.3 – Універсальний компонент вводу

За допомогою властивостей які передаються через props компонент можна конфігурувати як будь-яке з HTML полів для вводу, а також були додані окремі опції для асинхронного селекту, тобто випадаючого списку де опції завантажуються динамічно з серверу коли користувач вводить символи.

```
12  export class Login extends Component {
13    static propTypes = {};
14    state = {
15      controls: {
16        email: {
17          label: "Email",
18          elementType: "input",
19          elementConfig: {
20            type: "email",
21            placeholder: "Логін",
22          },
23          value: "",
24          validation: {
25            required: true,
26            isEmail: true,
27          },
28          valid: false,
29          touched: false,
30        },
31        password: {
32          label: "Пароль",
33          elementType: "input",
34          elementConfig: {
35            type: "password",
36            placeholder: "Пароль",
37          },
38          value: "",
39          validation: {
40            required: true,
41            minLength: 8,
42          },
43          valid: false,
44          touched: false,
45        },
46      },
```

Рисунок 3.4 – Приклад конфігурації універсальних компонентів вводу для форми логіну

3.2 Інструкція користувача

Для веб-застосунку розробленого для пацієнтів та лікарів було обрано мінімалістичний дизайн для зручності користування. Для користування системою користувачу необхідно увійти у свій обліковий запис. Система дозволяє директору зареєструвати свій мед. заклад і додати співробітників. Для пацієнтів створено окремий зручний інтерфейс реєстрації.

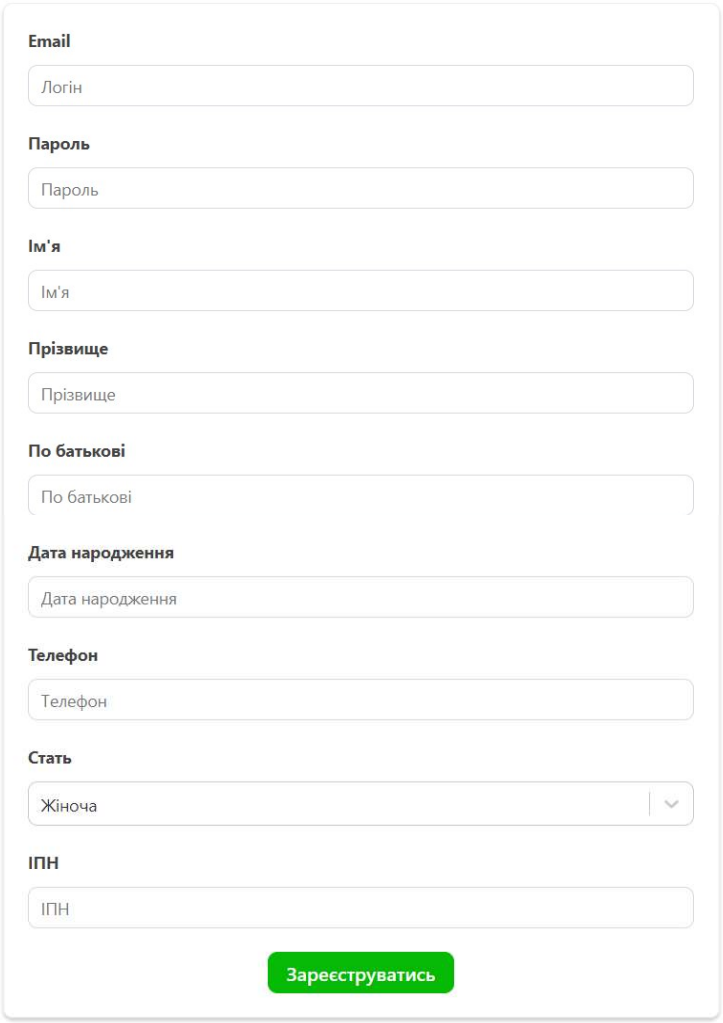


Рисунок 3.5 - Сторінка реєстрації.

Після реєстрації користувачу необхідно авторизуватись у системі створеним обліковим записом. На рис. 3.6 зображений єдиний інтерфейс авторизації користувача.

Рисунок 3.6 - Сторінка авторизації.

Після авторизації користувач потрапляє на головну сторінку і бачить пошук лікарів за прізвищем і назвою організації. На кожній сторінці відображається по до 30 профілів лікарів, до яких користувач може записатись на прийом. Переключитись на наступну сторінку можна за допомогою навігаційних кнопок «вперед» і «назад».

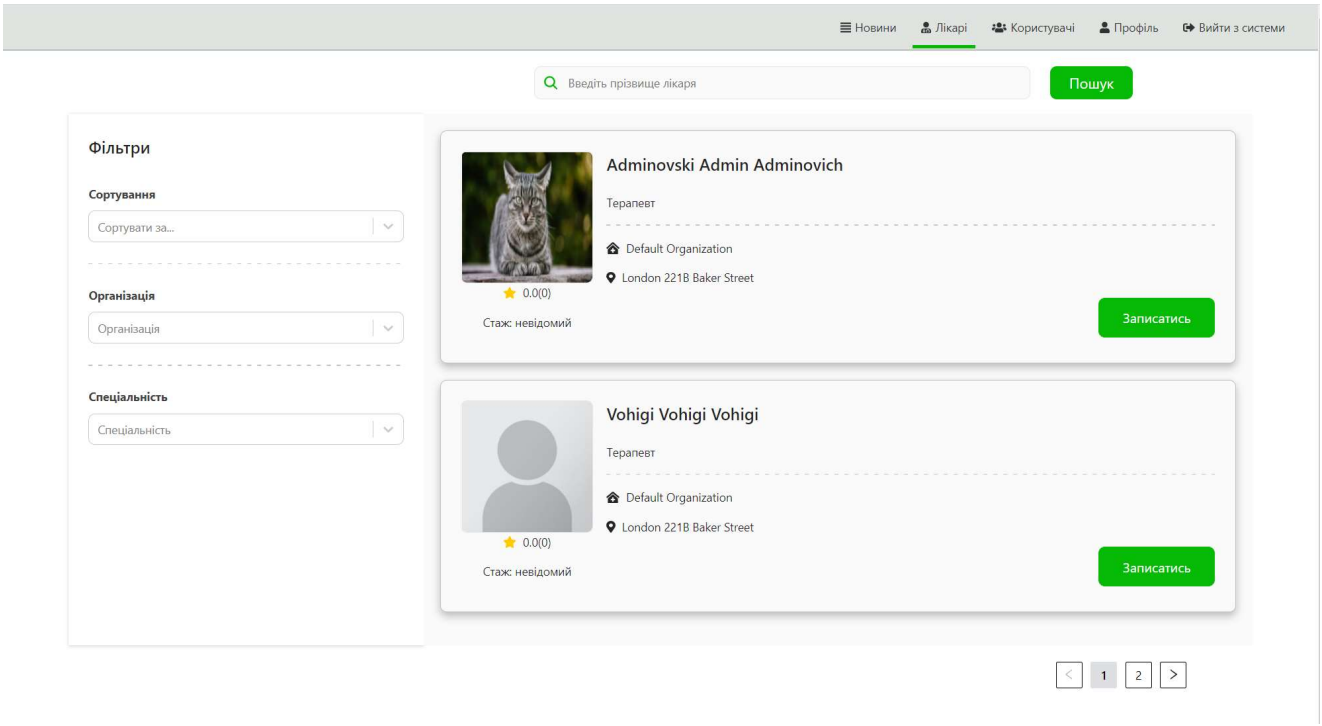


Рисунок 3.7 - Головна сторінка.

Для детального перегляду даних про лікаря і запису на прийом користувач повинен натиснути на кнопку «Записатись на прийом» у блоці препарату головної сторінки. Для нього відкриється сторінка лікаря, де буде відображена вся доступна інформація про організацію і лікаря, його телефон, а також можливість обрати час і день для запису на прийом.

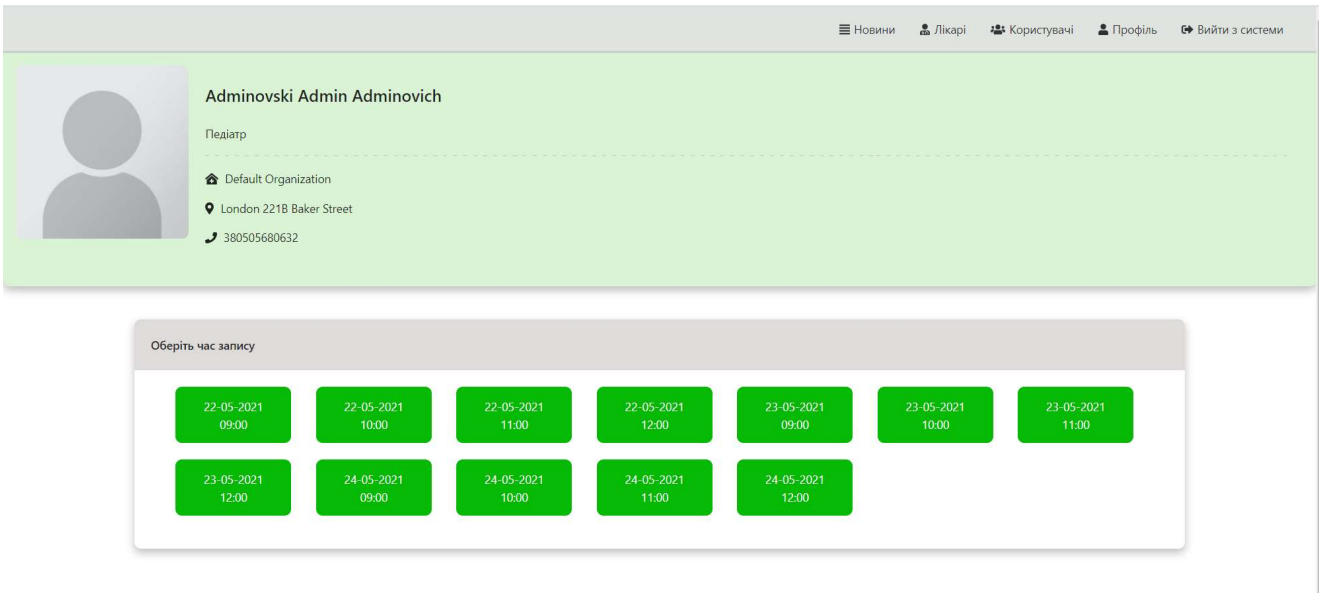


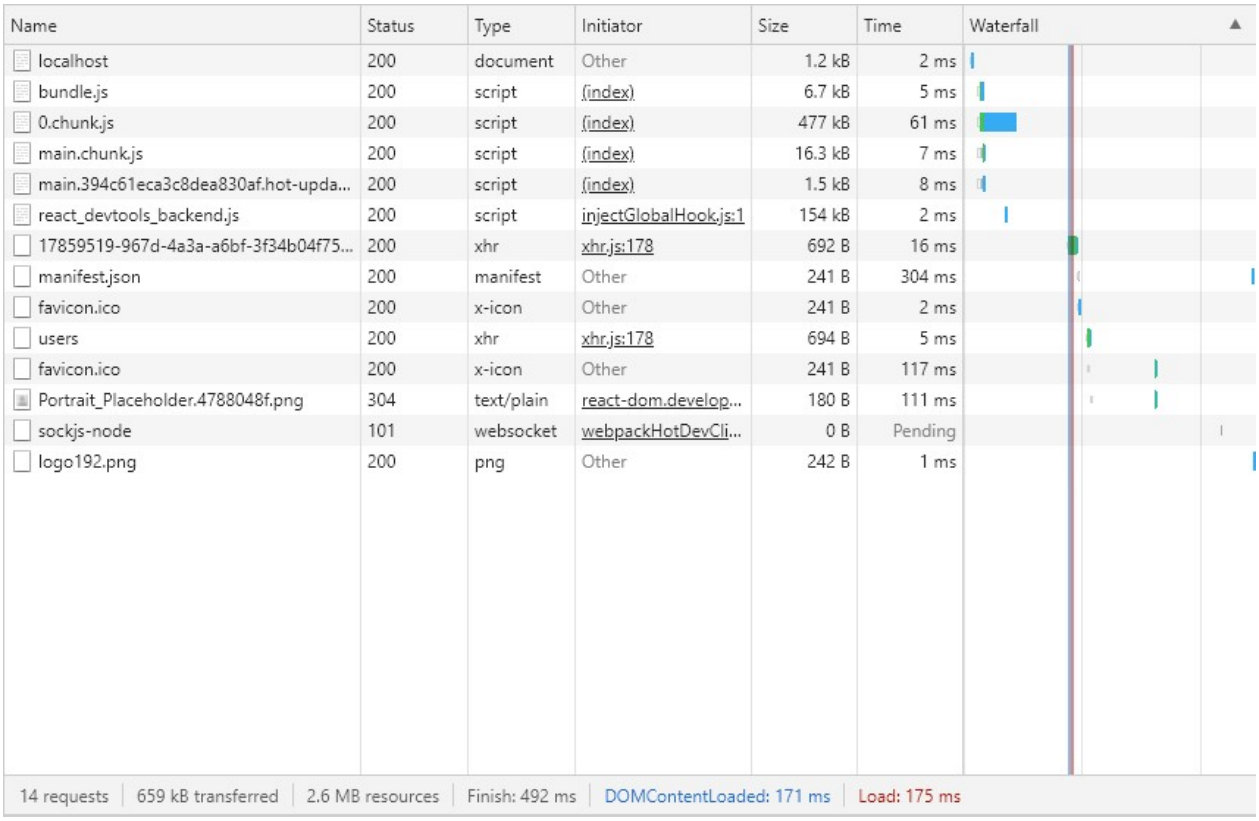
Рисунок 3.8 - Сторінка лікаря з записом на прийом.

Після натискання на обрану дату пацієнт може переглянути свої записи у своєму профілі і якщо, щось змінилось скасувати прийом.

3.3 Результати аудита додатку

Для виконання тестування веб−сайту було використано браузер Google Chrome з інструментами розробника а саме, "Network" та "Lighthouse".

Перше тестування націлене на отримання загальної швидкості завантаження веб−сторінки профілю лікаря. Дане тестування показало результат завантаження сторінки с повним завантаженням всіх зображень, шрифтів, текстів та стилів – 0.5 сек. Сайт має вагу 659 Кб. Такі результати є дуже добрими, тому що загальна швидкість завантаження звичайного сайту − 3 сек, та вага не менше 5 Мб



Name	Status	Type	Initiator	Size	Time	Waterfall
localhost	200	document	Other	1.2 kB	2 ms	
bundle.js	200	script	(index)	6.7 kB	5 ms	
0.chunk.js	200	script	(index)	477 kB	61 ms	
main.chunk.js	200	script	(index)	16.3 kB	7 ms	
main.394c61eca3c8dea830af.hot-upda...	200	script	(index)	1.5 kB	8 ms	
react_devtools_backend.js	200	script	injectGlobalHook.js:1	154 kB	2 ms	
17859519-967d-4a3a-a6bf-3f34b04f75...	200	xhr	xhr.js:178	692 B	16 ms	
manifest.json	200	manifest	Other	241 B	304 ms	
favicon.ico	200	x-icon	Other	241 B	2 ms	
users	200	xhr	xhr.js:178	694 B	5 ms	
favicon.ico	200	x-icon	Other	241 B	117 ms	
Portrait_Placeholder.4788048f.png	304	text/plain	react-dom.develop...	180 B	111 ms	
sockjs-node	101	websocket	webpackHotDevCli...	0 B	Pending	
logo192.png	200	png	Other	242 B	1 ms	

14 requests | 659 kB transferred | 2.6 MB resources | Finish: 492 ms | DOMContentLoaded: 171 ms | Load: 175 ms

Рисунок 3.9 - Тестування швидкості завантаження та ваги веб−сайту

Другий тест орієнтований на отримання результатів щодо семантично правильного використання HTML тегів, оцінці швидкості, навантаження на систему та оптимізації всього веб−сайту. За результатами тестування в інструменті "Lighthouse" було отримано наступні значення «див. рис. 3.10»:

- Perfomance (продуктивність) − 93/100
- Accessibility (доступність) − 97/100
- Best Practices(використання найкращих рішень) − 93/100
- SEO (оптимізація під пошукові системи) – 100/100

Ці дані говорять, що веб−сайт відповідає всім нормам та правилам створення веб−додатків та буде частіше пропонуватись в пошукових системах.

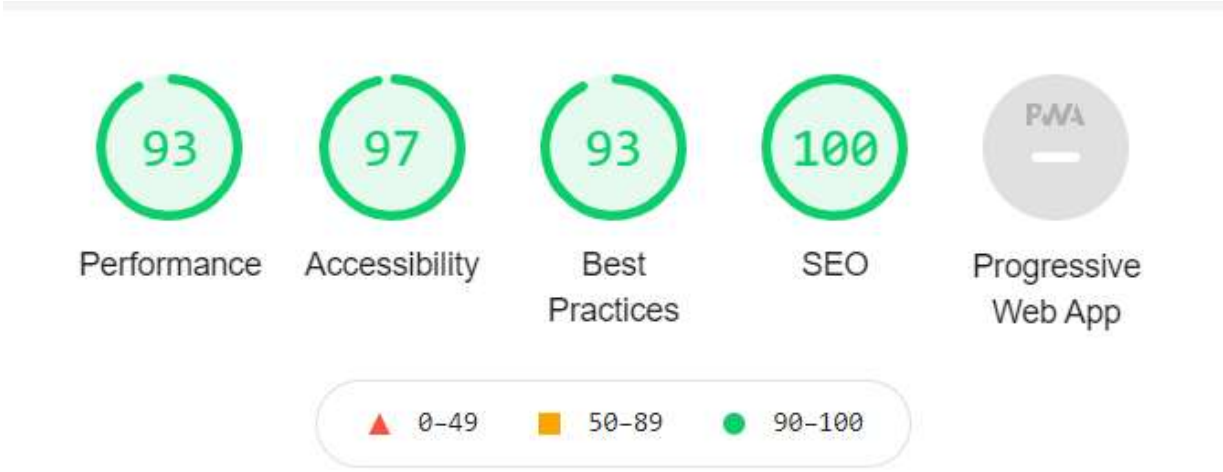


Рисунок 3.10 - Результати тестування у «Lighthouse»

ВИСНОВКИ

У третьому розділі було розглянуто результати проектування і розробки системи. Було виділено особливості архітектурних рішень такі як використання патерну фасад для спрощення роботи з підсистемою створення місць для запису на прийом і універсальний компонент вводу для клієнтської частини. Було сформовано інструкцію користувача, в інструкції розглянуті основні процеси:

- Реєстрація користувача
- Авторизація
- Пошук лікаря
- Запис на прийом

За допомогою сучасних інструментів була протестована доступність і швидкість роботи системи, тести показали що системи відповідає всім сучасним стандартам.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи була розроблена медична інформаційна система. В процесі виконання роботи було досягнуто:

1. Були досліджені теоретичні основи і бізнес-процеси медичних інформаційних систем. Вивчена на достатньому рівні сфера дослідження предмету кваліфікаційної роботи, а саме процес роботи медичних закладів первинної ланки, сучасні вимоги Національної служби здоров'я України до ведення прийомів.

2. Досліджені всі основні принципи, необхідні для побудови сучасної інформаційної системи, порівнянні два види баз даних і обрано оптимальний для виконання поставленого завдання. Переглянуті існуючі рішення в даній сфері і сформульовано технічні вимоги.

3. Спроектовано та реалізовано медичну інформаційну систему з високим показником надійності, інтерфейсом зрозумілим для всіх вікових категорій користувачів. Також при розробці системи було зроблено упор на швидкодію і продуктивність. Розроблений веб-застосунок має клієнт-серверну архітектуру. У якості серверу виступає REST API написане на фреймворці ASP.NET Core 5.0, який використовує мову C# 8.0 і реляційна база даних MySQL. Для створення клієнтської частини було використано мову JavaScript та бібліотеки ReactJS і Redux.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Панасюк О.І., Плескач В.Л Концептуальні підходи до побудови медичної інформаційної системи для поліклініки. *Сучасні електромеханічні та інформаційні системи*: монографія / під заг. ред. І.В. Панасюка. Київ : КНУТД, 2021. С. 29-62.
2. Данилевский Ю.Г., Петухов И.А., Шибанов В.С. Информационная технология в промышленности: СПб.: Машиностроение. Ленингр. отделение, 1988.
3. Медична інформаційна система *Вікіпедія*: офіц. веб-сайт. URL: https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B4%D0%B8%D1%87%D0%BD%D0%B0_%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0 (дата звернення: 25.03.2021).
4. Microsoft Docs *Microsoft Docs*: веб-сайт. URL: https://docs.microsoft.com/en-us/aspnet/core/security/authorization/policies?view=aspnetcore-5.0 (дата звернення: 26.03.2021)
5. Joseph Albahari, Ben Albahari «LINQ Pocket Reference: Learn and Implement LINQ for .NET Applications»: O'Reilly Media, 2008 174 p.
6. React 2020 *React*: веб-сайт. URL: https://uk.reactjs.org/ (дата звернення: 26.03.2021);
7. MongoDB 2020 *MongoDB*: веб-сайт. URL: https://www.mongodb.com/ (дата звернення: 26.03.2021);
8. SQL против NoSQL на примере MySQL и MongoDB. *TProger*: веб-сайт. URL: https://tproger.ru/translations/sql-vs-nosql/ (дата звернення: 26.03.2021);
9. James Chambers, David Paquette, Simon Timms ASP.NET Core Application Development: Bulding an Application in Four Sprints (Developer Reference): Microsoft Press, 2017. P. 25 - 277;

10. Mark J. Price C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development: Packt Publishing, 2019. 818 p.
11. Карпов, О.Э. Автоматизация процессов, цифровые и информационные технологии в управлении и клинической практике лечебного учреждения: научные труды. Москва: Деловой экспресс, 2016. 388 с.
12. Информационная технология, экономика, культура: сб. обзоров і рефератов: Москва: ИНИОН РАН, 1995.
13. Кобринский, Б.А. Автоматизированные регистры медицинского назначения: теория и практика применения: Москва: Менеджер здравоохранения, 2011. 148 с.
14. Берг А.И., Черняк Ю.Я. Информация и управление.: Москва, 1996.
15. Информационные технологии управления: Учебное пособие для вузов: Москва, 2005.
16. Юсупов Р.М., Заболоцкий В.П. Научные и методологические основы информатизации: СПб., 2005.
17. Мезенцев К.Н. Автоматизированные информационные системы: Учебник для студентов учреждений среднего профессионального образования: Москва: ИЦ Академия, 2013. 176 с.
18. Ясенев В. Н. Автоматизированные информационные системы в экономике: Учебно-методическое пособие: Москва: Юнити-Дана, 2007. 597с.
19. Варфоломеева А.О., Коряковский А.В. Информационные системы предприятия: Учебное пособие: Москва: НИЦ ИНФРА-М, 2013. 283с
20. Пирогов В.Ю. Информационные системы и базы данных: организация и проектирование: Учебное пособие: СПб.: БХВ-Петербург, 2009. 528с.
21. Helsi.me *Helsi*: веб-сайт. URL: https://helsi.me/ (дата звернення: 15.04.2021);
22. Патерни проектування – фасад *RefactoringGuru*: веб-сайт. URL: https://refactoring.guru/uk/design-patterns/facade (дата звернення: 15.04.2021);

ДОДАТКИ

ДОДАТОК А

Універсальний компонент для створення полів вводу

```
import React from "react";
import DatePicker from "react-datepicker";
import Select from "react-select";
import AsyncSelect from "react-select/async";
import "./_input.scss";
import "react-datepicker/dist/react-datepicker.css";
const colourStyles = {
  control: (styles) => ({ ...styles, backgroundColor: "white" }),
  option: (styles, { data, isDisabled, isFocused, isSelected }) => {
    return {
      ...styles,
      backgroundColor: isDisabled
        ? null
        : isSelected
        ? "#31b33c"
        : isFocused
        ? "#5ed668"
        : null,
      color: isDisabled ? "#ccc" : isSelected || isFocused ? "#fff" : "#333",
      cursor: isDisabled ? "not-allowed" : "default",

      ":active": {
        ...styles[":active"],
        backgroundColor: !isDisabled && "#31b33c",
      },
    };
  },
  input: (styles) => ({ ...styles }),
  placeholder: (styles) => ({ ...styles }),
  singleValue: (styles, { data }) => ({ ...styles }),
};
const Input = (props) => {
  let inputElement = null;
  const inputClasses = ["inputElement"];
  console.log(props.elementType + " " + props.value);
  if (props.className) inputClasses.push(props.className);
  if (props.invalid && props.shouldValidate && props.touched) {
    inputClasses.push("invalid");
  }

  switch (props.elementType) {
```

```
case "input":
  inputElement = (
    <input
      className={inputClasses.join(" ")}
      {...props.elementConfig}
      value={props.value}
      onChange={props.changed}
    />
  );
  break;
case "textarea":
  inputElement = (
    <textarea
      className={inputClasses.join(" ")}
      {...props.elementConfig}
      value={props.value}
      onChange={props.changed}
    />
  );
  break;
case "select":
  inputElement = (
    <Select
      options={props.elementConfig.options}
      onChange={props.changed}
      value={props.value}
      styles={colourStyles}
      placeholder={props.elementConfig.placeholder}
    />
  );
  break;
case "asyncSelect":
  inputElement = (
    <AsyncSelect
      value={props.value}
      placeholder={props.elementConfig.placeholder}
      cacheOptions
      loadOptions={props.loadOptions}
      defaultOptions
      onInputChange={props.changed}
      styles={colourStyles}
    />
  );
  break;
case "date":
  inputElement = (
    <DatePicker
      onChange={props.changed}
      selected={Date.parse(props.value)}
```

```
            className={inputClasses.join(" ")}
            dateFormat="yyyy-MM-dd"
            placeholderText={props.elementConfig.placeholder}
          />
        );
        break;
      default:
        inputElement = (
          <input
            className={inputClasses.join(" ")}
            {...props.elementConfig}
            value={props.value}
            onChange={props.changed}
          />
        );
    }

    return (
      <div
        className={
          props.containerClassName ? "input " + props.containerClassName : "input"
        }
      >
        {props.label && <label className="label">{props.label}</label>}
        {inputElement}
      </div>
    );
};

export default Input;
```