Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

**БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА**
на тему:
**Медичний інтернет-сервіс для обслуговування та реєстрації пацієнтів**

ПОЯСНЮВАЛЬНА ЗАПИСКА
08-23.БДР.039.00.000 ПЗ

Виконав студент 2 курсу, групи 1КІ-20мс
спеціальності123 — Комп'ютерна інженерія
Ткачук В.М.

Керівник роботи к.т.н., доц. каф. ОТ
Колесник І.С.
"13" 06 2022 р.

Рецензент д.т.н., проф., завідувач кафедри ЗІ
Лужецький В.А.
"14" 06 2022 р.

**Допущено до захисту**
д.т.н., проф. Азаров О.Д.

"15" 06 2022 р.

ВНТУ 2022

**ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ**

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітній рівень — бакалавр
Спеціальність — 123 Комп'ютерна інженерія

**ЗАТВЕРДЖУЮ**

Завідувач кафедри обчислювальної техніки

проф., д.т.н. О.Д. Азаров

"08" 02 2022 р.

**З А В Д А Н Н Я**

**НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ**

студенту **Ткачуку Валерію Михайловичу**

1 Тема роботи «Медичний інтернет–сервіс для обслуговування та реєстрації пацієнтів» керівник роботи Колесник Ірина Сергіївна к.т.н., доцент, затверджено наказом вищого навчального закладу від 24.03.2022 року № 66

2 Строк подання студентом роботи 13.06.2022.

3 Зміст пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз застосування сучасних інтернет–сервісів для обслуговування та реєстрації пацієнтів, вибір засобів для розробки інтернет–сервісу, розробка медичного інтернет–сервісу, висновки, перелік джерел посилання.

4 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): технічне завдання, файл обробки запитів авторизації, файл обробки запитів медичної організації, файл встановлених пакетів, головний файл backend, HTML–розмітка головної сторінки сайту, сервіс для повідомлень.

5 Дата видачі завдання 10.02.2022.

6 Календарний план виконання БДР приведений в таблиці 1.

Таблиця 1 — Календарний план

| № з/п | Назва етапів виконання бакалаврської дипломної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Постановка задачі роботи | 11.02.22 | вик. |
| 2 | Аналіз сучасних інтернет–сервісів для обслуговування та реєстрації пацієнтів | 12.02–28.02.22 | вик |
| 3 | Вибір засобів для розробки медичного інтернет–сервісу для обслуговування та реєстрації пацієнтів | 01.03–12.03.22 | вик |
| 4 | Проектування медичного інтернет–сервісу | 13.03–17.04.22 | вик |
| 5 | Підготовка матеріалів та опис розробки | 20.04–01.05.22 | вик |
| 6 | Оформлення пояснювальної записки та ілюстративного матеріалу | 03.05–30.05.22 | вик |
| 7 | Аналіз виконання роботи, висновки, додатки | 01.06–05.06.22 | вик |
| 8 | Перевірка якості виконання бакалаврського проекту та усунення недоліків | 25.05.22 | вик |

Студент Ткачук В.М

Керівник к.т.н., доц. Колесник І.С.

# АНОТАЦІЯ

Дана бакалаврська дипломна робота присвячена проектуванню медичного інтернет-сервісу для обслуговування та реєстрації клієнтів.

В даній роботі виконується аналіз сучасних медичних інтернет-сервісів, розглянуто аналоги та можливості їх застосування. Виконано аналіз засобів розробки клієнтської та серверної частини медичного інтернет-сервісу. Для налаштування клієнтської частини був використаний JavaScript фреймворки Angular, NgRx, бібліотеки як: RxJS, Angular material, Bootstrap. З серверної сторони використано node.js фреймворк Express.js. В якості бази даних було використано MongoDB.

Ключові слова: веб-розробка, веб-додаток, медичний сервіс, JavaScript, Node,js,Express.js, Angular12,MongoDB,RxJS.

## ABSTRACT

This bachelor's degree project is dedicated to the design of a medical Internet service for customer service and registration.

The project analyzes modern medical Internet services, considers analogues and applications. The analysis of means of development of client and server part of medical Internet service is executed. To configure the client part, JavaScript frameworks Angular, NgRx, libraries such as: RxJS, Angular material, Bootstrap were used. On the server side, the node.js Express.js framework is used. MongoDB was used as the database.



Keywords: web development, web application, medical service, JavaScript, Node, js, Express.js, Angular12, MongoDB, RxJS.

ЗМІСТ

# ВСТУП

В наш час тема здоров’я є досить важливо, адже вчасне виявлення проблеми зберігає людині життя. З розвитком комп’ютерних систем сфера медицини отримала потужний поштовх в наданні послуг. Інтернет став дуже зручним, доступним та привабливим рекламним майданчиком. Сучасні мережеві інформаційні технології дозволяють нам використовувати досить спектр різних видів інструментів, що дозволяють кожному рекламодавцю, нагадувати про свою особу певній великій аудиторії людей, так і можливість доносити свої власні рекламні сповіщення безпосередньо до цільової аудиторії, що в свою чергу дуже сильно полегшує продаж певної продукції та послуг [1]. Сучасні технології дозволяють отримати кожному якісне лікування та прекрасний сервіс.

Інтернет-сервіс дає можливість досить вигідно представити свою компанію, продукт чи спільноту професіоналів для широкого загала людей, щоб існувала можливість дізнатись про пропозиції товарів та послуг. Завдячуючи власному веб-ресурсу ви також матимете змогу збирати необхідну та дуже цінну інформацію про ваших потенційних клієнтів, можете організувати для них розсилання повідомлень, проводити різні рекламні кампанії та власні онлайн (вебінари, ефіри) та оффлайн (семінари та конференції) події тощо.

Запровадження інтернет-сервісу дозволяє швидко змінювати інформацію на сайті в будь-який час і з будь-якої точки світу, зо забезпечує поліпшення процесу замовлення продукції, логістики і тому подібне. Продумана автоматизація дозволяє заощадити величезні суми грошей і оптимізувати процеси взаємодії як всередині компанії так і безпосередньо з клієнтами. Створення повноцінного спільноти навколо веб-проекту з використанням інтернет-сервісу зміцнює позиції і допомагає виділитися серед конкурентів.

Найперспективніший на сьогоднішній день метод просування медичних послуг — просування через Інтернет. Саме тому в будь-якій клініці, у кожного приватного медичного кабінету і навіть в будь-якого лікаря рано чи пізно з'являється така нагальна потреба, як створення комерційного сайту. Кожна приватна клініка створює власний інтернет-сервіс для своїх клієнтів, на якому надає їм повний спектр інформації про певні хвороби, їх наслідки, лікування [2].

Інтернет сервіс VinMedical містить інформацію про лікарні міста Вінниця, в який ви можете записатися на прийом до лікаря. На ньому є можливість обрати фах лікаря який вам потрібен, безпосередньо вибрати лікаря та час прийому. За допомогою даного інтернет-сервісу кожен може без жодних проблем звернутися до лікаря та отримати профільну медичну допомогу найвищої якості.

Даний сервіс спрощує життя кожній людині, адже не в кожного є велика кількість часу та фізична можливість постійно відвідувати лікарню.

**Метою роботи** є розробка медичного інтернет-сервісу для обслуговування та реєстрації пацієнтів, який допоможе забезпечувати надання медичної допомоги пацієнту в режимі онлайн.

**Задачі дослідження** бакалаврської роботи:

— проаналізувати існуючі медичні інтернет-сервіси для обслуговування та реєстрації пацієнтів;

— проаналізувати аналоги сучасних технологій веб-програмування та їх відмінності;

— створити дизайн інтернет-сервісу;

— виконати верстку та наповнити контент сайту.

**Об'єкт дослідження** — процес створення медичного інтернет-сервісу для надання медичних послуг онлайн.

**Предмет дослідження** — методи та засоби для розробки багатокористувацького інтернет-сервісу.

**Методами дослідження** бакалаврської роботи є використання JavaScript технологій для реалізації поставленої задачі.

**Практичне значення одержаних результатів** — створено медичного інтернет- сервісу для обслуговування та реєстрації пацієнтів.

Апробація здійснена на науково-технічній коференції:

Ткачук В.М. Медичний інтернет-сервіс для обслуговування та реєстрації пацієнтів//Тези доповідей НТКП. Факультет інформаційних технологій та комп’ютерної інженерії (2022). Україна, Вінниця, 2022 р.: матеріали конференцій.

# 1 ОСНОВИ ДОСЛІДЖЕНЯ МЕДИЧНИХ ІНТЕРНЕТ-СЕРВІСІВ ДЛЯ ОБСЛУГОВУВАННЯ ТА РЕЄСТРАЦІЇ ПАЦІЄНТІВ

Сучасний світ вимагає конкурентоспроможних рішень у галузі інтернет тенденцій для ефективної реклами та просування свого товару, продукції, компанії. Повна діяльність організації повинна нести за собою головну та незмінну мету задоволення потреб користувачів, тому ще це — найкращий шлях для досягнення власних цілей росту та максимального підвищення рентабельності. Медичний інтернет-сервіс окрім підвищення рентабельності підприємства допомагає пацієнтам швидше отримувати консультацію лікаря.

## 1.1 Основні поняття та суть медичних інтернет-сервісів

Інформаційні технології з часом все щільніше проникають в кожну сферу нашого сьогоднішнього життя. Вони доторкнулися й медицини: в багатьох медичних закладах вже успішно застосовуються різні електронні системи для управління роботою, а також автоматизований документообіг, все більша кількість медичних центрів мають на меті створити власний «віртуальний офіс» в мережі Інтернет — веб-сайт [1].

Медичні установи, що мають власний сайт мають досить широкий ряд переваг. Зараз власний веб-сервіс є невід’ємною частиною сучасного типу ведення бізнесу: кожна поважаюча себе компанія просто зобов’язана мати свій власний сайт, інакше вона втрачає свій імідж та авторитет в очах потенційних клієнтів навпроти інших конкурентів, які вже мають віртуальні представництва в мережі. Адже саме в Інтернеті знаходиться значна частина можливих клієнтів, які коли-небудь чули чи звертали увагу про ваш медичний заклад, захочуть отримати якомога більше інформації про нього [2]. Інакше вони просто звернуться до ваших конкурентів, координати яких вони зможуть знайти без всяких проблем.

Багато сайтів містять форуми, де відвідувачі не тільки знайомляться з публікаціями, а ще й обговорюють різні питання. На веб-сайті є можливість

розмістити необмежену кількість файлів, які в свою чергу можуть зацікавити відвідувача або з якими йому необхідно буде ознайомитися перед зверненням до певної медичної установи (клініка або медичний центр). На ньому він самостійно матиме змогу знайти потрібну інформацію, не відволікаючи персонал установі різними зайвими телефонними дзвінками, що значно збереже час та сили ваших співробітників, розвантажить телефонні лінії, та знизить інформаційні витрати [3]. Жоден із інформаційних ресурсів не може забезпечити такої великої кількості можливостей дуже доступно та відносно дешево рекламувати свою компанію. Окрім інформаційної підтримки, на веб-сайті кожен відвідувач може скористатися і певними онлайн послугами такими як: дізнатися про склад лікарського персоналу, отримати відповідь на найпоширеніші питання, записатися на консультацію до лікаря тощо.

Проста можливість кожному записатись на прийом до певного лікаря через інтерактивну форму на сайті економить багато часу кожному клієнту та полегшує організацію роботи обліку клініки персоналу. Форма запису на прийом дуже гнучка і може мати інші варіанти, наприклад, відвідувач може записатися на прийом до конкретного лікаря зі списку, дізнатися його графік прийому, кількість днів або годин, записаних безкоштовно, тощо.

Часто ми бачимо, що підписка організована для клієнта саме для того, щоб кожен міг надсилати останні новини про дії клініки. Наявність сайту, на якому можна аналізувати статистику його відвідувань, надає багато корисної та дуже цікавої інформації. Статистика допомагає визначити та розробити способи залучення клієнтів, вона також дає можливість дізнатися, що їх цікавить та інше.

Ми розглянули найбільш основні переваги медичних центрів та клінік, які мають свої ресурси в мережі Інтернет. Проте застосування інформаційних технологій в їх діяльності цим не обмежується. Можливості сайтів досить великі, в свою чергу це дозволяє реалізувати будь-яку опцію чи певну послугу, яка може знадобитися відвідувачу нашого сайту або його керівництву.

## 1.2 Аналітичний огляд сучасних інтернет-сервісів для обслуговування та реєстрації пацієнтів

Перш ніж розглядати аналоги медичних інтернет — сервісів визначимо спочатку важливий термін «телемедицина». Телемедицина — використання різних високотехнічних комп'ютерних та телекомунікаційних технологій для процесу обміну медичною інформацією.

Пандемія дала серйозний поштовх для розвитку нових способів діагностики та лікування. Ще не так давно онлайн медицина не користувалася попитом, але за останні два роки популярність таких сервісів значно зросла. Карантинний режим, викликаний пандемією коронавірусу, показав, наскільки корисними можуть бути дистанційні послуги. Онлайн-прийом підвищує рівень безпеки як пацієнтів, так і лікарів, тому що ймовірність інфікування без виходу з дому суттєво знижується [4]. Простими словами телемедицина передбачає консультації з лікарями за допомогою засобів телекомунікації: телефонів, мобільного зв'язку, інтернету без візиту до медичної установи.

Зокрема, це дистанційні консультації пацієнтів із лікарями та лікарів із лікарями. Вперше віддалена медична консультація по відеозв'язку у форматі "лікар — лікар" пройшла ще 1965 року під час операції із заміни клапана на штучному серці. За кілька років — на початку 70-х — з'явилося саме визначення «телемедицина».

Телемедицина дуже швидко трансформує галузь охорони здоров'я, і в перспективі в найближчі кілька років вона, як очікується, може стати найбільш прийнятним і використовуваним методом для діагностики і призначення різних лікарських препаратів. За деякими оцінками, до 2026 року обсяг ринку телемедицини може зрости більше, ніж на 19% та перевищити 175 млрд дол. США, дані згідно з дослідницьким звітом від Global Market Insights, Inc. Розглянемо основні сервіси з телемедицини в Україні.

### 1.2.1 Поліклініка без черг

Це медична інформаційна система, яка була впроваджена в рамках медичної реформи. Спочатку система працювала як онлайн-сервіс для запису до лікарів, а з початком пандемії запустила нову функцію онлайн- консультацій. На сьогодні до платформи приєдналися понад 17 000 лікарів та 2,9 млн пацієнтів [5].

Щоб потрапити до потрібного лікаря, пацієнту не потрібно встановлювати спеціальні додатки — достатньо створити на сайті свій кабінет та вибрати зручний час. Ціна: консультації безоплатні.

Переваги та можливості «Поліклініки без черг» для кожного пацієнта:

— швидкий електронний запис до лікаря з будь-якого пристрою;

— безкоштовно;

— можливість реєстрації ваших рідних на один номер телефону;

— повне збереження історії ваших відвідувань у електронному кабінеті пацієнта;

— просте скасування запису, при необхідності. Лікар без проблем зможе прийняти іншого пацієнта;

— отримання електронного талону на прийом до лікаря у вигляді SMS інформування чи повідомлення на email.

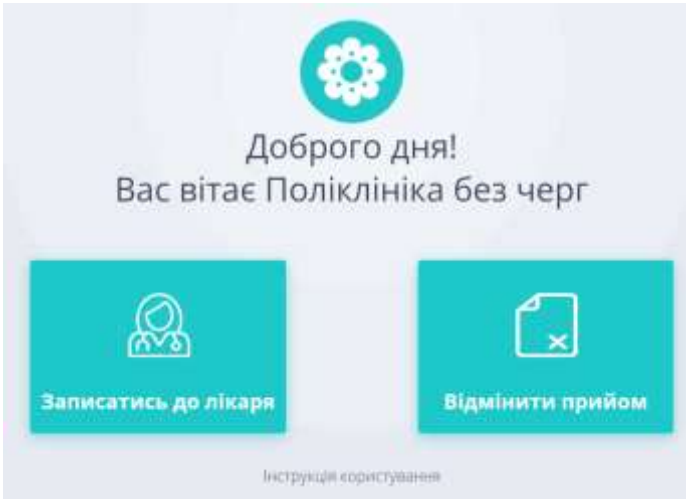



Рисунок 1.1 — Інформаційна система «Поліклініка без черг»

### 1.2.2 Платформа Telemed24

Telemed24 — сервіс медичної інформаційної системи Медстар, призначений для проведення консультацій та реєстрації результатів дистанційних прийомів. Використовувати цю систему безпечно, оскільки персональні дані захищені згідно з вимогами законодавства.

Для користування сервісом необхідно встановити мобільний додаток MedCard24 [6]. Через мобільний додаток можна записатися на онлайн-прийом, самостійно обравши час, день, лікаря та вказавши причину звернення.

За 15 хвилин до початку онлайн-консультації на смартфон приходить повідомлення з нагадуванням. Консультація відбувається наживо, у форматі відеодзвінка.

Ціна: телеконсультації через дану платформу є безоплатні для пацієнтів і лікарів. Лікарі ж, зокрема, є в першу чергу зацікавленими у проведенні онлайн-прийоми через ресурс Telemed24, тому що вони мають змогу отримати певну компенсацію за кожну таку консультації, як за звичайний лікарський прийом.

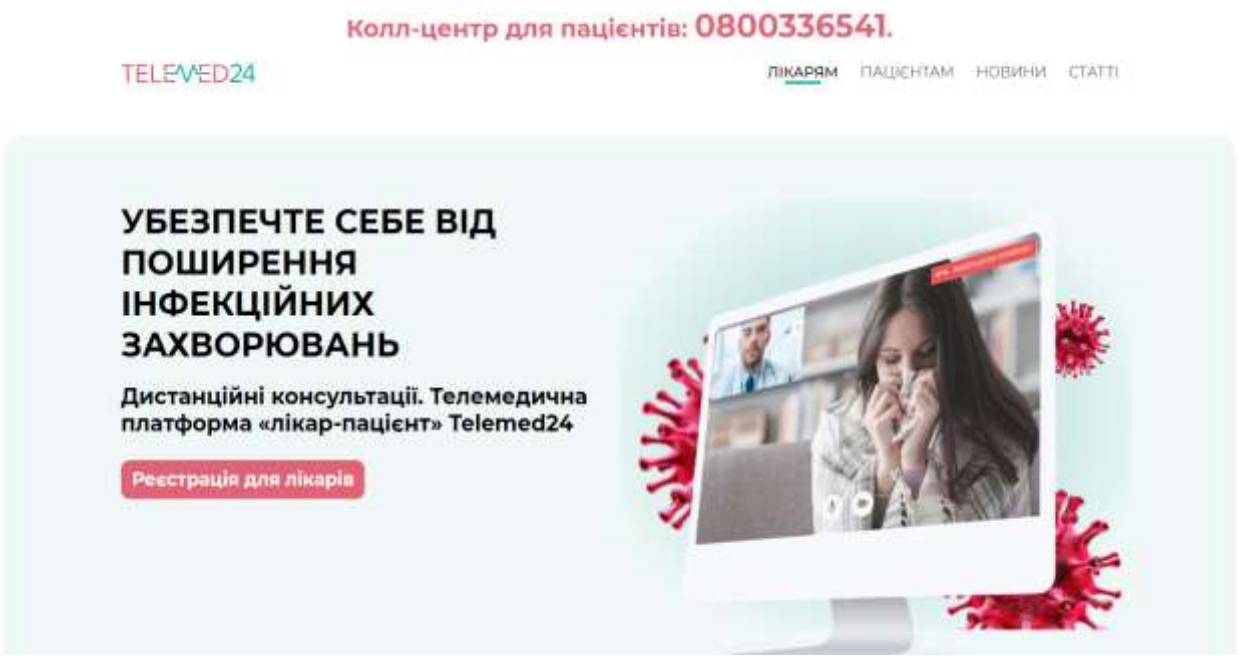



Рисунок 1.2 — Веб-сервіс «Telemed24»

### 1.2.3 Doctor Online від Київстар

У період карантину мобільний оператор «Київстар» разом із партнером «Онлайн Доктор» пропонує всім абонентам усіх операторів зв’язку безкоштовні консультації лікаря в додатку «Онлайн Доктор».

Кожен користувач, незалежно від того, де він перебуває, має унікальну можливість отримати безкоштовну цілодобову консультацію з понад 40 кваліфікованими лікарями та можливість отримати конкретні поради щодо лікування через чат, відео-чи аудіодзвінок.

Додаток «Доктор Онлайн» — працює у сфері телемедицини, розроблений однойменною компанією за підтримки «Київстар» [7]. У ньому кожен користувач має можливість звернутися до обраного фахівця за консультацією через чат, відео-чи аудіодзвінок, також можна отримати направлення на аналіз нашої партнерської лабораторії «Синево» та замовити ліки з додатком Liki24 .com і можна запланувати в календарі розклад прийому необхідних ліків, і з часом програма надішле вам нагадування про те, що вам потрібно отримати ліки.

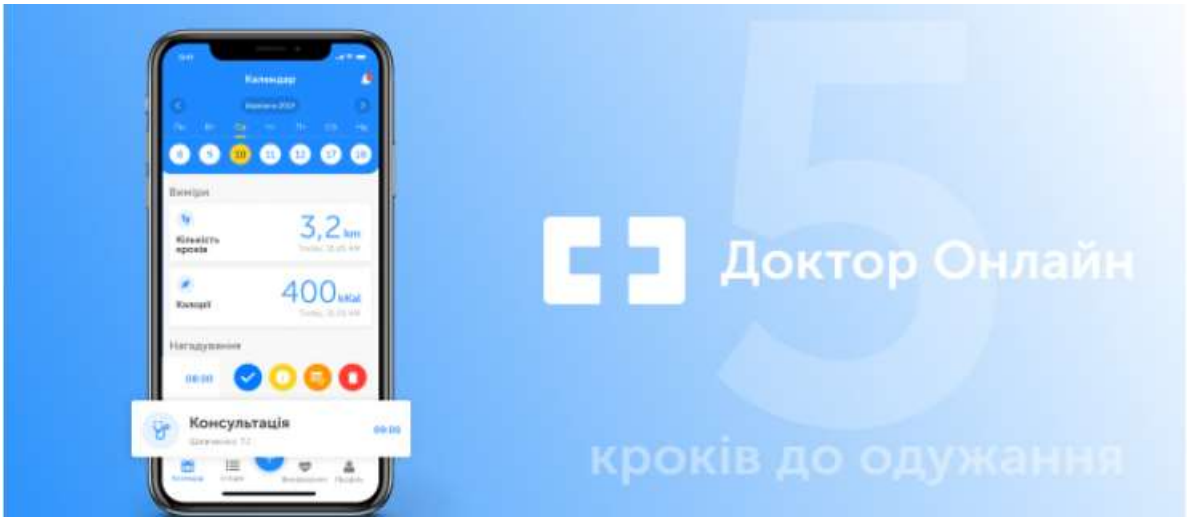
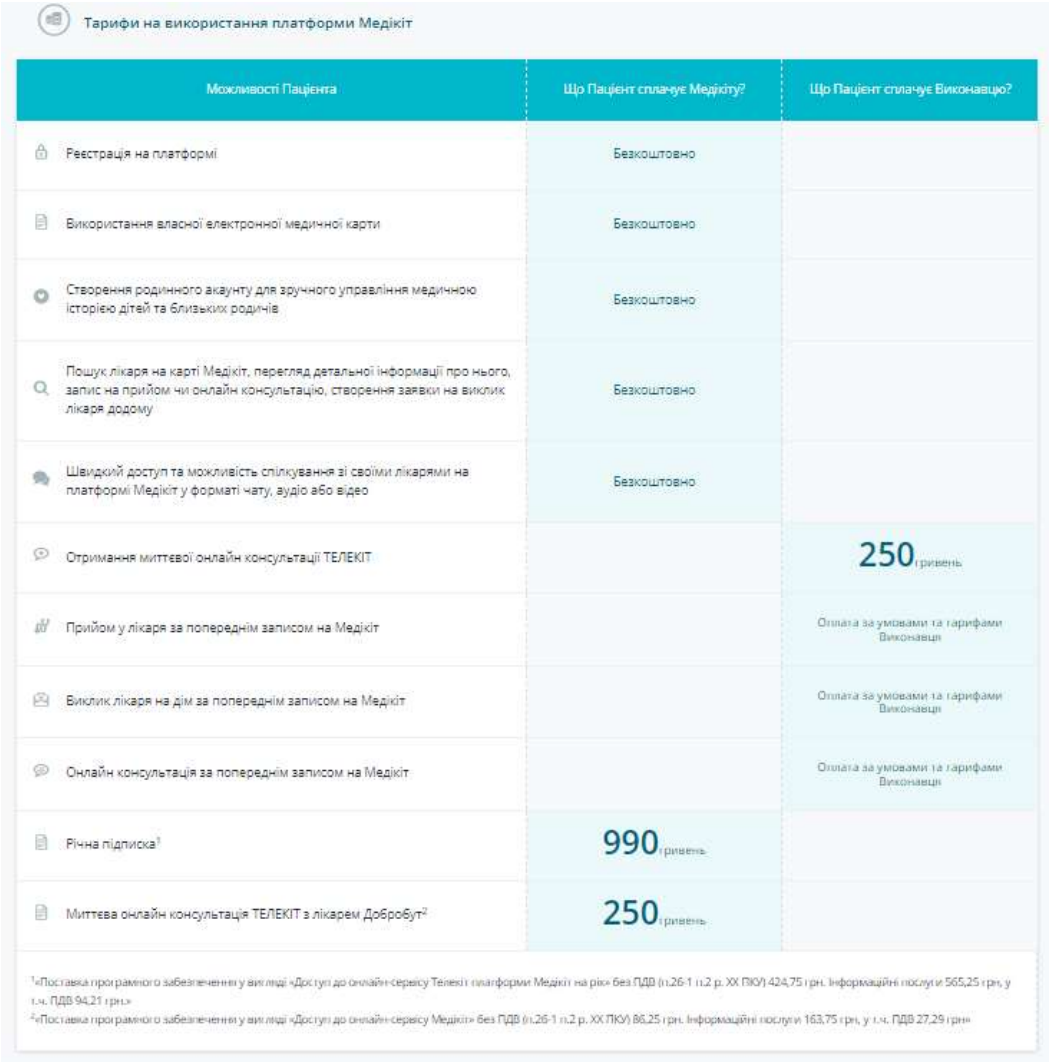



Рисунок 1.3 — Веб-сервіс «Доктор Онлайн»

### 1.2.4 Medikit

Medikit — це медична комунікаційна платформа, що надає свої послуги у форматі онлайн-консультацій «лікар-пацієнт» через онлайн-чат, відео або аудіозв'язок . З допомогою платформи кожен користувач може поставити власне питання щодо здоров'я та отримувати якісні консультації від різних лікарів. До платформи залучені понад 250 лікарів за 70 різними спеціалізаціями [8]. Після реєстрації, кожен лікар може консультувати пацієнтів та отримувати в результаті безготівкову оплату за надані ним послуги.

Актуальні ціни є зазначеними на рисунку 1.2.4.



Тарифи на використання платформи Медікіт

| Можливості Пацієнта | Що Пацієнт сплачує Медікіту? | Що Пацієнт сплачує Виконавцю? |
|---|---|---|
| Реєстрація на платформі | Безкоштовно | |
| Використання власної електронної медичної карти | Безкоштовно | |
| Створення родинного акаунту для зручного управління медичною історією дітей та близьких родичів | Безкоштовно | |
| Пошук лікаря на карті Медікіт, перегляд детальної інформації про нього, запис на прийом чи онлайн консультацію, створення заявки на виклик лікаря додому | Безкоштовно | |
| Швидкий доступ та можливість спілкування зі своїми лікарями на платформі Медікіт у форматі чату, аудіо або відео | Безкоштовно | |
| Отримання миттєвої онлайн консультації ТЕЛЕКІТ | | 250 гривень |
| Прийом у лікаря за попереднім записом на Медікіт | | Оплата за умовами та тарифами Виконавця |
| Виклик лікаря на дім за попереднім записом на Медікіт | | Оплата за умовами та тарифами Виконавця |
| Онлайн консультація за попереднім записом на Медікіт | | Оплата за умовами та тарифами Виконавця |
| Річна підписка[1] | 990 гривень | |
| Миттєва онлайн консультація ТЕЛЕКІТ з лікарем Добробут[2] | 250 гривень | |

[1]«Поставка програмного забезпечення у вигляді «Доступ до онлайн-сервісу Телекіт платформи Медікіт на рік» без ПДВ (п.26-1 п.2 р. XX ПКУ) 424,75 грн. Інформаційні послуги 565,25 грн, у т.ч. ПДВ 94,21 грн.»

[2]«Поставка програмного забезпечення у вигляді «Доступ до онлайн-сервісу Медікіт» без ПДВ (п.26-1 п.2 р. XX ПКУ) 86,25 грн. Інформаційні послуги 163,75 грн, у т.ч. ПДВ 27,29 грн»

Рисунок 1.4 — Веб-сервіс «Медікіт»

### 1.2.5 Likar Online

Likar Online — компанія, що спеціалізується виключно на телемедицині та віддалених консультаціях, яка працювала в цьому напрямку ще до пандемії. Це сервіс онлайн консультацій без здійснення візиту до медичного закладу. Likar online — одна з перших організацій, що запровадила послуги телемедицини на українському ринку.

Партнер сервісу Likar online — медичний центр Consilium Medical. Це сучасний багатопрофільний лікувально - профілактичний заклад, розташований в Києві, що займається вирішенням клінічних ситуацій різного рівня складності, включаючи найважчі випадки [9].

Кожен спеціаліст, що співпрацює із Likar online, є практикуючим лікарем та пройшов спеціальне навчання технологіям ведення телемедичних консультацій.

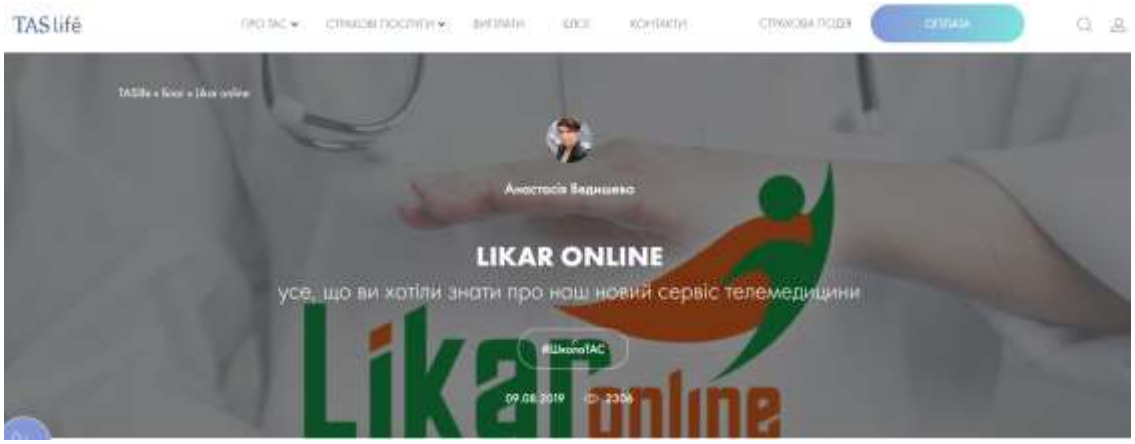



Рисунок 1.5 — Веб-сервіс «Likar Online»

Отже, медичний інтернет-сервіс — це унікальний інтерфейс між певним підприємством та його оточенням — партнерами, постачальниками та пацієнтами. Тому наразі створення сайта є одним із найбільш головних завдань підприємницької діяльності, насамперед і в мережі Internet.

Таким чином, телемедицина в Україні набирає обертів і, за прогнозами спеціалістів, найближчі роки набуде широкої популярності у населення.

# 2 ВИБІР ТА АНАЛІЗ ЗАСОБІВ ДЛЯ РОЗРОБКИ МЕДИЧНОГО ІНТЕРНЕТ-СЕРВІСУ ДЛЯ ОБСЛУГОВУВАННЯ ТА РЕЄСТРАЦІЇ ПАЦІЄНТІВ

Дана бакалаврська дипломна робота — це інтернет-сервіс VinMedical, який складається з клієнтської та серверної частини. Для налаштування клієнтської частини був використаний JavaScript фреймворки Angular, NgRx, бібліотеки як: RxJS, Angular material, Bootstrap. З серверної сторони використано node.js фреймворк Express.js. В якості бази даних було використано MongoDB.

## 2.1 Клієнтська частина

Frontend — є публічною частиною web-додатків (веб-сайтів), з якою кожен користувач може взаємодіяти і контактувати безпосередньо напряму. Frontend включає відображення певних функціональних завдань призначених для користувача інтерфейсу, які виконуються на клієнтській стороні, включно з обробкою запитів користувачів. Іншими словами, фронтенд — це все те, що бачить кожен користувач при відкритті web-сторінки. Його логіка розподілена між сервером і клієнтом, зберігання різних даних здійснюється переважно на сервері, а обмін інформацією відбувається у мережі. Інакше кажучи, це те, що бачить користувач і які дії виконує кожен раз, коли підключається до мережі інтернет та відкриває певний браузер.

Frontend розробка — це робота націлена на створення публічної частини web-додатку, з якою безпосередньо контактує користувач, та функціоналу, який майже завжди виконується на стороні клієнта. Тобто, фронтенд розробник працює для того, щоб кожна кнопочка на сайті, зображення, текст і вікно не тільки стояли на певному своєму місці, не заважали та перекривали один одного і виглядали цілісно (це веб-верстка), але і для виконання свого прямого призначення — підштовхували до якоїсь певної дії (наприклад, щоб кнопка “придбати” відкривала кошик, а “play” — запускала відтворення певного фільму або музичної композиції).

### 2.1.1 Фреймворк Angular

Angular (часто відомий як Angular 2 або Angular 2+) — це інтерфейсний фреймворк TypeScript з відкритим кодом, розроблений командою Angular і заархівований Wayback Machine, Google і приватними розробниками та компаніями спільноти 18 серпня 2021 року. Angular — це AngularJS, перероблений і перероблений тією ж командою розробників [1].

Як платформа, Angular включає:

— компонентний фреймворк для створення масштабованих веб-додатків;

— колекція добре інтегрованих в себе різних бібліотек, що охоплюють широкий спектр функціоналу, включаючи маршрутизацію, керування різними формами, клієнт-серверний зв’язок тощо;

— набір інструментів для розробників, які допоможуть вам розробляти, створювати, тестувати та оновлювати код.

AngularJS обробить цей тег користувача і перетворить на повноцінний календар, як сказано у вихідному коді. Також була представлена важлива концепція ін'єкції залежностей, яка дозволяє пов'язувати компоненти програми, полегшуючи повторне використання та тестування коду. Ми не будемо надалі детально заглиблюватися, але в AngularJS ще багато всього. AngularJS став популярним дуже швидко і отримав велику підтримку. Тим не менш, розробники вирішили піти ще далі і перейшли до створення нової версії Angular 2 (пізніше Angular без частини JS) . Фреймворк отримав нове ім'я не просто так: він був повністю переписаний та перепроектований, а багато концепцій було переглянуто.

Angular з кожним роком набирає популярність. Станом на серпень 2020 року щодня відбувається близько 1,5 мільйона завантажень Angular/core, що подвоюється щороку.

### 2.1.2 Інтерфейс та дизайн сайту

Інтерфейс — це «міст» між користувачем та системою. За допомогою певного інтерфейсу користувач зможе пояснити системі, що йому потрібно від неї, а вона в свою чергу це виконає. Користувач вирішить піти із сайту. Так поведінка інтернет — користувачів за даними Online Marketing Institute:

— 85% можуть піти із сайту, уразі якщо їм не сподобається дизайн інтерфейсу;

— 83% покинуть сайт, якщо вони будуть змушені робити велику кількість кліків, щоб знайти потрібну інформацію;

— 40% користувачів ніколи не повернуться на сайт, уразі якщо їм було важко використовувати його вперше.

Дизайн веб-сайту часто можна трактувати як розробку інтерфейсу, який дозволяє ефективно взаємодіяти із контентом. Створюючи інтерфейс сайту, варто орієнтуватися на те, як будуть взаємодіяти з ним потенційні користувачі . Щоб він був зручним для відвідувачів, достатньо використати декілька встановлених правил.

— перш за все, не варто вигадувати колесо. Найчастіше складну систему самовираження дизайнера відвідувачі не розуміють і масово йдуть із сайту. Тобто складний, перевантажений інтерфейс безглуздий і нерентабельний;

— ще одне просте правило, яке працює у всіх сферах життя — не ускладнюйте. Визначтеся з головними функціями сайту і зробіть наголос на них. Безліч кнопок і посилань на екрані може збити відвідувача з пантелику;

— оберіть тип вашого сайту. На популярних сайтах має бути максимально ефективне управління: просте, швидке і зручне.

Принципи оптимального інтерфейсу однакові і для веб-сайтів, і для програм, і для сервісів. Тому інтерфейс повинен бути зручним і зрозумілим для користувача. Для швидкої та зручної розробки інтерфейсу використано бібліотеки Angular material та Bootstrap.

### 2.1.3 Бібліотека Bootstrap

Bootstrap — є відкритим та безкоштовним HTML, CSS та JS фреймворком, який використовується різними веб-розробниками для швидкого написання адаптивних дизайнів сайтів та веб-додатків [5].

NG Bootstrap містить готові компоненти для використання Angular. Він розширює Bootstrap 4 для використання з Angular. Основна перевага полягає в тому, що NG Bootstrap не має залежностей від сторонніх бібліотек Javascript [16], таких як jQuery або Bootstrap JS, яких слід уникати при розробці за допомогою Angular.

Бібліотека Bootstrap наразі використовується по всьому світу не тільки різними незалежними розробниками, а іноді цілими великими компаніями. На його базі створено велику кількість різних сайтів, їхні приклади переглянути можна на сторінці Bootstrap Expo. Його основна сфера застосування — це фронтенд розробка сайтів та інтерфейсів адмін. Серед інших аналогічних систем (Foundation, UIkit, Semantic UI, InK та ін) фреймворк Bootstrap є найпопулярнішим серед них.

Чому Bootstrap такий популярний? Це пов'язано з тим, що він дозволяє верстати сайти в кілька разів швидше, ніж на чистому CSS і JavaScript. А в нашому світі час — це дуже цінний ресурс. Ще один аспект — доступність. Вона зводиться до того, що надає можливість навіть початківцю веб-розробнику (без глибоких знань і достатньої практики) створювати досить якісні макети.

Фреймворк Bootstrap — це набір набір CSS та JavaScript файлів. Щоб його використовувати ці файли, необхідно просто підключити до сторінки. Після цього вам стануть доступні інструменти фреймворку: колонкова система (сітка Bootstrap), класи і компоненти [5].

Отже, Bootstrap — це не просто набір готових інструментів (HTML фрагментів, класів, компонентів та плагінів), а добре спроектована фронтенд бібліотека, яку досить просто можна налаштувати під себе за допомогою редагування Sass змінних та використання міксинів.

### 2.1.4 Бібліотека RxJS

Програмування — це завжди вирішення проблем та пошук інструментів, які підходять для пошуку відповідей на конкретні питання. У разі RxJS вирішуване завдання полягає у можливості обробляти асинхронні виклики за допомогою багатьох подій. На цьому варто зупинитись докладніше.

Реактивне програмування — це парадигма асинхронного програмування, що стосується потоків даних і поширення змін. RxJS (Reactive Extensions for JavaScript) — це бібліотека для реактивного програмування з використанням спостережуваних, що полегшує створення асинхронного коду або коду на основі зворотного виклику.

RxJS забезпечує реалізацію Observable типу, яка потрібна до тих пір, поки тип не стане частиною мови і поки браузери не підтримають його. Бібліотека також надає допоміжні функції для створення спостережуваних і роботи з ними [2]. Ці допоміжні функції можна використовувати для:

— перетворення існуючого коду для асинхронних операцій у спостережувані;

— ітерація значень у потоці;

— відображення значень до різних типів;

— фільтрація потоків;

— створення кількох потоків.

З одного боку, RxJS це потужний інструмент, який дозволяє перетворювати складні послідовності дій на лаконічний код, з яким легко працювати.З іншого боку, ця простота грунтується на безлічі мовних механізмів, які вивчення потребує часу. Однак, отримані знання варті витрачених зусиль, коли розумієш, що за допомогою одного рядка коду можна зробити щось подібне до реалізації механізму «перетягнути і опустити», що потребує трьох наборів подій.

### 2.1.5 Фреймворк NgRx

NgRx — це фреймворк для створення реактивних додатків у Angular. NgRx натхненний шаблоном Redux — об’єднання подій у вашій програмі та отримання стану за допомогою RxJS. На високому рівні NgRx зберігає єдиний стан і використовує дії для вираження змін стану. NgRx відмінно керує складними станами, що робить його ідеальним для додатків з великою кількістю взаємодій користувачів і кількома джерелами даних [3].

NgRx складається з 5ти основних компонентів — Store, Actions, Reducers, Selectors та Effects. NgRx використовує концепцію односпрямованого потоку даних Redux, де всі дані програми проходять один і той же життєвий цикл. Цей односпрямований потік даних робить стан програми більш передбачуваним і, таким чином, легшим для розуміння. Цей потік застосовується лише до рівня управління станом і його не слід плутати з односпрямованим потоком даних рівня презентації. Життєвий цикл управління станом у NgRx зображений на рисунку 2.1.

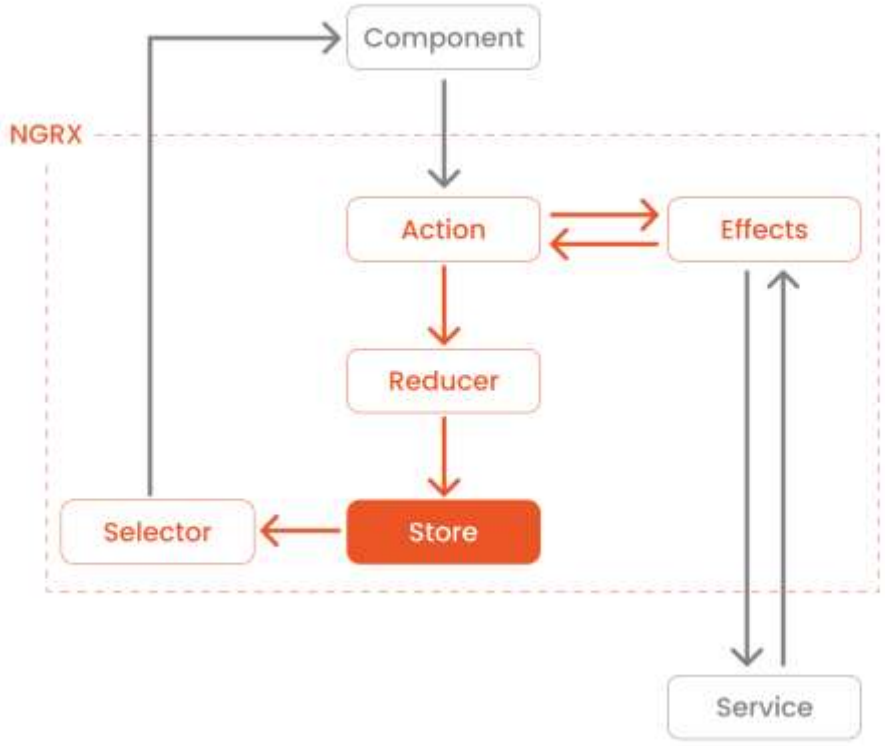



Рисунок 2.1 — Життєвий цикл управління станом у NgRx

## 2.2 Серверна частина

Back-end — програмна сторона веб-сторінки, невидима для користувача, яка відповідає за реалізацію відповідних сценаріїв, створення сторінок, власне з нею працюють розробники та адміністратори сайту, пов'язана з написанням скриптів для сервера. Бекенд-розробка відповідає за функціонал сайту і має справу з речами, яких ви не бачите, — такими як бази даних та сервери.

На відміну від frontend, backend це програмне забезпечення яке, як правило, розташоване на серверній стороні багатокомпонентної веб системи. Backend відповідає за дані проекту, забезпечує доступ до них, модифікацію, та видає їх на клієнтську сторону. Backend ще також називають «двигуном» веб-сайту.

Чим більше даних у проекті, а це тексти, зображення, відео, інформація профілів користувачі, таблиці, та різноманітні файли тим складнішим і потужнішим повинен бути backend. Разом з серверним програмним забезпеченням backend повинен видавати у короткі терміни підготовану інформацію для користувача на його запит. Найчастіше це генерація веб-сторінок та їх контенту. Наприклад, відвідувач на своєму телефоні клінкнув лінк, браузер передає запит на сторінку на сервер і у відповідь на цей запит backend веб-сайту генерує сторінку і надсилає її на телефон клієнта. У більшості випадків очікування дуже дратує відвідувачів тому «двигун» веб-сайту повинен працювати достатньо швидко та без затримок досилати потрібну інформацію.

Ситуація ускладнюється коли відвідувачів стає багато, і всі вони очікують на оперативне обслуговування своїх запитів. Загалом у багатьох проектах потрібно вибирати баланс між якістю представлення даних та швидкістю їх обслуговування.  Отже, серверна частина відіграє велику роль при створені веб-сервісу так як від backend залежить якість сайту та швидкість обслуговування користувачів.

### 2.2.1 Платформа Node.js

Одна з найпопулярніших платформ Node.js — платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Засновником платформи є Раян Дал (Ryan Dahl). Якщо раніше JavaScript застосовувався для обробки даних в браузері користувача, то node.js надав можливість виконувати JavaScript — скрипти на сервері та відправляти користувачеві результат їхнього виконання [6].

Node.js — це JavaScript-оточення побудоване на JavaScript-рушієві Chrome V8. Як асинхронне подієве JavaScript-оточення. Node.js створений під впливом таких систем як Event Machine в Ruby або Twisted в Python. Node.js використовує подієву модель значно ширше, він приймає цикл подій (event loop) за основу оточення, замість того, щоб використовувати його в якості бібліотеки. В інших системах завжди стається блокування виклику, щоб запустити цикл подій.

Зазвичай поведінка визначається через функції зворотнього виклику на початку скрипта і в кінці запускає сервер через блокуючий виклик, як от EventMachine::run(). В Node.js немає нічого подібного на виклик початку циклу подій. Node.js просто входить в подієвий цикл після запуску скрипта на виконання. Node.js виходить з подієвого циклу тоді, коли не залишається зареєстрованих функцій зворотнього виклику. Така поведінка схожа на поведінку браузерного JavaScript: подієвий цикл прихований від користувача.

HTTP є об'єктом першого роду в Node.js, розробленим з потоковістю та малою затримкою. Це робить Node.js хорошою основою для веб-бібліотеки або фреймворку. Те що Node.js спроектований без багатопоточності, не означає, що ви не можете використовувати можливості кількох ядер у вашому середовищі. Ви можете створювати дочірні процеси, якими легко керувати з допомогою API child_process.fork(). Модуль cluster побудований на цьому інтерфейсі і дозволяє вам ділитись сокетами між процесами та розподіляти навантаження між ядрами.

### 2.2.2 Фреймоврк Express.js

Express — це мінімальний і гнучкий фреймворк веб-додатків Node.js, який надає надійний набір функцій для розробки веб- та мобільних додатків. Його використання значно скорочує написання коду, а, отже, зменшується час, що витрачається на розробку. Це сприяє швидкому розвитку веб-додатків на основі Node. Нижче наведено деякі з основних функцій Express Framework.

— дозволяє налаштувати проміжне програмне забезпечення для відповіді на HTTP— запити;

— визначає таблицю маршрутизації, яка використовується для виконання різних дій на основі методу HTTP та URL-адреси;

— дозволяє динамічно відображати сторінки HTML на основі передачі аргументів шаблонам.

Express.js є гнучким та мінімалістичним фреймворком додатків. Він побудований навколо конкретних компонентів, що дає розробникам можливість експериментувати. Вони отримують блискавичне налаштування та чистий досвід роботи з JS, що робить Express.js сильним суперником у ніші швидкого прототипування та гнучкої розробки [7]. Express.js може бути використаний для: односторінкових додатків, багатосторінкових додатків, гібридних додатків. Основні характеристики Express.js:

— швидка розробка за сервера;

— дозволяє розробникам швидше створювати RESTful API;

— підтримує архітектуру MVC;

— підтримка NoSQL баз даних з коробки.

Express.js ідеально підходить для швидкого створення веб-застосунків і сервісів, оскільки має легкодоступні інструменти генерації API. Це частина базованої на JavaScript технології MEAN software stack, яка дозволяє використовувати Express.js для створення будь-якої корпоративної браузерної програми.

### 2.2.3 СУБД MongoDB

MongoDB — система управління базами даних, яка працює з документоорієнтованою моделлю даних. На відміну від реляційних СУБД, MongoDB не потребує таблиці, схеми або окремої мови запитів. Інформація зберігається як документів чи колекцій [8].

MongoDB має ряд властивостей, які виділяють її на тлі інших продуктів:

— кросплатформенність. СУБД розроблена мовою програмування C++, тому легко інтегрується під будь-яку операційну систему (Windows, Linux, MacOS та інших.);

— формат даних. MongoDB використовує власний формат зберігання інформації — Binary JavaScript Object Notation (BSON), побудований на основі мови JavaScript;

— документ. Якщо реляційні БД використовують рядки, MongoDB — документи, які зберігають значення і ключі;

— замість таблиць MongoDB використовує колекцію. Вони містять різні типи наборів даних;

— реплікація. Система зберігання інформації у СУБД представлена вузлами. Існує один головний та безліч вторинних. Дані реплікуються між крапками. Якщо один первинний вузол виходить із ладу, то вторинний стає головним;

— індексація. Технологія застосовується до будь-якого поля у документі на розсуд користувача. Проіндексована інформація обробляється швидше;

— для збереження великого розміру даних MongoDB використовує власну технологію GridFS, що складається з двох колекцій. У першій (files) містяться імена файлів та метадані за ними. Друга (chunks) зберігає сегменти інформації, розмір яких не перевищує 256 Кб;

— СУБД здійснює пошук за спеціальними запитами. Наприклад, користувач може створити діапазонний запит та миттєво отримати відповідь.

— балансувальник навантаження використовується в СУБД не тільки для розподілу навантаження між різними базами даних, а й для горизонтального масштабування;

— MongoDB може постачатися для кінцевого клієнта як хмарне рішення.

СУБД використовують для зберігання подій у системі (логування), запису інформації з датчиків моніторингу на підприємстві, а також у сфері електронної комерції та мобільних додатків.

MongoDB належить до класу NoSQL СУБД та працює з документами, а не із записами. Це кросплатформовий продукт, який легко впроваджується у будь-яку операційну систему. Ряд унікальних особливостей дозволяє використовувати СУБД під певні завдання, у яких вона забезпечує максимальну продуктивність та надійність.

MongoDB реалізує новий підхід до побудови баз даних, де немає таблиць, схем, запитів SQL, зовнішніх ключів та багатьох інших речей, які притаманні об'єктно-реляційним базам даних [9].

З часів динозаврів було звичайно зберігати всі дані в реляційних базах даних (MS SQL, MySQL, Oracle, PostgresSQL). При цьому було не так важливо, а чи підходять реляційні бази даних для зберігання даного типу даних, чи ні.

Але, навіть з огляду на всі недоліки традиційних баз даних та переваги MongoDB, важливо розуміти, що завдання бувають різні та методи їх вирішення бувають різні. У якійсь ситуації MongoDB дійсно покращить продуктивність вашої програми, наприклад, якщо треба зберігати складні структури дані. В іншій ситуації краще буде використовувати традиційні реляційні бази даних. Крім того, можна використовувати змішаний підхід: зберігати один тип даних у MongoDB, а інший тип даних у традиційних БД.

Отже, у даному розділі було проаналізовано засоби для розробки медичного інтернет-сервісу для обслуговування та реєстрації пацієнтів та обрано відповідні методи для подальшої розробки.

# 3 РЕАЛІЗАЦІЯ МЕДИЧНОГО ІНТЕРНЕТ- СЕРВІСУ ДЛЯ ОБСЛУГОВУВАННЯ ТА РЕЄСТРАЦІЇ ПАЦІЄНТІВ

Розділ розробки медичного інтернет-сервісу для обслуговування та реєстрації пацієнтів описує реалізацію бакалаврської дипломної роботи. Зі сторони frontend був використаний angular, backend реалізований завдяки node.js та фреймворку express.js. На малюнку 3.1 наведено схему роботи нашого rest серверу.

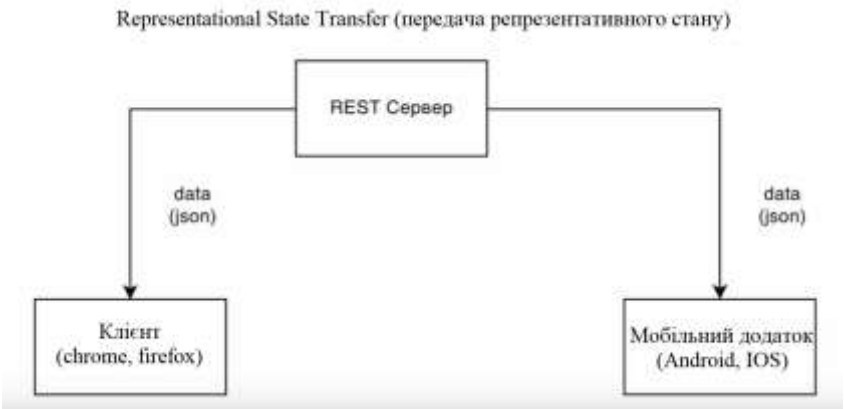



Рисунок 3.1 — Схема роботи звичайного сервера

В бакалаврській дипломній роботі реалізована REST архітектура, яка взаємодіє з допомогою HTTP запитів, виконуючи стандартні функції: створення, оновлення, читання, видалення записів у ресурсі. Дані відправляються на сервер та у зворотньому шляху у форматі JSON. В свою чергу це полегшує роботу, адже ми можемо використовувати один сервер для сайту та мобільного додатку, який працює на операційній системі IOS або Android.

На рисунку 3.2 представлена загальна схема реалізації сервісу, на якому позначено усі наші технології та їхня взаємодія. А саме клієнт через angular відправляє запит на сервер який в своє чергу за допомогою модуля passport.js перевіряє нашого користувача на його права в системі. Після цього express робить звернення до бази даних відносно запиту від клієнта, отримує або відправляє інформацію. Отримавши результат роботи з базою даних, сервер

відправляє на frontend відповідь з результатом роботи, супроводжуючи це статусом виконання.

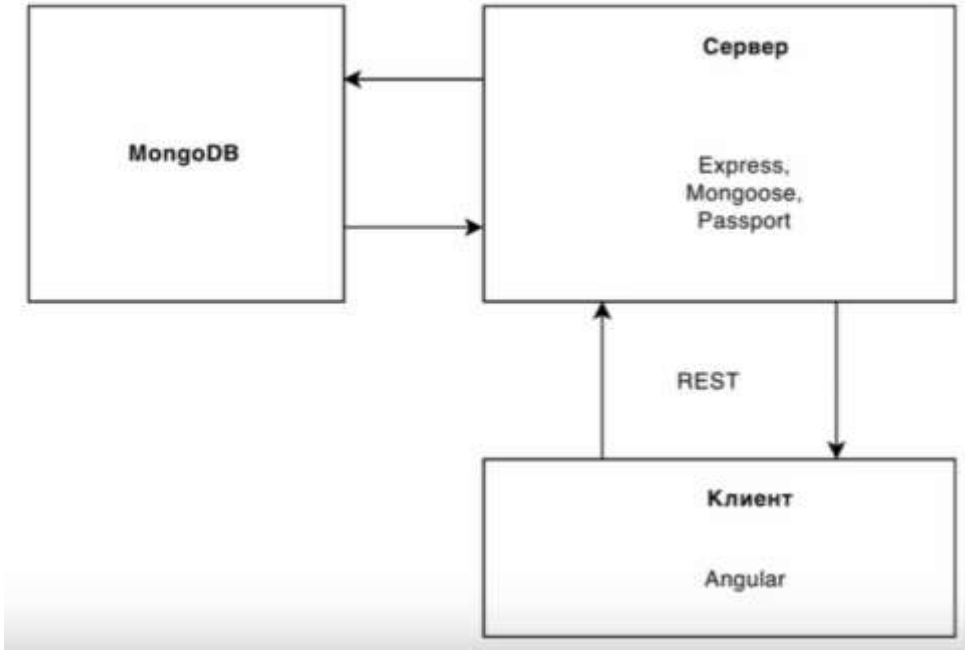



Рисунок 3.2 — Схема роботи сервісу

## 3.1 Реалізація бази даних

В бакалаврській дипломній роботі реалізована база даних MongoDB. Я вибрав її, тому що це NoSQL база, вона може зберігати дані у форматі JSON. Реалізовано 5 колекцій: doctor-specialities, doctors, medicalinstitutions, orders, users. Кожна з них містить необхідний набір полів для роботи. Кожне поле містить певний опис, а саме назву, значення за замовчуванням, тип даних та інше. Колекція users містить поля для введення електронної пошти, пароля, ініціалів та роді користувача. Дана база є дуже гнучкою у використанні, з легкістю обробляє великі об’єми даних, оскільки забезпечує високоефективне рішення для їхнього зберігання [10]. Замість таблиць база даних MongoDB зберігає свої дані в колекціях. Колекція містить один або кілька документів JSON. Вони аналогічні записам або рядкам у таблиці реляційної бази даних. Кожен документ має одне або кілька полів; поля подібні до стовпців у таблиці

реляційної бази даних. На лістингу 3.1 наведена схема моделі медичної організації.

Лістинг 3.1 — Схема моделі медичної організації

```
const medicalInstitutionSchema = new Schema({
    fullName: {
        type: String,
        required: true},
    shortName: {
        type: String,
        required: true},
    address: {
        type: String,
        required: true},
    imageSrc: {
        type: String,
        default: ''},
    offices: [{
            typeId: {
                type: String,
                required: true},
            typeName: {
                type: String,
                required: true},
            number: {
                type: Number,
                required: true},
            doctors: {
                ref: 'doctors',
                type: Schema.Types.ObjectId},
            schedule: {
                workingDays: [{
                        type: String,
                        required: true}],
                weekendDayStart: {
                    type: String,
                    required: true },
                weekendDayEnd: {
                    type: String,
                    required: true} } }  ]})
```

## 3.2 Реалізація frontend

В даному проєкті для розробки я використав фреймворк Angular13. На даному етапі він повністю задовольняє мої потреби в реалізації даного інтернет-сервісу. Я обрав його через використання в ньому TypeScript та через те, що його структура та архітектура спеціально створені для великої масштабованості проєкту.

При розробці я дотримувався правила, щоб максимально розбивати проект на модулі, для кращої масштабованості та зручної розробки [11]. Адже кожен модуль повинен відповідати виключно за свій функціонал, так само і всі компоненти повинні бути прив’язаними до певного модуля, і міститись з ним в одній папці.

Папка features містить основні файли сторінки. Кожна сторінка містить окрему папку, в яку своя структура з сервісами, компонентами, розміткою, станом, стилями, моделями, типами, інтерфейсами, загальними модулями та модулями та модулями авторизації. Директорія shared містить загальні компоненти, які будуть використовуватися по всьому проєкті. Також міститься велика кількість інтерфейсів, класів, директив, пайпів, сервісів та модулів. В папці assets містяться файли стилів. Кожен файл відповідає за стилізування певної частини. Створено загальні класи, які відповідають за розташування, стилізацію, відступи, полегшують роботу з кольорами, містяться готові частини, які імпортуються  локальні файли. Також було налаштовано системний файл tsconfig.json в якому налаштована робота TypeScript компілятора.

### 3.2.1 Авторизація та реєстрація на стороні клієнта

Авторизації та реєстрації в даній роботі була виділена велика увага. Адже безпека користувача це одна із найважливіших тем.

Процес авторизації дуже важливий, адже ми повинні створити для користувача всі умови, щоб це було зручно, швидко та безпечно. Кожне поле проходить певну валідацію. Вони є обов’язковими для введення та мають

обмеження на кількість символів для введення, додатково email перевіряється на коректність [12]. Кнопка для відправки даних залишається неактивною, поки дані не будуть введені правильно. Це реалізовано за допомогою бібліотеки NgRx. Вони в свою чергу відповідає за локальне зберігання даних. Таким чином при відправці даних ми фіксуємо інформацію, що запит надіслано на сервер і помічаємо, що поки він в обробці, ми блокуємо доступ для відправлення повторних запитів. Таким чином ми сприяємо вирішенню певних неочікуваних помилок зі сторони сервера та зменшуємо навантаження на нього і тим самим покращуємо продуктивність.

Також зі сторони бекенду є багато додаткових перевірок. При успішній авторизації ми отримуємо повідомлення про успіх, а при помилці отримуємо сповіщення для подальших дій. Внизу в нас є повідомлення, яке вказує, що ми можемо зареєструватися, якщо у нас відсутній акаунт. Дані процеси описані на рисунках 3.3 і 3.4.

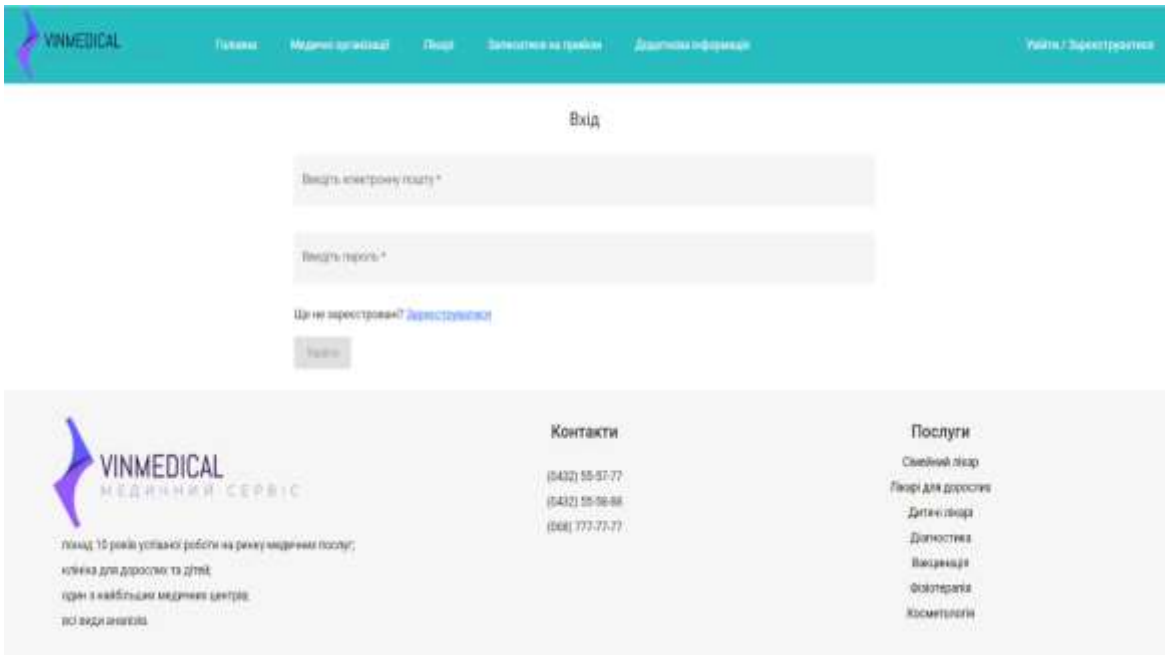



Рисунок 3.2 — Сторінка авторизації

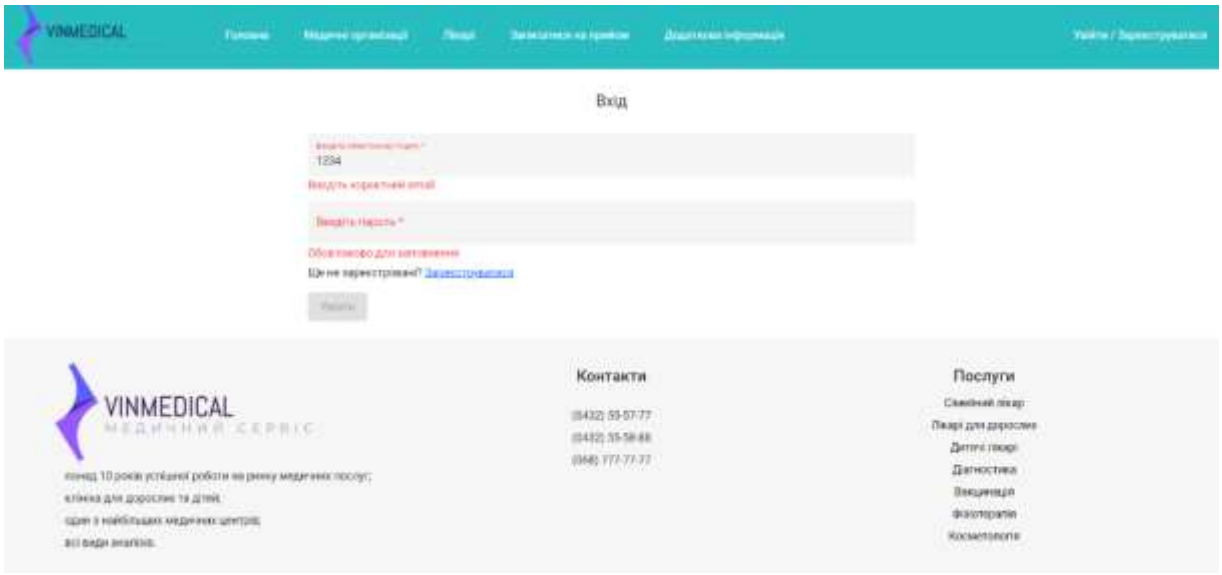



Рисунок 3.4 — Приклад валідації полів

Процес реєстрації відповідає таким же самим принципам, що і авторизації. Але для нього нам потрібно, щоб користувач вказав дані в полі для введення ініціалів. Процес відправлення запиту на сервер відбувається за допомогою бібліотеки NgRx.

Вона надає гнучкий функціонал для відокремлення для того, щоб ми могли відокремити логіку роботи з сервером в окремих файлах за заданою структурою. Запити ми реалізовуємо в ефектах [13].

Після натискання кнопки зареєструватися у нас відбувається подія, при якій ми змінюємо стан нашого компонента, і розміщуємо інформацію, що відбувається процес і ми блокуємо під час нього нашу кнопку відправлення, щоб користувач не зміг натискати на неї велику кількість раз, тим самим, навантажуючи сервер.

Подія одразу запускає запит на сервер в ефектах, використовуючи сервіс. Після успішного чи невдалого завершення, користувач знову має можливість реєстрації.

На рисунку 3.5 приклад сторінки реєстрації.

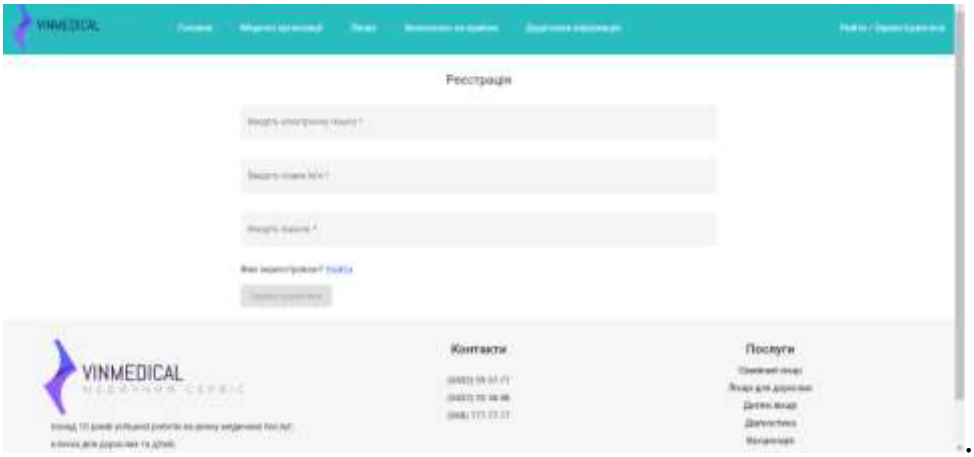



Рисунок 3.5 — Приклад сторінки реєстрації

Лістинг 3.2 — Реалізація сторінки реєстрації

```
<div class="text— center mt30 title">Реєстрація</div>
<div class="w— 50 m— auto m— block20"
     [formGroup]="form">
  <div class="mt30 d— block">
    <app— input
      [required]="true"
      [label]="'Введіть електронну пошту'"
      [control]="getFormControl(form.get('email'))"></app— input></div>
  <div class="mt30 d— block">
    <app— input
      [required]="true"
      [label]="'Введіть повне ім\'я'"
      [control]="getFormControl(form.get('fullName'))"
    ></app— input></div>
  <div class="mt30 d— block">
    <app— input [type]="'password'" [required]="true" [label]="'Введіть
пароль'"[control]="getFormControl(form.get('password'))"
    ></app— input></div>
  <div class="mt25">Вже зареєстровані? <a [routerLink]="['../',
'login']">Увійти</a></div>
  <button (click)="onSubmit()"
    [disabled]="(buttonSubmitStatus | async) || form.invalid"
    mat— raised— button
    color="primary" type="submit" class="mt15">
    Зареєструватися</button></div>
```

Продовження лістингу 3.2

```
@Component({
  selector: 'app– register',
  templateUrl: './register.component.html',
  styleUrls: ['./register.component.scss']
})
export class RegisterComponent extends Base implements OnInit {
  public form: FormGroup;
  public buttonSubmitStatus: Observable<boolean> =
this.store.pipe(select(isSubmittingSelector));
  constructor(
    private store: Store<AppStateInterface>
  ) {super();}
  ngOnInit(): void {
    this.initializeForm();}
  private initializeForm(): void {
    this.form = new FormGroup({
      email: new FormControl(null, [Validators.required, Validators.email]),
      password: new FormControl(null, [Validators.required, Validators.minLength(10)]),
      fullName: new FormControl(null, [Validators.required, Validators.minLength(15),
Validators.maxLength(80)])})}
  public onSubmit(): void {
    this.store.dispatch(registerAction({ request: this.form.value }));
  }}
```

### 3.2.2 Блок медичних організацій

В бакалаврській дипломній роботі реалізовано блок з медичними організаціями. Там ми маємо можливість знайти організацію, переглянути потрібну організацію, отримати інформацію про лікарів та кабінети в яких вони працюють з графіком роботи. А саме головне ми маємо можливість записатися до лікаря [14]. На сторінці є пошук, завдяки чому кожен користувач має можливість знайти потрібну йому організацію. Реалізація пошуку працює за співпадінням символів, та не чутлива до регістру. На рисунку 3.6 приклад сторінки зі списком медичних організацій.

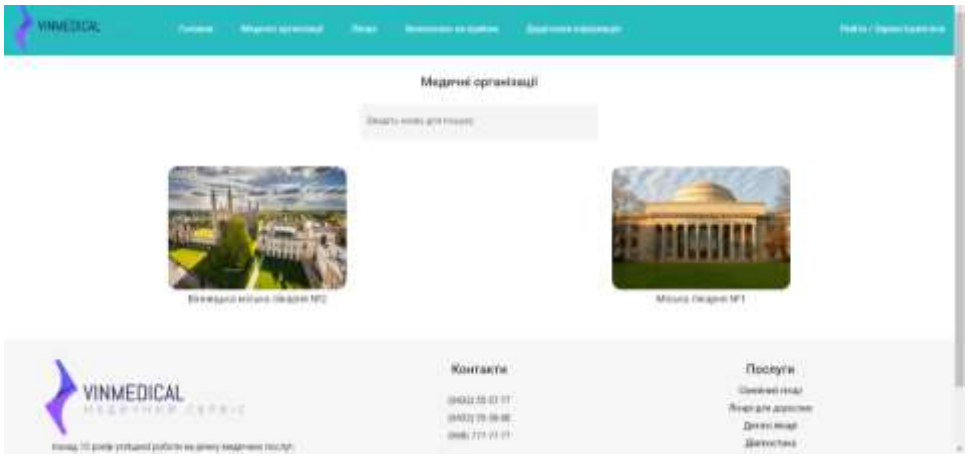



Рисунок 3.6 — Приклад сторінки зі списком медичних організацій

Перемикання між сторінками з лікарями, кабінетами та записом до лікаря відбувається за допомогою табів. Вони завантажуються динамічно, та тільки за викликом, що є позитивним для продуктивності нашого сервісу. Реалізовано це за допомогою lazy loading. Інформація про це міститься в модулях налаштування маршрутів. На рисунку 3.7 зображена сторінка з лікарями.

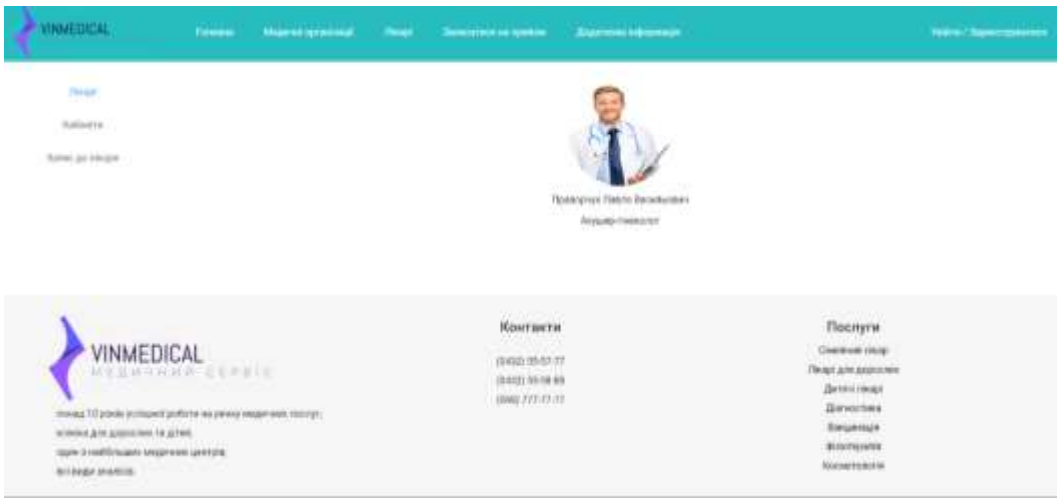



Рисунок 3.7 — Приклад сторінки зі списком лікарів

На рисунку видно, активний таб на якому ми знаходимось, на вкладці з кабінетами знаходиться інформація про кабінети в лікарні, там розміщено номер кабінету, спеціальність лікаря, та час прийому. При перемиканні на кожну з цих сторінок ми отримуємо дані з локального місця зберігання, відносно того, який

url адрес. Виконано вивід даних для за допомогою системної директиви ngFor, яка при кожній ітерації виводить певний об’єкт і за допомогою інтерполяції ми виводимо необхідну інформацію.  На рисунку 3.8 інформація про кабінети.


Рисунок 3.8 —  Приклад сторінки зі списком кабінетів

### 3.2.3 Спеціальності лікарів

В бакалаврській дипломній роботі реалізована сторінка в якій міститься інформація про спеціальності лікарів. Кожен охочий може перейти і ознайомитись зі всіма спеціальностями. Також на сторінці є налаштований пошук. Кожна сторінка при наведенні збільшується. Спеціальності в проекті використовують не тільки на даній сторінці але і в інших, адже ми маємо певний функціонал, де потрібно вказати спеціальність лікаря [15]. Наприклад сторінки де ми добавляємо лікаря або показуємо його спеціальність, чи реєструємося на прийом. На рисунку 3.9 зображена сторінка з спеціальностями.


Рисунок 3.9 —  Приклад сторінки зі списком кабінетів

### 3.2.4 Ролі користувача

В бакалаврській дипломній роботі реалізовані ролі, щоб кожна людина мала певний доступ до певного функціоналу. Було створено ролі такі як: Admin, Doctor, User.

Користувач Admin має найбільші можливості у нашій системі і немає жодних обмежень. Відносно ролі користувача ми надаємо йому певні можливості і в кожного відображається дещо інший інтерфейс. Адміністратор має можливість додати нову медичну організацію, оновити існуючу або видалити її.

Також він може добавляти, редагувати та видаляти для кожної лікарні медичний персонал та кабінети, та прив’язувати до них певних лікарів. Кожен звичайний користувач має змогу записатися на прийом до лікаря з можливістю вибору лікаря. Приклад сторінки доступної тільки адміністратору наведено на рисунку 3.10

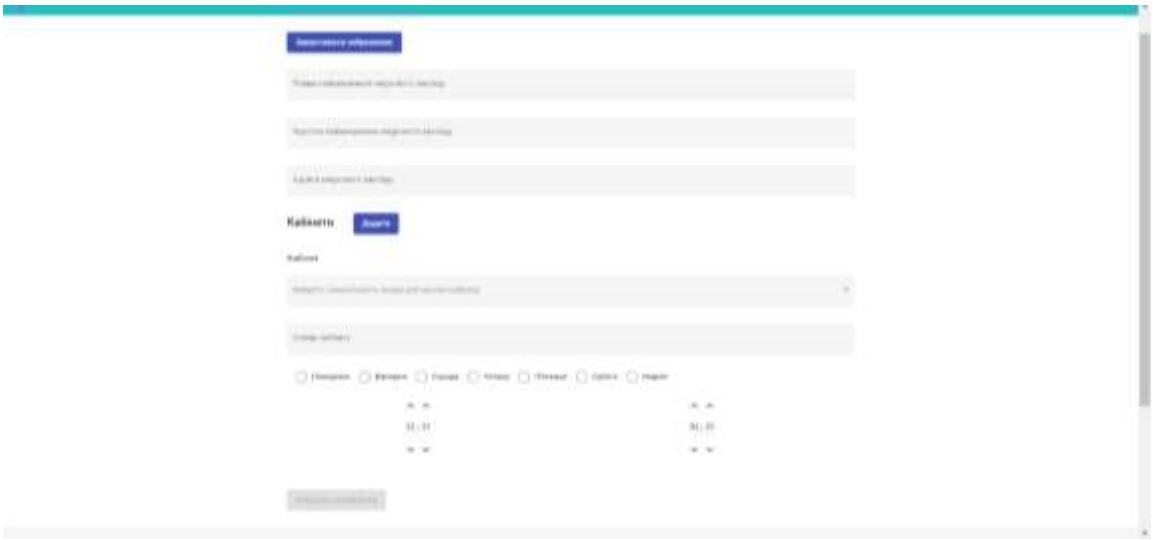


Рисунок 3.10 — Приклад сторінки для створення медичної організації для адміністратора

Але відносно ролі ми можемо приховати не тільки сторінку, а ще елемент. Це виконано за допомогою спеціальної директиви. Приклад директиви на лістингу 3.3.

Лістинг 3.3 — Реалізація директиви

```
@Directive({
  selector: '[appShowForRoles]'
})
export class HideForRolesDirective extends Base implements OnInit {
  @Input('appShowForRoles') set showRoles(roles: Array<string>) {
    if (roles.length) {
      this.checkRoles(roles);
    } else {
      this.viewContainer.createEmbeddedView(this.templateRef);
    }}
  constructor(
    private templateRef: TemplateRef<any>,
    private viewContainer: ViewContainerRef,
    private store: Store<AppStateInterface>
  ) {
    super();}
  public ngOnInit(): void {}
  private checkRoles(expectedRoles: Array<string>) {
    new Promise((resolve) => {
      this.store.pipe(
        select(rolesSelector),
        filter(v => !!v),
        takeUntil(this.destroy$))
.subscribe((res: Array<string>) => resolve(res));
    })
.then((userRoles: Array<string>) => {
      if (!userRoles.length) {
        this.viewContainer.clear();
      } else {
        const userMatch = expectedRoles.findIndex(idx =>
userRoles.includes(idx));
        userMatch >= 0 ?
this.viewContainer.createEmbeddedView(this.templateRef) :
          this.viewContainer.clear();
      }
})}}
```

### 3.2.5 Налаштування WebStorm IDE

Для розробки проєкту було використано редактор Webstorm. це інтегроване середовище для розробки на JavaScript та пов'язаних з ним технологіях. Він дозволяє автоматизувати рутинну роботу та легко справлятися зі складними завданнями, роблячи розробку більш цікавою.

Загалом, компанія JetBrains надає розумні IDE для різних мов програмування. Тим самим заохочуючи все більше нових клієнтів. Для стуентів є можливість користуватися ліцензійним програмним забезпеченням WebStorm на безоплатній основі. В свою чергу це заоохочує їх стати майбутніми клієнтами та розробляти нові продукти якісними і зручними.

WebStorm чудово розуміється на структурі проєктів і допомагає розробникам у всіх аспектах написання коду. Код Autodill, безпечний надійний помічник, що допомагає здійснювати пошук потенційних проблем і поради щодо їх вирішення завжди будуть у вас під рукою.

Однією з головних переваг IDE є те, що вони інтегрують усі необхідні інструменти. Використовуйте WebStorm для налаштування та тестування клієнтського коду та програм Node.js, а також для роботи з системами керування версіями. Є можливість використовувати інтегровані вкладиші, інструменти збірки, термінал і HTTP-клієнт.

Незалежно від розміру проєкту, WebStorm швидко запускає код. Можна шукати файли, класи чи коди, переглядаючи всі результати в одному місці. Переходити до визначення функцій, методів, змінних, компонентів або класів, знайдіть їх використання за кілька кліків.

Наявна функція розміщення нових співробітників —усі функції доступні з коробки. Розробляйте програму в режимі реального часу разом із командою та спілкуйтеся безпосередньо з колегами з IDE. Спільний доступ до конфігурацій проекту, включаючи налаштування стилю коду. Отримайте максимум користі від інтеграції з Git і GitHub.

### 3.2.6 Процес додавання лікаря

На даній сторінці був реалізований функціонал додавання та створення медичного працівника з прив’язкою до медичної організації. Таким чином, кожна медична організація містить перелік кабінетів, в кожному з яких працює певний спеціаліст. Кожен кабінет містить свій тип, номер, лікаря та графік роботи. Адміністратор в свою чергу без жодних проблем може змінити лікаря в кожному кабінеті та усю інформацію про нього.

При створенні медичного персоналу ми повинні вказати таку інформацію як: прізвище ім’я та ім’я по-батькові, освіту, обрати спеціальність лікаря та його дату народження. На кожному полі реалізована перевірка на валідність даних, а на поле дати народження реалізовано обмеження на вік лікаря. Таким чинок лікар не може бути молодший 21-го року — система не дає можливості продовжити реєстрацію. Також, є можливість розміщення фото лікаря на веб-сайті, тим самим кожен охочий може обрати потрібного медичного працівника та ознайомитись з його даними.  Приклад реалізації даної сторінки розміщений на рисунку 3.11

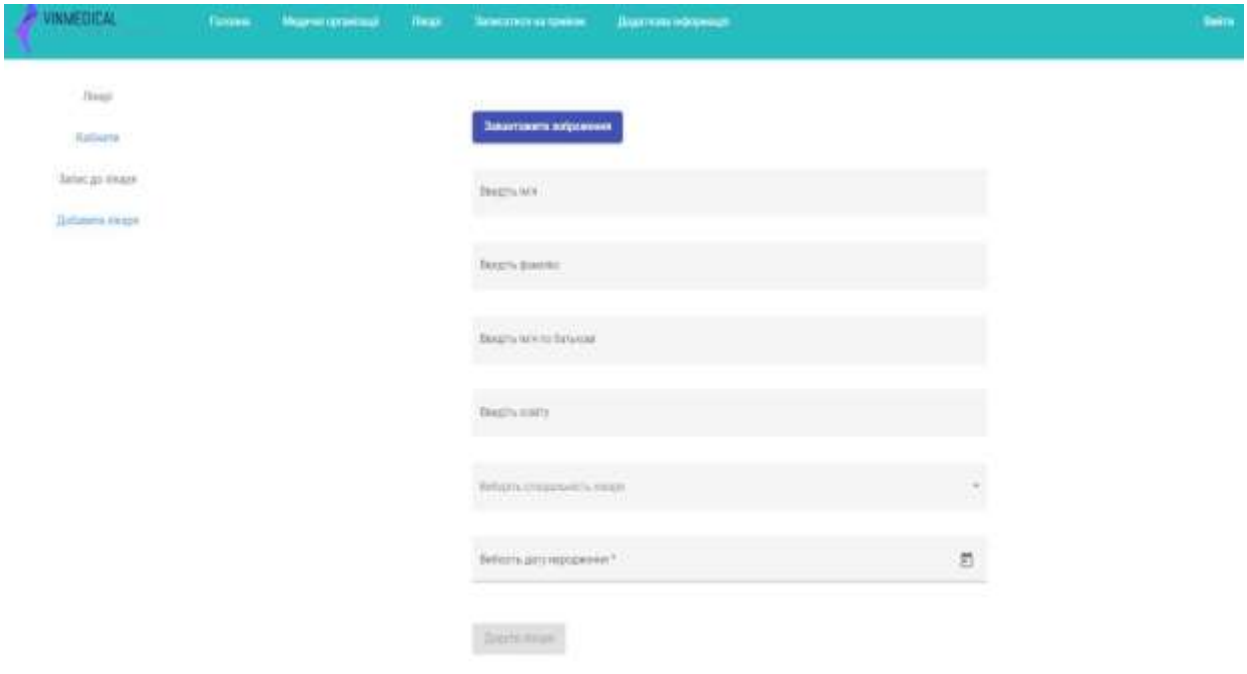



Рисунок 3.11 —  Процес додавання лікаря

## 3.3 Реалізація backend

В бакалаврській дипломній роботі зі сторони backend я використав платформу node.js з використанням фреймворку express.js. Для забезпечення захисту був використаний passport.js — це проміжне програмне забезпечення для аутентифікації. Підтримує її за допомогою імені користувача та пароля, Facebook, Twitter тощо. Була використана бібліотека moment.js, яка є однією з найкращих для роботи з часом, що значнго пришвидшує та спрощує розробку.

Структура файлів дуже зручна та гнучка для масштабованості. В папці models зібрані моделі колекцій для нашої бази даних. У бакалаврській дипломній роботі реалізовано, спроектовано, змодельовано поля, валідацію даних та структуру колекцій. В routes в нас зібрані маршрути, по яким ми надсилаємо запити до нашого серверу. В кожному маршруті ми описуємо шлях, Controllers описують дії, які ми виконуємо при певному запиті. В них містяться звернення до бази даних, це зберігання, видалення, оновлення та отримання інформації, а також фільтрація та сортування даних [17]. В директорії utils зберігається усі зображення нашого сайту, фото лікарів та медичних організацій. В папці middleware знаходяться різні допоміжні функції, які реалізують необхідний функціонал для коректної роботи бакалаврської дипломної роботи.

Директорія keys містить ключі він нашого сайту. А саме код шифрування даних при реєстрації та вході в систему та ключ підключення до бази даних mongoDB.

### 3.3.1 Passport.js модуль

Реалізація зручного та високофункціонального способу авторизації за допомогою як і звичайних даних, так і facebook, google та інших. Для роботи ми вказуємо стратегію, за якою даний модуль буде працювати, а також ключ шифрування даних. Створюємо функцію, яка по токену, який приходить з frontend частини, знаходить даного користувача, та перевіряє токен на

працездатність. Після чого у разі негативного сценарію відмовляє в доступі, інакше надає необхідний доступ до системи. На рисунку 3.12 структура passport.js в бакалаврській дипломній роботі.

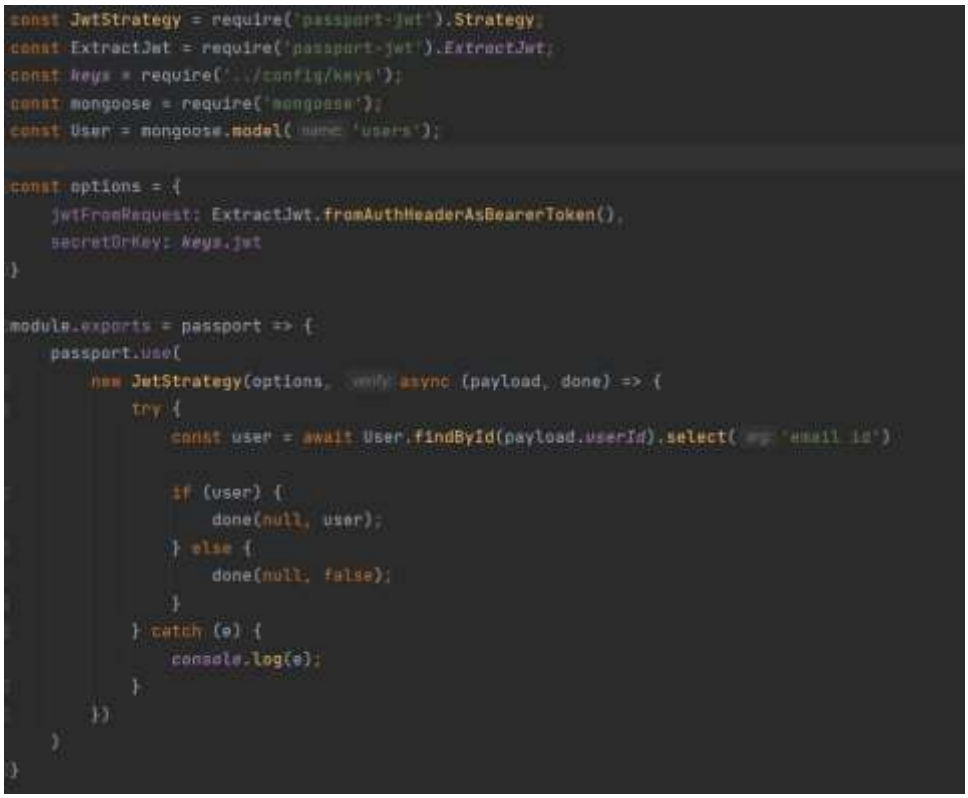



Рисунок 3.12 — Реалізація модуля passport.js

### 3.3.2 Авторизація та реєстрація на стороні сервера

Процес реєстрація користувача дуже цікавий. При реєстраці. Ми повинні одразу перевірити, чи є в нас користувач зі вказаним email адресом. У разі його унікальності ми створюємо нового користувача, але ми не можемо зберігати його пароль у відкритому вигляді, том ми повинні його зашифрувати [18]. Під час шифрування ми повинні вказати ключ та складність. Цим займається спеціальна бібліотека під назвою bcrypt. Приклад використання бібліотеки зазначений на лістингу 3.4.

Лістинг 3.4 — реалізація шифрування пароля

```
const salt = bcrypt.genSaltSync(10);
const password = req.body.password;
```

Продовження лістингу 3.4

```
const user = new User({
    email: req.body.email,
    password: bcrypt.hashSync(password, salt),
    fullName: req.body.fullName,
    roles: ['User']
});
```

Процес авторизація вигляде наступним чином. Ми відправляємо запит з нашими даними і одразу шукаємо нашого користувача в базі даних, якщо ми його не знаходимо, то отримуємо повідомлення, що йому потрібно зареєструватися. Далі ми звіряємо паролі, та уразі того, якщо вони збігаються, ми генеруємо токен та відправляємо його на фронтенд частину [19]. Зберігаємо його в localstorage, адже при кожному наступному відправленні токена буде зручно дістати його та надіслати в заголовку до запиту. Він забезпечить доступ користувачу до системи. І буде відправлятись кожен раз при наступних запитах до сервера, тим самим підтверджуючи особу користувача.

Безпосередньо в токені зашифровані унікальний ідентифікатор користувача та його електронна пошта. І при кожному запиті passport.js буде перевіряти токен на корректність та час дії. Адже кожен токен має свій час життя, протягом якого він є активним. У нашому випадку токен живе одн годину [20]. На рисунку 3.13 наведено приклода JWT токена який приходить після успішної авторизації.

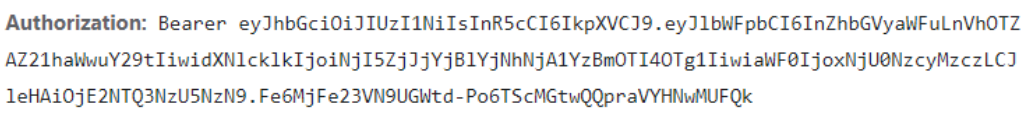



Рисунок 3.13 — JWT(Json Web Token)

### 3.3.3 Модуль завантаження зображень

В бакалаврській дипломній роботі реалізована можливість завантаження зображення. А саме при створені медичної організації та лікаря. Цей процес

відбувається завдяки бібліотеці multer, яка забезпечує даний функціонал [21]. Вона є досить гнучкою та простою у використанні. При налаштування ми повинні вказати декілька основних речей. А саме місце розташування картинок, тобто місце куди вони будуть завантажуватись, далі ми присвоюємо кожній унікальне ім’я для ідентифікації [22].

Вказуємо формати зображень, які ми хочемо зберігати, наприклад можемо зберігати картинки тільки jpeg або png. Та обмежуємо користувача розмірами зображень. Адже тут нам не потрібні зображення великого розміру і в свою вони займають багато місця, що є не дуже добре. Кожна функція приймає три параметри, перший це безпосередньо сам запит, другий це наше зображення, яке ми завантажуємо та третє зображення як певна функція, яка має певну реалізацію.

Даний модуль ми реєструємо в головному файлі index.js. При експорті необхідно вказати блок код з інформацією про місце зберігання та назву, Фільтрацію даних відносно типу самого зображення та його розмір. На рисунку 3.14 наведено приклад налаштування multer для завантаження зображень в бакалаврській дипломній роботі.


Рисунок 3.14 — Налаштування multer

# ВИСНОВКИ

У бакалаврській дипломній роботі було проаналізовано існуючі медичні інтернет-сервіси для обслуговування та реєстрації пацієнтів.

В проекті був виконаний аналіз сучасних медичних інтернет–сервісів, розглянуто аналоги та можливості застосування. Були проаналізовані засоби розробки клієнтської та серверної частини медичного інтернет–сервісу. Для розробка даного сервісу був використаний JavaScript фреймворки Angular, бібліотека для роботи зі станом NgRx та UI бібліотеки такі як: RxJS, Angular material, Bootstrap, Moment та інші. Зі сторони сервера використано node.js фреймворк Express.js, passport.js для авторизації. В якості бази даних було використано MongoDB.

В результаті виконання дипломної роботи розроблено медичний інтернет-сервіс, який допомагає обслуговувати та реєструвати пацієнтів в режимі онлайн. Наразі він працює злагоджено та без жодних проблем.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. What is Angular (Що таке Ангуляр). [Електронний ресурс]. Режим доступу: https://angular.io/guide/what— is— angular.

2. Фреймворк NGRX. [Електронний ресурс]. Режим доступу: https://ngrx.io/.

3. Бібліотека RXJS. [Електронний ресурс]. Режим доступу: https://rxjs.dev/

4. Бібліотека Material Angular. [Електронний ресурс]. Режим доступу: https://material.angular.io/

5. Бібліотека Bootstrap. [Електронний ресурс]. Режим доступу: https://ng— bootstrap.github.io/#/home

6. Програмна платформа Node.js. [Електронний ресурс]. Режим доступу: https://nodejs.org/uk/

7. Фреймворк Express.js. [Електронний ресурс]. Режим доступу: https://expressjs.com/ru/

8. СУБД MongoDB. [Електронний ресурс]. Режим доступу: https://www.mongodb.com/

9. Переваги MongoDB. [Електронний ресурс]. Режим доступу: https://uk.education— wiki.com/5891922— advantages— of— mongodb

10. Бібліотека lodash. [Електронний ресурс]. Режим доступу: https://lodash.com/

11. Бібліотека moment.js. [Електронний ресурс]. Режим доступу: https://momentjs.com/

12. Stack Overflow. [Електронний ресурс]. Режим доступу: https://stackoverflow.com/

13. Бібліотека ng-select. [Електронний ресурс]. Режим доступу: https://github.com/ng-select/ng-select

14. Модуль datetime-picker. [Електронний ресурс]. Режим доступу: https://www.npmjs.com/package/@angular-material-components/datetime-picker

15. Менеджер npm. [Електронний ресурс]. Режим доступу: https://www.npmjs.com/

16. JavaScript документація. [Електронний ресурс]. Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript

17. Бібліотека multer. [Електронний ресурс]. Режим доступу: https://www.npmjs.com/package/multer

18. Бібліотека passport. [Електронний ресурс]. Режим доступу: https://www.passportjs.org/

19. Веб документація [Електронний ресурс]. Режим доступу: https://webreference.com/

20. HTML документація. [Електронний ресурс]. Режим доступу: https://htmlreference.io/

21. HTML, CSS документація. [Електронний ресурс]. Режим доступу: https://www.w3schools.com/

22. Приклади реалізації коду. [Електронний ресурс]. Режим доступу: https://metanit.com/

51

**ДОДАТОК А**

Технічне завдання

Міністерство освіти та науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ ВНТУ

д.т.н., проф.

О. Д. Азаров

"29" 04 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврського дипломного проекту

«Медичний інтернет-сервіс для обслуговування та реєстрації пацієнтів»

08-23.БДР.039.00.000 ПЗ

Науковий керівник: доцент к.т.н.

Колесник І.С.

Студент групи 1КІ-20м

Ткачук В.М.

Вінниця 2022

1. Найменування та область застосування

Робоча назва проекту «Медичний інтернет— сервіс для обслуговування та реєстрації пацієнтів», обслуговування та реєстрації пацієнтів

2. Основи для розробки

Основою для розробки є сучасні веб технології.

3. Мета та призначення розробки

Розробка медичного інтернет-сервісу для обслуговування та реєстрації пацієнтів, який допоможе забезпечувати надання медичної допомоги пацієнту в режимі онлайн.

4. Етапи виконання роботи та очікувані результати

Робота виконується в вісім етапів, що наведені в таблиці А.1.

Таблиця А.1— Етапи виконання роботи

| № етапу | Назва етапу | Термін виконання | | Очікувані результати |
|---|---|---|---|---|
| | | початок | кінець | |
| 1 | Постановка задачі роботи | 11.02.22 | 11.02.22 | Вступ |
| 2 | Аналіз сучасних інтернет-сервісів для обслуговування та реєстрації пацієнтів | 12.02.21 | 28.02.21 | Розділ 1 |
| 3 | Вибір засобів для розробки медичного інтрнет-сервісу для обслуговування та реєстрації пацієнтів | 01.03.21 | 12.03.21 | Розділ 2 |
| 4 | Проектування медичного інтрнет-сервісу | 13.03.21 | 17.04.21 | Розділ 3 |
| 5 | Підготовка матеріалів та опис розробки | 20.04.21 | 01.05.21 | ПЗ, презентація |
| 6 | Оформлення пояснювальної записки та ілюстративного матеріалу | 03.05.21 | 30.05.21 | |
| 7 | Аналіз виконання роботи, висновки, додатки | 01.06.21 | 05.06.21 | |
| 8 | Перевірка якості виконання бакалаврського проекту та усунення недоліків | 25.05.22 | 25.05.22 | |

53

5. Матеріали, що подаються до захисту бакалаврської дипломної роботи

Пояснювальна записка, графічні і ілюстративні матеріали, відзив наукового керівника, рецензія опонента, анотації українською та іноземною мовами, довідка про відповідність оформлення бакалаврської дипломної роботи діючим вимогам.

6. Порядок контролю виконання та захисту БДР

Виконання етапів графічної та розрахункової документації БДП контролюється науковим керівником згідно зі встановленими термінами. Захист БДП відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

7. Вимоги до оформлення БДР

— ДСТУ 3008 : 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302 : 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104-2006 «Единa система конструкторської документації. Основні написи»;

— Документами на які посилаються у вище вказаних.

Технічне завдання до виконання отримав _____ Ткачук В.М.

## ДОДАТОК Б

Файл обробки запитів авторизації

```
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const User = require('../models/user');
const keys = require('../config/keys');
const errorHandler = require('../utils/errorHandler');
module.exports.login = async function (req, res) {
    const candidate = await User.findOne({email: req.body.email});
    if (candidate) {
        const passwordResult = bcrypt.compareSync(req.body.password,
candidate.password);
        if (passwordResult) { // generate token
            const token = jwt.sign({
                email: candidate.email,
                userId: candidate._id
            }, keys.jwt, {expiresIn: 60 * 60});
            res.status(200).json({
                token: `Bearer ${token}`,
                email: candidate.email,
                fullName: candidate.fullName,
                id: candidate._id,
                roles: candidate.roles})
        } else {  res.status(401).json({
                message: 'Паролі не співпали. Спробуйте знову'
            })}} else {
```

```
        res.status(404).json({
            message: 'Користувача з таким email не знайдено'
        })}}
module.exports.register = async function (req, res) {
    const candidate = await User.findOne({email: req.body.email});
    if (candidate) {
        res.status(409).json({
            message: 'Такий email вже зайнятий, спробуйте інший'
        })} else {
        const salt = bcrypt.genSaltSync(10);
        const password = req.body.password;
        const user = new User({
            email: req.body.email,
            password: bcrypt.hashSync(password, salt),
            fullName: req.body.fullName,
            roles: ['User']});
        try {
            await user.save();
            res.status(201).json(user)
        } catch (error) {
            errorHandler(res, error);}}}
module.exports.getUserByJWT = async function (req, res) {
    const candidate = await User.findOne({ _id: req.user._id });
    console.log(candidate);
    try {
        res.status(200).json({
            email: candidate.email,
            fullName: candidate.fullName,
            id: candidate._id,
            roles: candidate.roles
```

```
    })} catch (error) {
    errorHandler(res, error);}}
```

## ДОДАТОК В

Файл обробки запитів медичної організації

```
const MedicalInstitution = require('../models/medical— institution');
const errorHandler = require('../utils/errorHandler');
module.exports.create = async function (req, res) {
  try {
    req.body.offices = JSON.parse(req.body.offices);
    const newMedicalInstitution = await new MedicalInstitution(
      {
        ...req.body,
        imageSrc: req.file ? req.file.path: ''
      }).save();
    res.status(201).json(newMedicalInstitution);
  } catch (e) {
    errorHandler(res, e);
  }}
module.exports.getById = async function (req, res) {
  try {
    const medicalInstitution = await MedicalInstitution.findById(req.params.id);
    res.status(200).json(medicalInstitution);
  } catch (e) {
    errorHandler(res, e);}}
module.exports.getAll = async function (req, res) {
  try {
    const medicalInstitutions = await MedicalInstitution.find();
    res.status(200).json(medicalInstitutions);
  } catch (e) {
```

```
        errorHandler(res, e);
    }}
module.exports.update = async function (req, res) {
    try {
        if (req.file) {
            req.body.imageSrc = req.file.path;}
        const medicalInstitutions = await MedicalInstitution.findOneAndUpdate(
            { _id: req.params.id },
            { $set: req.body },
            { new: true });
        res.status(200).json(medicalInstitutions);
    } catch (e) {errorHandler(res, e);}}
module.exports.delete = async function (req, res) {
    try {
        await MedicalInstitution.remove({ _id: req.params.id });
        res.status(200).json({
            message: 'Позиція була видалена'
        })} catch (e) {
        errorHandler(res, e);}}
```

## ДОДАТОК Г

Файл встановлених пакетів


```
{
  "name": "client",
  "version": "0.0.0",
  "scripts": {
    "ng": "ng",
    "start": "ng serve --proxy-config proxy.conf.json",
    "build": "ng build",
    "watch": "ng build --watch --configuration development",
    "test": "ng test"},
  "private": true,
  "dependencies": {
    "@angular-material-components/datetime-picker": "^7.0.1",
    "@angular-material-components/moment-adapter": "^7.0.0",
    "@angular/animations": "~13.0.0",
    "@angular/cdk": "^13.2.5",
    "@angular/common": "~13.0.0",
    "@angular/compiler": "~13.0.0",
    "@angular/core": "~13.0.0",
    "@angular/forms": "~13.0.0",
    "@angular/material": "^13.2.5",
    "@angular/material-moment-adapter": "^12.2.13",
    "@angular/platform-browser": "~13.0.0",
    "@angular/platform-browser-dynamic": "~13.0.0",
    "@angular/router": "~13.0.0",
    "@ng-bootstrap/ng-bootstrap": "^12.0.1",
    "@ng-select/ng-select": "^8.1.1",
```

```
  "@ngrx/effects": "^13.2.0",
  "@ngrx/store": "^13.2.0",
  "@ngrx/store-devtools": "^13.2.0",
  "@popperjs/core": "^2.11.4",
  "bootstrap": "^5.1.3",
  "lodash": "^4.17.21",
  "moment": "^2.29.3",
  "rxjs": "~7.4.0",
  "tslib": "^2.3.0",
  "zone.js": "~0.11.4"},
"devDependencies": {
  "@angular-devkit/build-angular": "~13.0.4",
  "@angular/cli": "~13.0.4",
  "@angular/compiler-cli": "~13.0.0",
  "@angular/localize": "^13.0.3",
  "@types/jasmine": "~3.10.0",
  "@types/node": "^12.11.1",
  "jasmine-core": "~3.10.0",
  "karma": "~6.3.0",
  "karma-chrome-launcher": "~3.1.0",
  "karma-coverage": "~2.0.3",
  "karma-jasmine": "~4.0.0",
  "karma-jasmine-html-reporter": "~1.7.0",
  "typescript": "~4.4.3"}}
```

## ДОДАТОК Д

Головний файл backend

```
const express = require('express');
const mongoose = require('mongoose');
const passport = require('passport');
const bodyParser = require('body— parser');
const authRoutes = require('./routes/auth');
const medicalInstitutionRoutes = require('./routes/medical— institution');
const orderRoutes = require('./routes/order');
const doctorsRoutes = require('./routes/doctors');
const doctorsSpecialityRoutes = require('./routes/doctor— specialities');
const keys = require('./config/keys');

const app = express();
mongoose.connect(keys.mongoURI)
    .then(() => console.log('MongoDB Connected'))
    .catch(error => console.log(error));
app.use(passport.initialize());
require('./middleware/passport')(passport);
app.use(require('morgan')('dev'));
app.use('/uploads', express.static('uploads'));
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());
app.use(require('cors')());
app.use('/api/auth', authRoutes);
app.use('/api/medicalinstitution', medicalInstitutionRoutes);
app.use('/api/order', orderRoutes);
app.use('/api/doctors', doctorsRoutes);
app.use('/api/doctorsspeciality', doctorsSpecialityRoutes);
module.exports = app;
```

## ДОДАТОК Е

Вигляд головної сторінки сайту

```
<div class="content w-75 m-auto position-relative p35">
  <ngb-carousel *ngIf="images">
    <ng-template ngbSlide>
      <div class="picsum-img-wrapper">
        <img [src]="images[0]" alt="Random first slide">
      </div> <div class="carousel-caption"> </div>
    </ng-template>
    <ng-template ngbSlide>
      <div class="picsum-img-wrapper">
        <img [src]="images[1]" alt="Random second slide">
      </div><div class="carousel-caption"></div>
    </ng-template>
    <ng-template ngbSlide>
      <div class="picsum-img-wrapper">
        <img [src]="images[2]" alt="Random third slide">
      </div> <div class="carousel-caption"> </div>
    </ng-template>
    <ng-template ngbSlide>
      <div class="picsum-img-wrapper">
        <img [src]="images[3]" alt="Random fourth slide">
      </div>
      <div class="carousel-caption">
      </div>
    </ng-template>
  </ngb-carousel>
</div>
```

## ДОДАТОК Ж

Сервіс для повідомлень

```
import { Injectable } from "@angular/core";
import { MatSnackBar } from "@angular/material/snack-bar";
import { SnackBarTypes } from "./models/snackBar.types";
import { SnackBarEnums } from "./models/snackBar.enums";
@Injectable({   providedIn: 'root'})
export class SnackBarService {
  constructor(
    private _snackBar: MatSnackBar  ) {}
  private open(type: SnackBarTypes = 'success', message: string): void {
    this._snackBar.open(message, 'Закрити', {
      horizontalPosition: 'right',
      verticalPosition: 'top',
      duration: 5000,
      panelClass: SnackBarEnums[type] });
  }
  public success(message: string = 'Виконано успішно'): void {
    this.open('success', message);
  }
  public warning(message: string = 'Перевірте будь ласка дані'): void {
    this.open('warning', message);
  }

  public fail(message: string = 'Помилка завантаження даних'): void {
    this.open('fail', message);
  }
}
```

63

**ДОДАТОК И**
ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА
НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: _Медичний інтернет-сервіс для обслуговування та реєстрації пацієнтів

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

**Показники звіту подібності Unicheck**

Оригінальність 86,6% Схожість 13,4%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Ткачук В.М.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Колесник І.С.
(підпис) (прізвище, ініціали)