

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Розробка веб-орієнтованої системи для стоматологічної клініки»

Виконала студентка групи ЗІПЗ-20-1
ДАНИЛКО Софія Сергіївна

Керівник роботи:
ЧЕПЮК Ларіна Олексіївна

Консультант:
ЛОКТИКОВА Тамара Миколаївна

Рецензент:
ЧИЖМОТРЯ Олена Геннадіївна

Житомир – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«26» лютого 2024 р.

ЗАВДАННЯ
на випускню кваліфікаційну роботу

Здобувач вищої освіти: **ДАНИЛКО Софія Сергіївна**

Керівник роботи: **ЧЕПЮК Ларіна Олексіївна**

Тема роботи: **«Розробка веб-орієнтованої системи для стоматологічної клініки»,**

затверджена Наказом закладу вищої освіти від **«23» лютого 2024 р. №75с**

Вихідні дані для роботи: **об'єктом дослідження є інформаційні технології для розробки веб-орієнтованої системи для стоматологічної клініки, а предметом дослідження - процес взаємодії пацієнтів зі стоматологічною клінікою через веб-сайт.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Локтікова Т.М.	04.03.2024р.	04.03.2024р.
2	Локтікова Т.М.	01.04.2024р.	01.04.2024р.
3	Локтікова Т.М.	13.05.2024р.	13.05.2024р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з веб-орієнтованої системи для стоматологічної клініки і пояснювальної записки, яка містить 66 сторінок, 38 ілюстрацій та 12 таблиць.

Метою роботи є розробка веб-орієнтованої системи для стоматологічної клініки, яка забезпечить зручність для пацієнтів шляхом онлайн-запису на прийом у будь-який зручний для них час, незалежно від робочих годин клініки.

У роботі визначено основні завдання автоматизації у стоматологічній клініці, проаналізовано інформаційні технології, що використовуються для створення веб-сайтів, та обґрунтовано вибір інструментальних засобів. Представлено детальний аналіз вимог до системи, опис функціональних можливостей сайту та структури бази даних.

Було обрано клієнт-серверну архітектуру, що дозволяє створити гнучку, масштабовану та функціональну систему для потреб стоматологічної клініки. Мова програмування Kotlin використана як для фронтенду, так і для бекенду веб-сайту, оскільки вона є швидкою та ефективною, забезпечуючи оптимальну продуктивність. Фронтенд розроблено на фреймворку Kobweb, спеціально створеному для роботи з Kotlin, тоді як для бекенду використано Kobweb API, що базується на потужному фреймворку Ktor для веб-сервісів. Розроблено та протестовано інтерфейс користувача, систему управління даними пацієнтів, а також налаштування безпеки і конфіденційності даних.

Також наведено результати тестування функціональності веб-сайту та його відповідності встановленим вимогам.

КЛЮЧОВІ СЛОВА: АВТОМАТИЗАЦІЯ, БЕЗПЕКА, САЙТ, САМОЗАПИС, СТОМАТОЛОГІЧНА КЛІНІКА

					ІПЗ.КР.Б-121-24-ІПЗ			
		№ докум.						
Розроб.	Данилко С.С.				Розробка веб-орієнтованої системи для стоматологічної клініки. Пояснювальна записка	Літ.	Арк.	Аркушів
Керівник	Чепюк Л.О.						3	66
Рецензент.	Чижмотря О.Г.							
Зав. каф.	Вакалюк Т.А.					Житомирська політехніка, група ЗІПЗ-20-1		

ABSTRACT

The bachelor's degree thesis consists of a web-oriented system for a dental clinic and an explanatory note containing 66 pages, 38 illustrations, and 12 tables.

The aim of the work is to develop a web-oriented system for a dental clinic that will provide convenience for patients by allowing them to schedule appointments online at any time convenient for them, regardless of the clinic's working hours.

The work defines the main tasks of automation in a dental clinic, analyzes the information technologies used for website creation, and justifies the choice of tools. It provides a detailed analysis of the system requirements, describes the functional capabilities of the website, and the structure of the database.

A client-server architecture was chosen to create a flexible, scalable, and functional system for the dental clinic. Kotlin programming language was used for both frontend and backend to ensure optimal performance. The frontend was developed on the Kobweb framework tailored for Kotlin, while the backend utilized Kobweb API based on the powerful Ktor framework for web services. User interface, patient data management system, and data security and confidentiality settings were developed and tested.

The testing results of the website's functionality and its compliance with the specified requirements are also presented.

KEYWORDS: AUTOMATION, SECURITY, WEBSITE, SELF-APPOINTMENT, DENTAL CLINIC.

					III3.KP.Б-121-24-П3	
						4

ЗМІСТ

РЕФЕРАТ.....	3
ABSTRACT.....	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ СИСТЕМИ СТОМАТОЛОГІЧНОЇ КЛІНІКИ	9
1.1 Постановка задачі	9
1.2 Аналіз аналогів програмного продукту	11
1.3 Вибір архітектури веб-орієнтованої системи.....	16
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	21
Висновки до першого розділу	25
РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ДЛЯ СТОМАТОЛОГІЧНОЇ КЛІНІКИ.....	26
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи	26
2.2 Розробка бази даних систем.....	33
2.3 Проектування та реалізація алгоритмів роботи системи	36
2.4 Реалізація функціоналу	38
Висновки до другого розділу.....	41
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ	42
3.1 Порядок встановлення та налаштування параметрів сайту	42
3.2 Структура інтерфейсу веб-додатку та порядок роботи	44
3.3 Тестування роботи веб-додатку	53
Висновки до третього розділу	57
ВИСНОВКИ	58
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	60
ДОДАТКИ.....	62

					ІПЗ.КР.Б-121-24-ПЗ	
						5

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AS – Автономна система

TCP – Transmission Control Protocol

IP – Internet Protocol

VPN - Virtual Private Network

WAN – Wide Area Network

LAN – Local Area Network

					ІІЗ.КР.Б-121-24-ІІЗ	
						6

ВСТУП

Актуальність теми полягає у тому, що у сучасному світі інтернет відіграє важливу роль у житті людей, і медичні послуги не є виключенням. Створення веб-сайту для стоматологічної клініки з можливістю самозапису на прийом відповідає зростаючим потребам пацієнтів у зручності та оперативності. Інтернет-запис значно спрощує процес отримання медичних послуг, дозволяючи пацієнтам економити час і уникати непотрібних телефонних дзвінків або особистих візитів до клініки для запису. Такий підхід також підвищує рівень обслуговування та задоволеності пацієнтів, що є важливим для підтримки позитивної репутації клініки.

Мета випускної кваліфікаційної роботи бакалавра є створення веб-сайту для стоматологічної клініки, забезпечення зручності для пацієнтів, дозволити пацієнтам записуватися на прийом онлайн у будь-який зручний для них час, не залежно від робочих годин клініки.

Для реалізації поставленої мети потрібно виконати наступні **завдання**:

1. Провести огляд функціональності веб-сайтів для медичних установ, методів організації онлайн-запису та забезпечення безпеки та конфіденційності даних пацієнтів.

2. Здійснити вибір інструментів проектування, включаючи мову програмування Kotlin, для розробки фронтенду та бекенду веб-сайту.

3. Розробити та впровадити веб-сайт для стоматологічної клініки, забезпечивши зручність використання, надійність та безпеку даних пацієнтів

Об'єктом дослідження в даному випадку є інформаційні технології для розробки веб-орієнтованої системи для стоматологічної клініки.

Предметом дослідження є процес взаємодії пацієнтів зі стоматологічною клінікою через веб-сайт. Це включає в себе всі аспекти, пов'язані з пошуком інформації про клініку, записом на прийом, управлінням записами та іншими можливостями, які надає веб-сайт. Також є ефективність, зручність і функціональність веб-сайту стоматологічної клініки, спрямованого на

					ІПЗ.КР.Б-121-24-ПЗ	
						7

покращення досвіду пацієнтів при взаємодії з клінікою.

Дослідження також охоплює аналіз ролі сучасних інформаційних технологій у забезпеченні зручності та ефективності медичних послуг, можливості інтеграції веб-сайтів з внутрішніми системами клінік та забезпечення безпеки даних пацієнтів.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ СИСТЕМИ СТОМАТОЛОГІЧНОЇ КЛІНІКИ

1.1 Постановка задачі

Метою кваліфікаційної роботи бакалавра є створення сайту для стоматологічної клініки з самозаписом до лікаря. Розробка фронтенду та бекенду, реалізація на Kotlin, сучасній мові програмування.

Впровадження веб-сайту з функцією самозапису для стоматологічної клініки може значно покращити ефективність роботи і збільшити прибуток, одночасно забезпечуючи високий рівень обслуговування та безпеку пацієнтів.

Визначивши основні завдання автоматизації у стоматологічній клініці, розглянемо автоматизацію обліку пацієнтів:

1. При зборі та зберіганні персональних даних веб-сайт повинен збирати та зберігати персональні дані пацієнтів, включаючи ім'я, контактну інформацію (номер телефону, електронна пошта), адресу, дату народження. Ці дані будуть зберігатися в безпечному середовищі, відповідно до законодавчих вимог про захист персональних даних.

2. Під час реєстрації нових пацієнтів веб-сайт повинен надавати можливість реєстрації нових пацієнтів онлайн. процес реєстрації повинен бути простим і зручним для користувачів. Він повинен включати певну форму для заповнення.

3. В онлайн-графіку прийомів веб-сайт повинен мати інтерактивний графік прийомів, що відображає доступність стоматологів на кожен день. Пацієнти можуть переглядати графік, вибирати зручний час та стоматолога для прийому.

4. Під час онлайн-бронюванню пацієнти можуть забронювати прийом онлайн через веб-сайт. Система перевіряє доступність стоматолога на

					ІПЗ.КР.Б-121-24-ПЗ	
						9

обраний час і автоматично створює запис про прийом.

5. Під час підтвердження бронювання після бронювання прийому пацієнт отримує лист-підтвердження на свою електронну пошту з деталями прийому. Цей лист містить інформацію про дату, час, стоматолога та тип прийому.

6. Під час нагадування про прийом веб-сайт має функцію нагадування про прийом за допомогою sms, електронної пошти або push-сповіщень пацієнти можуть вибрати зручний спосіб отримання нагадування.

7. Під час скасування прийому пацієнти можуть скасувати свій прийом онлайн через веб-сайт. Система надішле повідомлення стоматологу про скасування прийому і отримає підтвердження від нього.

8. У графіку роботи адміністратор клініки може створювати і редагувати графіки роботи стоматологів і іншого персоналу. Система має можливість встановлювати зміни і надавати інформацію про вільні та зайняті часові проміжки для кожного працівника.

9. Під час покращення якості обслуговування пацієнтів веб-сайт має забезпечити швидкий і зручний доступ до інформації про послуги, стоматологів, графік прийомів та іншу релевантну інформацію. Це дозволить пацієнтам самостійно записатися на прийом, отримати необхідні відомості про лікування і зменшити час очікування на прийом.

10. Під час збільшення ефективності роботи автоматизація процесів бронювання прийомів скоротить час на рутинні операції і звільнить персонал для виконання більш важливих завдань. Це дозволить збільшити продуктивність праці та скоротити витрати на адміністративні процеси.

11. При підвищенні рівня конкурентоспроможності веб-сайт повинен стати ключовим інструментом маркетингової стратегії клініки. Забезпечення доступності онлайн-записів, інформації про послуги, відгуків пацієнтів та інших релевантних даних дозволить приваблювати нових пацієнтів і збільшувати рентабельність бізнесу.

12. Під час підвищення рівня безпеки та конфіденційності даних веб-сайт має бути розроблений з врахуванням вимог безпеки і конфіденційності даних. Система повинна захищати медичну інформацію пацієнтів від несанкціонованого доступу і забезпечувати її збереження.

13. При зменшенні часу обслуговування пацієнтів, пацієнти можуть самостійно записатися на прийом через веб-сайт, що значно скоротить час на очікування в черзі і зменшить часову затратність на обслуговування.

14. Збільшення кількості пацієнтів доступність онлайн-записів та зручний інтерфейс веб-сайту дозволить приваблювати нових пацієнтів, які цінують зручність і ефективність обслуговування.

15. При збільшенні рівня задоволеності пацієнтів відбувається покращення якості обслуговування завдяки зручному доступу до інформації, швидкому бронюванню прийомів і прозорості фінансових операцій призведе до збільшення рівня задоволеності пацієнтів.

16. Під час скорочення витрат на адміністративні процеси автоматизація процесів обліку, планування та управління фінансами значно скоротить кількість паперових документів і звільнить персонал від рутинних завдань, що призведе до економії часу і коштів.

17. Під час збільшення прибутковості залучення нових пацієнтів, збільшення ефективності роботи і скорочення витрат дозволять збільшити прибуток стоматологічної клініки.

18. При підвищенні рівня безпеки та конфіденційності даних веб-сайт повинен забезпечувати безпеку і конфіденційність медичної інформації пацієнтів, що дозволить зменшити ризик несанкціонованого доступу і втрати даних.

1.2 Аналіз аналогів програмного продукту

У сучасному світі цифровізація є невід'ємною частиною будь-якого бізнесу, включаючи медичну сферу. Сайти для стоматологічних клінік із функцією самозапису забезпечують зручність для пацієнтів, а також

					ІПЗ.КР.Б-121-24-ПЗ	
						11

підвищують ефективність роботи клініки. Цей розділ присвячений аналізу аналогів програмного продукту, розробленого на мові програмування Kotlin, для створення сайту стоматологічної клініки з можливістю самозапису.

Розглянемо аналоги веб-сайтів таких як HB Clinic (рис. 1.1), Familia (рис. 1.3), та KES Clinic (рис. 1.5).

HB Clinic [21] - це сайт медичного центру, який надає різноманітні послуги, включаючи стоматологічні.

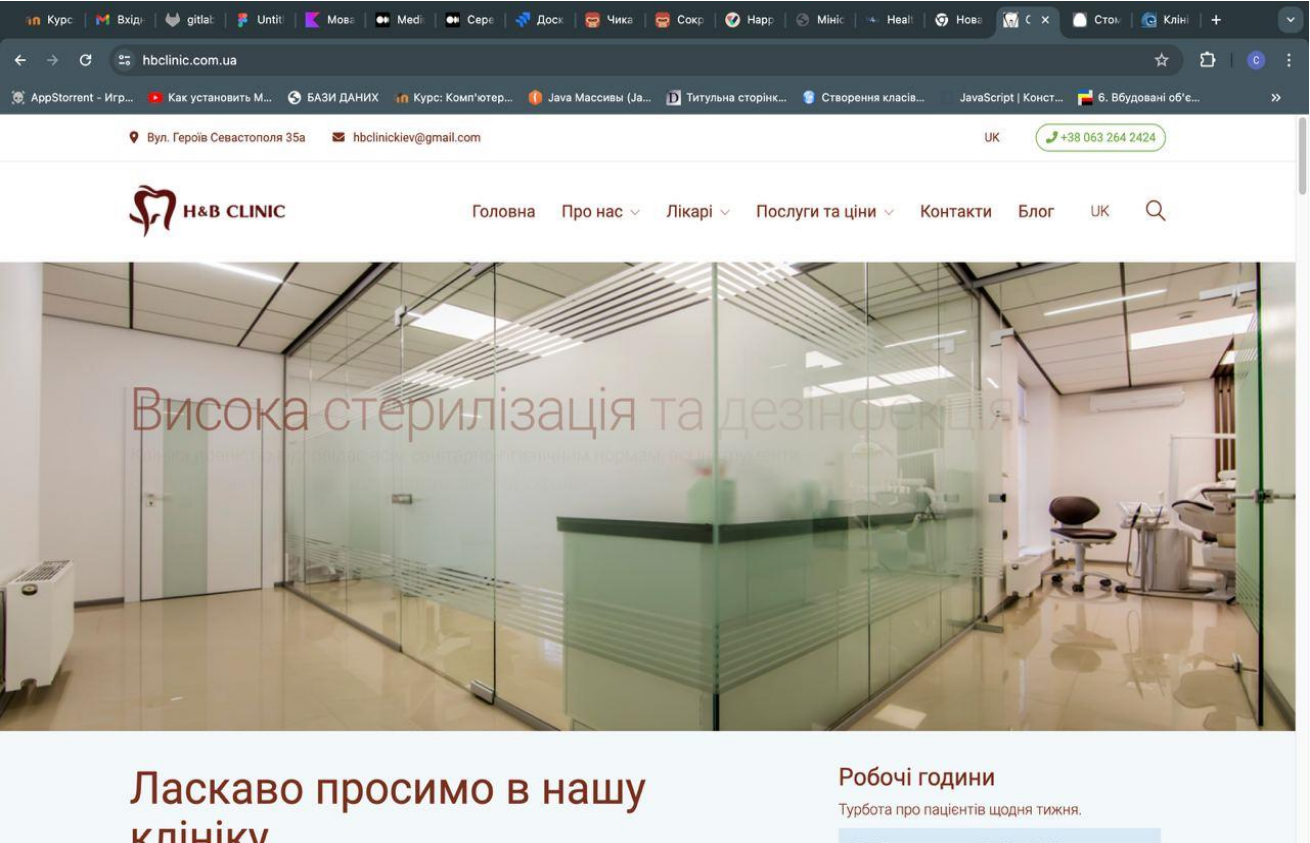


Рисунок 1.1 – Головна сторінка сайту HB Clinic

Сайт дозволяє пацієнтам записуватися на прийом онлайн (рис. 1.2), обирати зручний час та лікаря, а також переглядати доступні послуги.

Функціонал:

- онлайн-запис на прийом;
- перегляд доступних послуг;
- інформація про лікарів та спеціалістів;
- контактна інформація та карта розташування клініки.

Переваги:

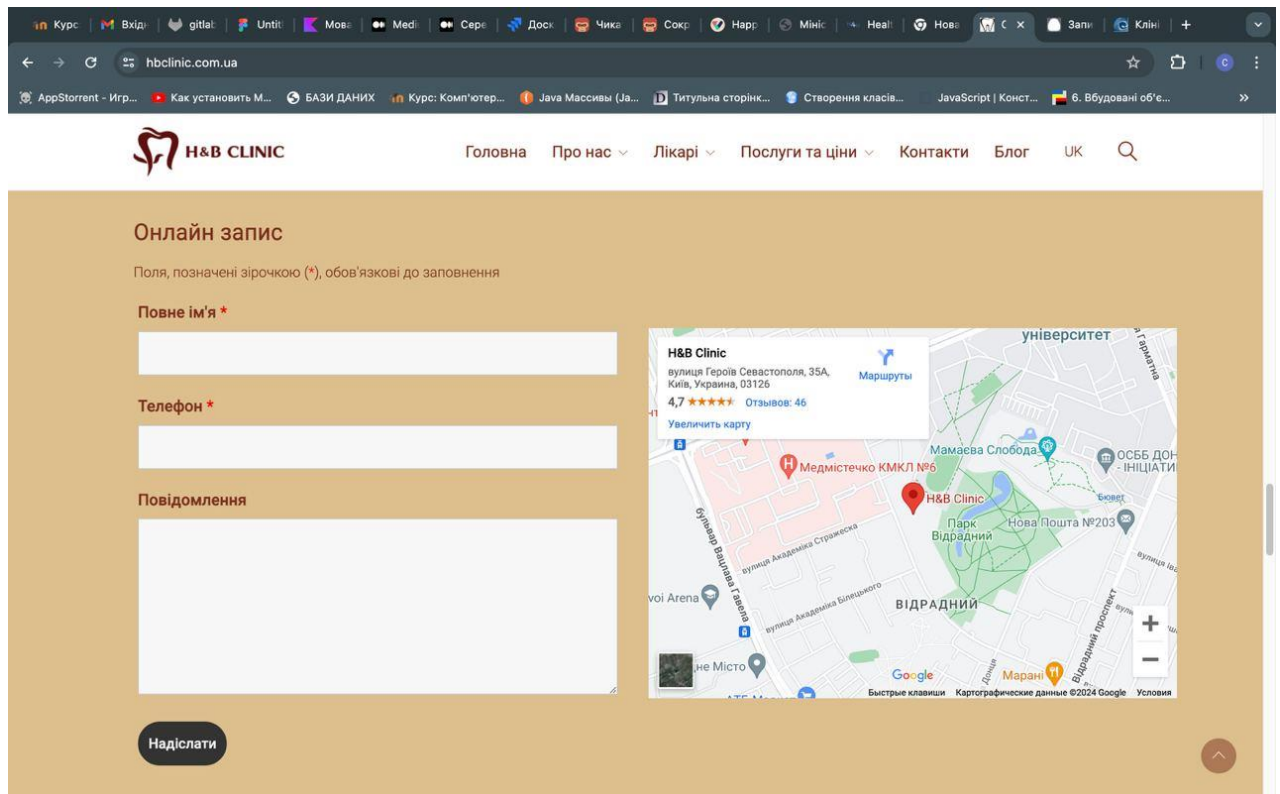


Рисунок 1.2 – Сторінка з самозаписом

- зручний інтерфейс для користувачів;
- можливість швидкого запису на прийом;
- докладна інформація про лікарів.

Недоліки:

- відсутність мобільного додатку;
- обмежені можливості інтеграції з іншими системами.

Familia [22] - це сайт мережі медичних центрів, який пропонує широкий спектр медичних послуг, включаючи стоматологічні. Сайт дозволяє пацієнтам записуватися на прийом онлайн (рис. 1.4), отримувати консультації та переглядати інформацію про клініки та лікарів.

Функціонал:

- онлайн-запис на прийом;
- консультації онлайн;
- перегляд інформації про клініки та лікарів;
- блог з корисними статтями.

Переваги:

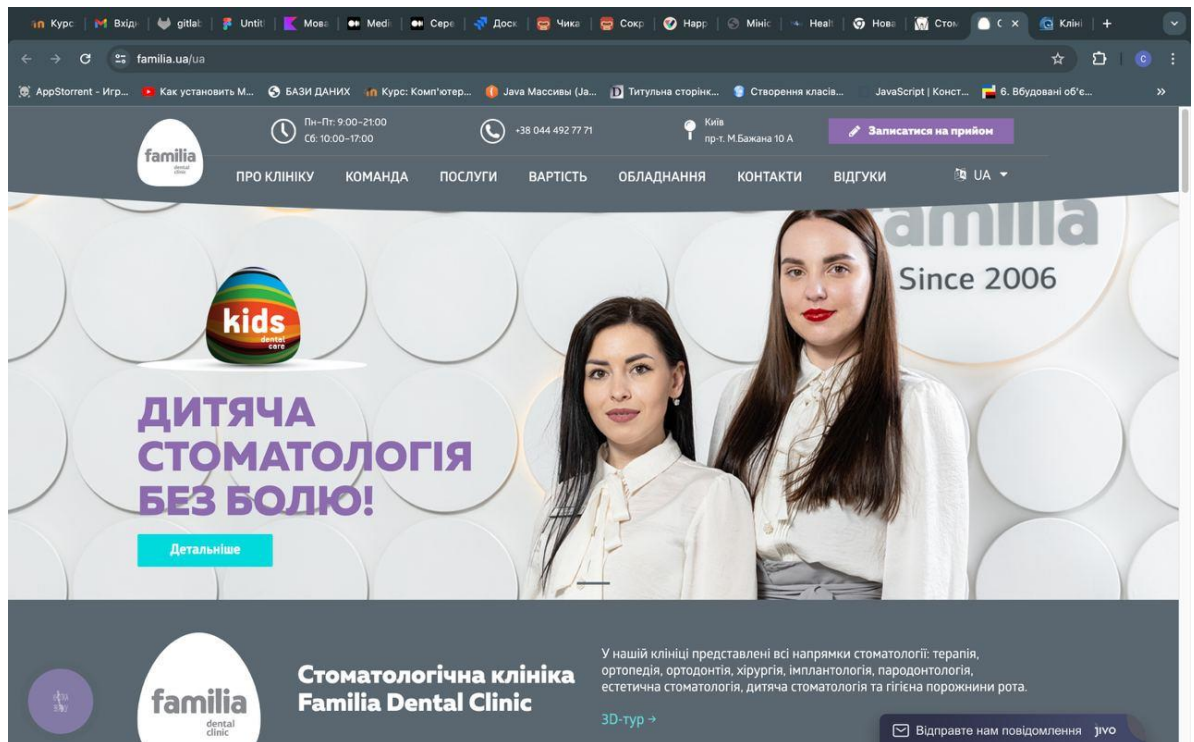


Рисунок 1.3 – Головна сторінка сайту Familia

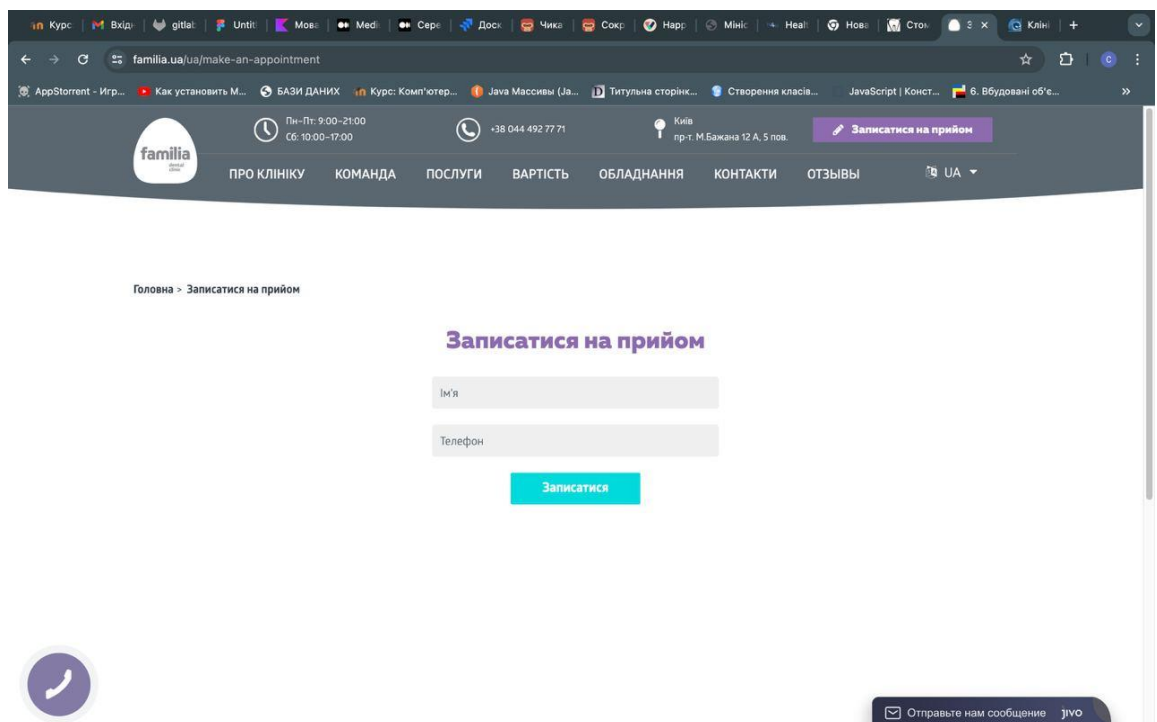


Рисунок 1.4 – Сторінка з самозаписом

- широкий спектр послуг;
- можливість отримання онлайн-консультацій;
- регулярне оновлення контенту.

Недоліки:

- інтерфейс може бути складним для нових користувачів;
- обмежена кількість лікарів у деяких регіонах.

KES Clinic [23] - це сайт стоматологічної клініки (рис. 1.5), яка спеціалізується на комплексних стоматологічних послугах. Сайт надає можливість пацієнтам записуватися на прийом онлайн (рис. 1.6), переглядати інформацію про послуги та лікарів, а також дізнаватися про акції та спеціальні пропозиції.

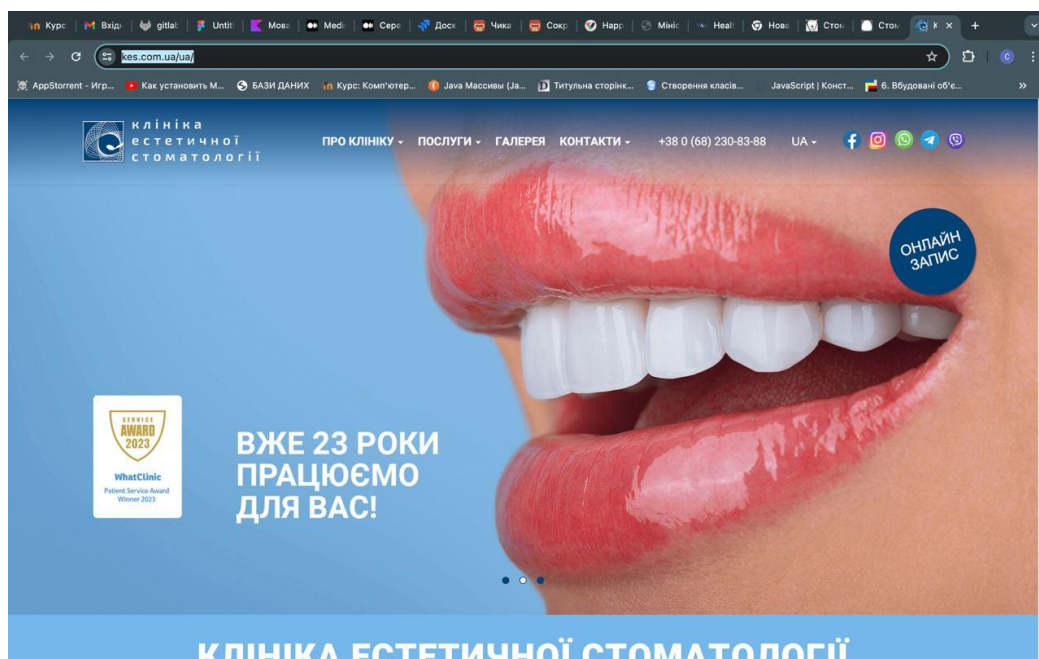


Рисунок 1.5 – Головна сторінка сайту KES Clinic

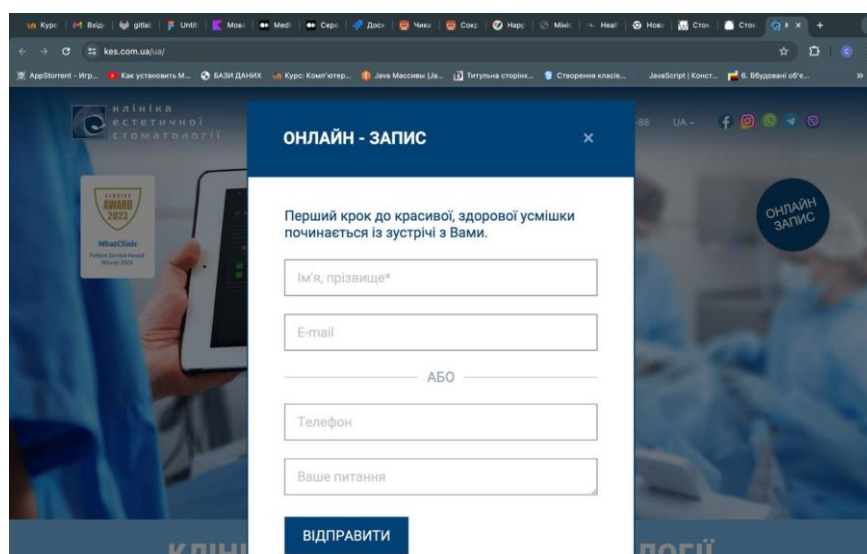


Рисунок 1.6 – Сторінка з самозаписом

Функціонал:

- онлайн-запис на прийом;
- перегляд інформації про послуги та лікарів;
- інформація про акції та спеціальні пропозиції;
- контактна інформація та карта розташування клініки.

Переваги:

- спеціалізація на стоматологічних послугах;
- зручний та інтуїтивний інтерфейс;
- інформація про акції та спеціальні пропозиції.

Недоліки:

- відсутність мобільного додатку;
- обмежені можливості інтеграції з іншими системами.

Проаналізувавши аналоги програмних продуктів для онлайн-запису до стоматологів, можна зробити висновок, що кожен з них має свої сильні та слабкі сторони. Вибір найбільш відповідного рішення залежить від конкретних потреб клініки. Розробка власного сайту на мові програмування Kotlin може забезпечити високий рівень налаштовуваності та інтеграції з існуючими системами клініки, що дозволить оптимізувати процеси управління та підвищити задоволеність пацієнтів.

1.3 Вибір архітектури веб-орієнтованої системи

Розробка будь-якого програмного забезпечення, особливо такого комплексного, як веб-орієнтована система для стоматологічної клініки, починається з вибору оптимальної архітектури. Архітектура визначає структуру системи, взаємодію її компонентів та закладає фундамент для подальшої розробки, масштабування та підтримки. Існує багато підходів до побудови архітектури веб-додатків, кожен з яких має свої переваги та недоліки залежно від специфіки проекту. Було розглянуто деякі з них:

1. Монолітна архітектура.
2. Багаторівнева архітектура (n-tier architecture).

					ІПЗ.КР.Б-121-24-ПЗ	
						16

3. Мікросервісна архітектура.

4. Клієнт-серверна архітектура.

При використанні монолітної архітектури вся логіка додатку зосереджена в межах одного модуля, який одночасно відповідає за роботу з інтерфейсом, бізнес-логікою та даними. Перевагами такого підходу є простота розгортання та розробки на початкових етапах. Але такі недоліки як низька гнучкість та масштабованість, збільшення кодової бази, що ускладнює розробку та впровадження змін та висока зв'язність модулів яка знижує відмовостійкість нівелює вище зазначені переваги

Стосовно багаторівневої архітектури (n-tier architecture) можна сказати, що основою такого підходу є розділення додатку на логічні рівні (презентаційний, бізнес-логіки, доступу до даних), які можуть бути розгорнуті на різних серверах. До основних переваг тут можливо віднести підвищену гнучкість та масштабованість, покращену безпеку за рахунок розділення відповідальності. Але недоліки при цьому є також суттєвими - підвищена складність розробки та розгортання, потреба в більш досвідченій команді розробників.

Наступною цікавою але не менш складною у реалізації є мікросервісна архітектура. Тут додаток розбивається на незалежні модулі (мікросервіси), кожен з яких відповідає за окрему бізнес-функцію та має власну базу даних. Суттєвими перевагами при цьому є максимальна гнучкість та масштабованість, можливість використання різних технологій для різних мікросервісів. Основні недоліки - висока складність розробки, розгортання та підтримки, потреба в досвідченій команді DevOps.

Наступним було розглянуто клієнт-серверну архітектуру. При цьому підході додаток розділяється на дві частини: клієнтську (фронтенд) та серверну (бекенд), які взаємодіють за допомогою мережі. Тут присутні наступні переваги: чіткий розподіл відповідальності між клієнтом та сервером, висока гнучкість у виборі технологій для фронтенду та бекенду, добра масштабованість, оскільки сервер може обробляти запити від багатьох клієнтів одночасно. До основних недоліків відносяться можливі затримки при передачі

					ІПЗ.КР.Б-121-24-ПЗ	
						17

даних між клієнтом та сервером, підвищена складність забезпечення безпеки даних, що передаються по мережі.

Враховуючи вищезазначене, для розробки веб-орієнтованої системи для стоматологічної клініки було обрано клієнт-серверну архітектуру. Цей вибір зумовлений такими основними факторами як оптимальне співвідношення складності та функціональності, чіткий розподіл відповідальності, використання сучасних технологій. Перший фактор полягає в тому, що клієнт-серверна архітектура забезпечує необхідну гнучкість та масштабованість, не ускладнюючи при цьому розробку та розгортання системи. Другий фактор - фронтенд відповідає за відображення інтерфейсу та взаємодію з користувачем, а бекенд - за обробку даних та бізнес-логіку. І на останок, використання Kotlin та Kobweb дозволяє створити швидкий та зручний інтерфейс користувача, тоді як MongoDB забезпечує ефективне зберігання та обробку даних.

Таким чином, клієнт-серверна архітектура є оптимальним рішенням для даного проекту, оскільки дозволяє створити гнучку, масштабовану та функціональну систему, яка задовольнятиме потреби стоматологічної клініки.

Серверна частина (бекенд) є ключовим елементом веб-орієнтованої системи, оскільки вона відповідає за обробку даних, реалізацію бізнес-логіки та взаємодію з базою даних. Вибір оптимальної архітектури бекенду є критично важливим для забезпечення продуктивності, масштабованості та надійності всієї системи.

Розглянемо декілька популярних архітектурних підходів для побудови бекенду:

1. Модель "Модель-Вид-Контролер" (MVC) - це класичний підхід, який розділяє бекенд на три взаємопов'язані частини: модель, що описує структуру даних та бізнес-логіку, вид який відповідає за представлення даних користувачу та контролер, що обробляє запити від користувача та оновлює модель та вид. Основними переваги - чіткий розподіл відповідальності між компонентами, полегшене тестування та підтримка коду. До основних недоліків відноситься

можливість бути громіздким для невеликих проектів та збільшення складності при роботі з динамічними веб-інтерфейсами.

2. RESTful API з мікросервісами, тут серверна частина складається з набору незалежних мікросервісів, кожен з яких відповідає за певну функцію та взаємодіє з іншими сервісами за допомогою REST API. Цей підхід зумовлює такі переваги як висока гнучкість та масштабованість, можливість використання різних мов програмування та технологій для різних сервісів. Але також є суттєві недоліки: підвищена складність розробки, розгортання та підтримки, а також потреба в досвідченій команді розробників.

3. GraphQL API - даний підхід використовує єдину кінцеву точку для всіх запитів до даних, що дозволяє клієнту отримувати тільки необхідну інформацію. Перевагами тут є зменшення кількості запитів до сервера та підвищена гнучкість для клієнтських додатків. Також він має суттєві недоліки: більш складна реалізація на стороні сервера, можливі проблеми з продуктивністю при складних запитах.

4. Serverless архітектура - це набираючий популярність підхід пов'язаний з використанням хмарних сервісів для виконання коду без необхідності керування серверами. Це зумовлює наступні переваги - автоматичне масштабування та висока доступність, оплата тільки за фактично використані ресурси. При цьому також є і недоліки: залежність від хмарного провайдера, обмеження на час виконання коду.

Для даного проекту обрано архітектуру RESTful API, побудовану з урахуванням принципів MVC. На мою думку, це оптимальне рішення з огляду на наступні фактори:

- баланс між простотою та функціональністю - MVC забезпечує чітку структуру проекту та полегшує його підтримку, тоді як REST API надає гнучкість у взаємодії з клієнтською частиною;
- ефективність розробки - Kotlin та фреймворк Ktor дозволяють швидко створювати ефективні та масштабовані REST API;

- інтеграція з MongoDB - Ktor має чудову підтримку для роботи з MongoDB, що спрощує обробку даних. Обрана архітектура дозволила створити гнучкий та масштабований бекенд, який відповідає всім вимогам веб-орієнтованої системи для стоматологічної клініки. Вона забезпечує ефективну взаємодію з клієнтською частиною, обробку даних та реалізацію бізнес-логіки.

Було визначено структуру клієнтської частини, клієнтська частина (фронтенд) веб-орієнтованої системи є візуальним інтерфейсом, з яким безпосередньо взаємодіють користувачі. Від вибору архітектури фронтенду залежить зручність використання, швидкість роботи та загальне враження від системи.

Розглянемо декілька поширених підходів до побудови фронтенду:

1. Серверний рендеринг (Server-Side Rendering, SSR). HTML-сторінки формуються на стороні сервера при кожному запиті від клієнта. Тут наявні наступні переваги: швидке початкове завантаження сторінки, краща SEO-оптимізація, оскільки пошукові системи легше індексують HTML-контент. Цей підхід не позбавлений недоліків, до яких відносяться збільшене навантаження на сервер, менша інтерактивність та динамічність інтерфейсу.

2. Клієнтський рендеринг (Client-Side Rendering, CSR) з використанням JavaScript-фреймворків, на стороні клієнта динамічно оновлює інтерфейс, отримуючи дані від сервера через API. Це зумовлює наступні переваги як висока інтерактивність та динамічність інтерфейсу, можливість створення односторінкових додатків (SPA - Single Page Application) з плавними переходами. До недоліків відноситься повільніше початкове завантаження сторінки, та потребу в оптимізації для SEO.

3. Гібридний підхід (SSR + CSR) поєднує переваги обох підходів як початковий рендеринг на сервері для швидкого завантаження та динамічне оновлення на клієнті для інтерактивності. Це надає йому переваги, оптимальний баланс між швидкістю, SEO та інтерактивністю. До основного недоліку віднесемо підвищену складність розробки.

4. Компонентно-орієнтована архітектура, інтерфейс будується з невеликих, незалежних та багаторазових компонентів. Основними перевагами тут є покращена модульність та повторне використання коду, спрощення розробки та підтримки складних інтерфейсів.

Для даного проекту обрано компонентно-орієнтовану архітектуру з використанням фреймворку Kobweb, з урахуванням наступних факторів: ефективність та простота розробки, сучасний підхід, нескладну інтеграцію з бекендом.

Ефективність та простота розробки зумовлена використанням Kobweb, написаним на Kotlin, що дозволяє створювати веб-інтерфейси декларативно, використовуючи лаконічний синтаксис та потужні інструменти.

Компонентно-орієнтована архітектура є сучасним стандартом розробки фронтенду, що забезпечує гнучкість, масштабованість та простоту підтримки.

Інтеграція з бекендом полягає в тому, що Kobweb має вбудовану підтримку для взаємодії з бекендом через REST API, що спростить обмін даними між клієнтом та сервером.

Обрана архітектура дозволила створити швидкий, інтерактивний та зручний у використанні фронтенд, який відповідає всім вимогам користувачів веб-орієнтованої системи для стоматологічної клініки.

Таким чином, комбінація обраних архітектурних рішень для бекенду та фронтенду, а також використання сучасних технологій, таких як Kotlin, Kobweb та MongoDB, стали запорукою успішної реалізації ефективної та масштабованої веб-орієнтованої системи для стоматологічної клініки.

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Мову програмування Kotlin було обрано для розробки як фронтенду, так і бекенду веб-сайту стоматологічної клініки.

Kotlin - це сучасна, статично типізована мова програмування, яка набирає популярності у веб-розробці. Вона має велику спільноту розробників, що

					ІПЗ.КР.Б-121-24-ПЗ	
						21

забезпечує доступність ресурсів та підтримки.

Kotlin має строгу систему типів, що допомагає виявити помилки на ранніх етапах розробки та покращує загальну безпеку коду. Швидка і ефективна мова, що забезпечує оптимальну продуктивність як для фронтенду, так і для бекенду. Kotlin має лаконічний синтаксис, який зрозумілий і простий у навчанні, що дозволяє швидко розробляти код. Добре інтегрується з іншими популярними технологіями, що використовуються в веб-розробці, такими як Java, Spring Framework, JavaScript, React, Angular, Vue.js.

Для розробки фронтенду використано фреймворк Kobweb. Kotlin-орієнтований, Kobweb розроблений спеціально для створення веб-додатків на Kotlin, що дозволяє використовувати всі переваги мови, включаючи її безпеку, продуктивність і зручність використання. Kobweb використовує SSR, що дозволяє генерувати HTML-сторінки на сервері перед відправкою до клієнта. Це забезпечує швидше завантаження сторінок, покращує SEO-оптимізацію та підвищує доступність для користувачів. Інтегрується з Jetpack Compose, що дозволяє створювати сучасний, інтерактивний і візуально привабливий фронтенд. Kobweb підтримує Material 3 Design System, що забезпечує сучасний і доступний візуальний стиль для веб-сайту. Спрощує процес розробки фронтенду, дозволяючи швидко створювати інтерактивні елементи і компоненти завдяки інтеграції з Jetpack Compose і Material 3 Design System.

Фреймворк Kobweb API (Ktor)

Для розробки бекенду використано Kobweb API, який базується на Ktor, потужному фреймворку для створення веб-сервісів.

Ktor - це легкий і швидкий фреймворк, розроблений для Kotlin. Він дозволяє створювати надійні і масштабовані веб-сервіси з високою продуктивністю.

Ktor підтримує Kotlin Coroutines, що дозволяє використовувати асинхронну обробку для збільшення продуктивності і оптимізації використання ресурсів. Він є дуже гнучким і дозволяє розробляти веб-

сервіси різних типів (REST, WebSocket, і т.д.). Ktor має простий і інтуїтивно зрозумілий API, що спрощує навчання і розробку.

Для зберігання даних використано NoSQL-базу даних MongoDB. MongoDB - це гнучка база даних, яка дозволяє зберігати різні типи даних, включаючи структуровані, напівструктуровані та неструктуровані дані. Також масштабована база даних, яка може обробляти великі обсяги даних, що робить її ідеальною для великих стоматологічних клінік. Забезпечує швидкий доступ до даних, що важливо для оперативного обслуговування пацієнтів.

MongoDB використовує документну модель, що спрощує збереження і обробку даних про пацієнтів, послуги, прийоми та іншу інформацію. Має офіційні драйвери для Kotlin, що спрощує інтеграцію з бекендом веб-сайту.

Розглянемо мінімальні вимоги до серверного та клієнтського обладнання:

- процесор: Intel Core i5 або еквівалентний, з частотою не менше 2.5 ГГц;
- оперативна пам'ять: 8 ГБ;
- жорсткий диск: SSD-накопичувач об'ємом не менше 256 ГБ;
- операційна система: Linux (Debian/Ubuntu) або Windows Server;
- веб-сервер: Apache або Nginx.

Розглянемо мінімальні вимоги до серверного та клієнтського обладнання:

- процесор: Intel Core i7 або еквівалентний, з частотою не менше 3.0 ГГц;
- оперативна пам'ять: 16 ГБ або більше;
- жорсткий диск: SSD-накопичувач об'ємом не менше 512 ГБ;
- операційна система: Linux (Debian/Ubuntu) ;
- веб-сервер: Nginx.

Серверне обладнання має забезпечувати високу продуктивність для обробки запитів від клієнтів, а також для роботи бекенду та бази даних. Сервер повинен мати можливість масштабуватися для обробки зростання кількості користувачів і даних. Серверне обладнання має бути надійним і забезпечувати безперебійну роботу веб-сайту.

Клієнтське обладнання має мінімальні вимоги:

1. Операційна система: Windows, macOS або Linux
2. Веб-браузер: Chrome, Firefox, Safari, Edge
3. Інтернет-з'єднання: 1 Мбіт/с

Веб-сайт повинен правильно працювати на всіх популярних платформах і браузерах. Клієнтське обладнання має забезпечувати швидку швидкість завантаження сторінок і відгуку веб-сайту. Веб-сайт повинен бути доступним для всіх користувачів з різними рівнями швидкості інтернет-з'єднання.

Розглянемо інструменти для тестування та розгортання фронтенду:

1. Jetpack Compose UI Test: Інструмент для тестування інтерфейсу, розроблений Google для Jetpack Compose. Дозволяє автоматизувати тестування фронтенду, перевіряючи функціональність і зручність користування веб-сайту.

2. Kobweb's built-in testing framework: Kobweb надає вбудований інструмент для тестування фронтенду, що дозволяє проводити основні тести без необхідності використовувати зовнішні бібліотеки.

3. Selenium: Популярний інструмент для автоматизованого тестування веб-додатків, що дозволяє симулювати дії користувача і перевіряти функціональність веб-сайту.

Розглянемо інструменти для тестування та розгортання тестування бекенду:

1. Ktor's testing framework: Ktor надає вбудований інструмент для тестування бекенду, що дозволяє проводити unit-тести і інтеграційні тести для перевірки правильної роботи API і взаємодії з базою даних.

2. JUnit 5: Популярний інструмент для тестування в Java, що може бути використаний для тестування бекенду, особливо якщо код бекенду має залежності від Java-бібліотек.

3. Mockito: Популярна бібліотека для мокування в Java, що дозволяє тестувати код бекенду без необхідності взаємодії з зовнішніми системами,

наприклад, з базою даних.

Розглянемо інструменти для тестування бази даних:

1. MongoDB's built-in testing tools: MongoDB надає вбудовані інструменти для тестування бази даних, що дозволяє проводити тестування функціональності і продуктивності бази даних.

2. Robo 3T: Популярний інструмент для управління MongoDB, що дозволяє виконувати запити, аналізувати дані і виконувати тестування бази даних.

Розглянемо інструменти інструменти для розгортання:

1. Docker: Популярна платформа для контейнеризації, що дозволяє розгорнути веб-сайт і всі його залежності в одному контейнері, що полегшує розгортання і забезпечує більшу мобільність.

2. Kubernetes: Платформа для управління контейнерами, що дозволяє розгорнути і масштабувати веб-сайт в хмарному середовищі.

3. AWS, Azure, Google Cloud: Хмарні платформи, що надають різні інструменти і сервіси для розгортання і управління веб-сайтами.

Вибір інструментів розгортання залежить від конкретних потреб веб-сайту, а також від того, чи веб-сайт буде розгорнуто на власних серверах, віртуальних машинах або в хмарі.

Висновки до першого розділу

Проведено детальний аналіз використання інформаційних технологій для створення веб-сайту стоматологічної клініки. Визначено основні задачі, які має вирішувати сайт, включаючи автоматизацію обліку пацієнтів, планування прийомів та управління персоналом. Проведено аналіз аналогів програмного забезпечення для обґрунтування вибору архітектури та інструментальних засобів. Було розглянуто і визначено вимоги до апаратного та програмного забезпечення для створення ефективного та зручного веб-сайту клініки.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ ДЛЯ СТОМАТОЛОГІЧНОЇ КЛІНИКИ

2.1 Визначення варіантів та використання об'єктно-орієнтованої структури системи

Після аналізу поставленої мети для проектування були визначені такі актори як пацієнт, адміністратор клініки, стоматолог, системний адміністратор.

Випадки використання для актора "Пацієнт":

1. Реєстрація.
2. Вхід в систему.
3. Перегляд графіка прийомів.
Бронювання прийому.
4. Перегляд історії лікування.
5. Оплата послуг.
6. Відміна прийому.
7. Отримання рекомендацій.
8. Залишення відгуку.

Випадки використання для актора "Адміністратор клініки":

1. Управління графіком прийомів.
2. Реєстрація пацієнтів.
3. Відстеження доступності стоматологів.
4. Відправлення сповіщень пацієнтам.
5. Управління платежами.
6. Аналіз статистики відвідуваності.

Випадки використання для актора "Стоматолог":

1. Перегляд графіка прийомів.
2. Редагування графіка прийомів.
3. Введення даних про лікування пацієнта.
4. Перегляд історії лікування пацієнта.
5. Видача рекомендацій пацієнту.

6. Випадки використання для актора "Системний адміністратор":
7. Налаштування користувачів і прав доступу.
8. Моніторинг безпеки системи.
9. Багаторазове резервування даних.
10. Оновлення програмного забезпечення.

При реалізації даного проекту були прийняті до уваги три перші актори і згідно цього для візуального представлення була створена діаграма варіантів використання (рис. 2.1).

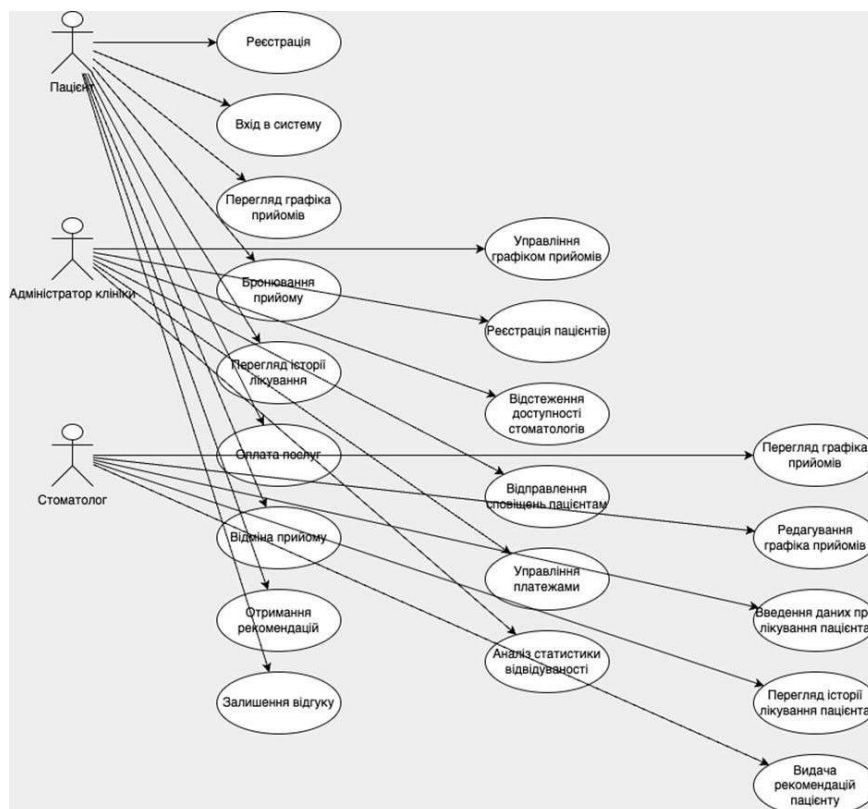


Рисунок 2.1 – Діаграма варіантів використання

Опис діаграми варіантів використання наведено в табл. 2.1 – 2.6.

Таблиця 2.1

Опис прецеденту «Реєстрація пацієнта»

Назва варіанту використання	Реєстрація пацієнта
Основні актори	Пацієнт, адміністратор
Короткий опис	Пацієнт може самостійно зареєструватися в системі або бути зареєстрованим адміністратором клініки.

	В процесі реєстрації збираються необхідні особисті дані пацієнта.
--	---

Таблиця 2.2

Опис прецеденту «Запис на прийом»

Назва варіанту використання	Запис на прийом
Основні актори	Пацієнт, адміністратор
Короткий опис	Пацієнт або адміністратор можуть записати пацієнта на прийом до лікаря, вибравши дату, час та необхідного фахівця.

Таблиця 2.3

Опис прецеденту «Перегляд медичної історії»

Назва варіанту використання	Перегляд медичної історії
Основні актори	Пацієнт, лікар
Короткий опис	Пацієнт може переглядати свою медичну історію, а лікар - історії своїх пацієнтів, включаючи записи про попередні візити, діагнози та проведені лікування.

Таблиця 2.4

Опис прецеденту «Редагування медичних записів»

Назва варіанту використання	Редагування медичних записів
Основні актори	Лікар
Короткий опис	Лікар має можливість редагувати медичні записи пацієнтів, вносячи нову інформацію про діагнози, проведені процедури та рекомендації.

Таблиця 2.5

Опис прецеденту «Оплата послуг»

Назва варіанту використання	Оплата послуг
Основні актори	Пацієнт, адміністратор
Короткий опис	Пацієнт може оплачувати надані медичні

	послуги через систему, а адміністратор може переглядати та підтверджувати проведені оплати.
--	---

Таблиця 2.6

Опис прецеденту «Формування звітності»

Назва варіанту використання	Формування звітності
Основні актори	Адміністратор
Короткий опис	Адміністратор може генерувати різноманітні звіти щодо діяльності клініки, включаючи звіти по пацієнтах, наданих послугах та фінансові звіти.

Основною метою розробки цього проекту є створення комплексної системи для надання широкого спектру послуг клієнтам. Ця система покликана спростити та оптимізувати процеси обслуговування, забезпечуючи швидкий та ефективний доступ до необхідних послуг. Ключовим завданням є розробка зручного та інтуїтивно-зрозумілого інтерфейсу, який дозволить користувачам легко знаходити та замовляти потрібні їм послуги.

Не менш важливою ціллю є підвищення якості надання послуг шляхом впровадження сучасних технологій та автоматизації ключових процесів. Це дозволить забезпечити високий рівень обслуговування, скоротити час очікування та помилки у виконанні замовлень. Крім того, система має надавати можливість детального аналізу даних про надані послуги, що сприятиме прийняттю виважених управлінських рішень та постійному вдосконаленню системи.

Врешті-решт, основна мета проекту полягає у створенні комплексного рішення, яке задовольнить потреби клієнтів, підвищить ефективність операційної діяльності та забезпечить стійкий розвиток компанії на довгострокову перспективу. Досягнення цієї мети вимагає злагодженої роботи команди та впровадження передових технологічних рішень.

Розглянемо різні вимоги користувачів.

Вимоги пацієнтів:

					ІПЗ.КР.Б-121-24-ПЗ	
						29

- реєстрація та авторизація надає можливість самостійної реєстрації в системі з введенням основних персональних даних;
- авторизація в системі за допомогою електронної пошти та пароля;
- запис на прийом надає можливість перегляду доступних дат та часу прийому для запису на прийом до лікаря;
- можливість обирати лікаря за спеціалізацією;
- отримання підтвердження запису на прийом;
- перегляд медичної історії надає можливість перегляду історії відвідувань, діагнозів, призначеного лікування;
- можливість завантаження/перегляду медичних документів (скани, результати обстежень тощо);
- можливість перегляду вартості послуг;
- можливість онлайн-оплати за надані послуги;
- отримання підтвердження про оплату;
- відгуки та оцінка лікарів;
- можливість залишати відгуки та оцінки про роботу лікарів;
- перегляд відгуків і оцінок інших пацієнтів.

Вимоги лікарів:

- управління розкладом прийомів надає можливість переглядати записані на прийом пацієнтів;
- можливість підтверджувати, скасовувати або переносити прийоми;
- ведення медичних записів надає можливість створювати, редагувати та переглядати медичні записи пацієнтів;
- можливість додавати результати обстежень, діагнози, призначене лікування;
- перегляд історії пацієнтів надає можливість переглядати повну медичну історію пацієнтів.

Вимоги адміністраторів:

- управління користувачами надає можливість реєструвати нових користувачів (пацієнтів, лікарів, адміністраторів) ;

					ІПЗ.КР.Б-121-24-ПЗ	
						30

- можливість редагувати профілі користувачів;
- можливість призначати/змінювати ролі користувачів;
- управління послугами та прайсом надає можливість додавати, редагувати та видаляти інформацію про послуги, їхні ціни;
- управління промоакціями надає можливість створювати, редагувати та публікувати інформацію про поточні знижки та акції;
- аналітика та звітність надає можливість перегляду статистики відвідувань, проведених процедур, доходів тощо;
- можливість генерувати звіти за різними параметрами.

Функціональними вимогами є управління користувачами:

1. Реєстрація нових користувачів (пацієнтів, лікарів, адміністраторів), авторизація користувачів, редагування профілів користувачів, присвоєння ролей користувачам (пацієнт, лікар, адміністратор), управління розкладом прийомів, перегляд розкладу прийомів лікарів.
2. Запис пацієнтів на прийом має підтвердження, скасування або перенесення прийомів.
3. Ведення медичних записів містить створення, редагування та перегляд медичних записів пацієнтів, також додавання результатів обстежень, діагнозів, призначеного лікування.
4. Управління медичною інформацією має перегляд повної медичної історії пацієнтів, завантаження та перегляд медичних документів (скани, результати обстежень).
5. Управління оплатою послуг містить перегляд вартості медичних послуг, проведення онлайн-оплати за послуги, отримання підтвердження про оплату.
6. Управління відгуками та оцінками має можливість залишати відгуки та оцінки лікарям і перегляд відгуків та оцінок, залишених іншими пацієнтами.

Розглянемо нефункціональні вимоги:

					ІПЗ.КР.Б-121-24-ПЗ	
						31

1. Продуктивність відповідає за швидкий час відгуку системи при виконанні основних операцій (менше 2 секунд) та можливість обслуговування щонайменше 100 одночасних користувачів.

2. Безпека відповідає за шифрування даних (паролів, медичної інформації) під час передачі та зберігання, багатофакторна автентифікація для адміністраторів та регулярне резервне копіювання даних.

3. Надійність відповідає за відсутність простоїв системи більше 1 години на місяць, швидке відновлення системи після збоїв (менше 30 хвилин).

4. Зручність використання має містити інтуїтивно зрозумілий та адаптивний користувацький інтерфейс, мобільну адаптацію системи для зручного доступу з різних пристроїв.

5. Масштабованість має містити можливість збільшення кількості користувачів, лікарів, медичних послуг без суттєвого впливу на продуктивність

Системні вимоги операційної системи:

- мінімальна версія: Windows 10, macOS 10.15 (Catalina), Ubuntu 18.04;
- підтримка Windows, MacOS, Linux.

Апаратне забезпечення містить такі мінімальні вимоги:

- процесор: Intel Core i5 або еквівалентний AMD;
- оперативна пам'ять: 8 ГБ;
- вільне місце на диску: 50 ГБ.

Рекомендовані вимоги:

- процесор: Intel Core i7 або еквівалентний AMD, оперативна пам'ять: 16 ГБ, вільне місце на диску: 100 ГБ;

- програмне забезпечення має містити MongoDB 4.4 або новішої версії,
- visual Studio Code, Xcode або IntelliJ IDEA;
- мережеві вимоги мають підключення до Інтернету зі швидкістю не менше 10 Мбіт/с, рекомендована швидкість: 50 Мбіт/с або вища;
- безпека містить підтримку SSL/TLS для безпечної передачі даних, реалізацію механізмів автентифікації, авторизації та контролю доступу,

налаштування безпечних конфігурацій для MongoDB, сервера додатків та інших компонентів;

- масштабованість має можливість горизонтального масштабування компонентів системи (наприклад, реплікація MongoDB, кластеризація додатків), підтримку горизонтального масштабування навантаження на базі хмарної інфраструктури (наприклад, Azure, AWS, GCP) ;

- підтримка інтеграції з зовнішніми системами, такими як платіжні шлюзи, системи обліку, CRM тощо;

- резервне копіювання та відновлення містить регулярне резервне копіювання даних MongoDB і можливість відновлення системи з резервної копії.

2.2 Розробка бази даних системи

Для збереження даних (файлів та інформації) в системі використано документоорієнтована модель бази даних. MongoDB зберігає дані у форматі JSON - подібних документів, що дозволяє зберігати різні типи даних в одному документі.

База даних системи складається з наступних таблиць:

- users – таблиця для збереження даних користувачів;
- appointments – таблиця для збереження даних про прийом;
- medicalRecords – таблиця для збереження медичних даних пацієнтів;
- specialty – таблиця опису спеціалістів;
- doctorSpecialty – зв'язуюча таблиця;
- review – таблиця для збереження відгуків пацієнтів;
- inventory – таблиця для збереження необхідного інвентаря;
- promotion – таблиця для збереження промоакцій клініки;
- services – таблиця для збереження послуг клініки;
- payments – таблиця для збереження платежів;
- notifications – таблиця для збереження повідомлень для пацієнтів.

Таблиця 2.7

Опис таблиці Users

Поле	Тип	Призначення
_id	ObjectId	Ідентифікатор
firstName	string	Ім'я
lastName	string	Фамілія
email	string	Email користувача
password	string	Пароль користувача
role	string	Роль (пацієнт, лікар, адміністратор)
createdAt	long	Дата створення
updatedAt	long	Дата останнього оновлення

Таблиця 2.8

Опис таблиці Appointment

Поле	Тип	Призначення
_id	ObjectId	Ідентифікатор
patientId	string	Ідентифікатор пацієнта (посилання на User)
doctorId	string	Ідентифікатор лікаря (посилання на User)
date	string	Дата призначення
time	string	Час призначення
status	string	Статус (заплановано, підтверджено, скасовано)
createdAt	long	Дата створення
updatedAt	long	Дата останнього оновлення

Таблиця 2.9

Опис таблиці MedicalRecord

Поле	Тип	Призначення
_id	ObjectId	Ідентифікатор
patientId	string	Ідентифікатор пацієнта (посилання на User)
doctorId	string	Ідентифікатор лікаря (посилання на User)
diagnosis	string	Діагноз
treatment	string	Уход
notes	string	Нотатки
attachments	string	Додатки

createdAt	long	Дата створення
updatedAt	long	Дата останнього оновлення

Таблиця 2.10

Опис таблиці Specialty

Поле	Тип	Призначення
_id	ObjectId	Ідентифікатор
name	string	Назва
description	string	Опис

Таблиця 2.11

Опис таблиці DoctorSpecialty (зв'язуюча таблиця)

Поле	Тип	Призначення
_id	ObjectId	Ідентифікатор
doctorId	string	Ідентифікатор лікаря (посилання на User)
specialtyId	string	(посилання на Specialty)

Таблиця 2.12

Опис таблиці Review

Поле	Тип	Призначення
_id	ObjectId	Ідентифікатор
patientId	string	Ідентифікатор пацієнта (посилання на User)
doctorId	string	Ідентифікатор лікаря (посилання на User)
rating	string	Рейтинг
comment	string	Коментар
createdAt	string	Дата створення

На основі створених таблиць було сформовано діаграму бази даних (рис.2.2).

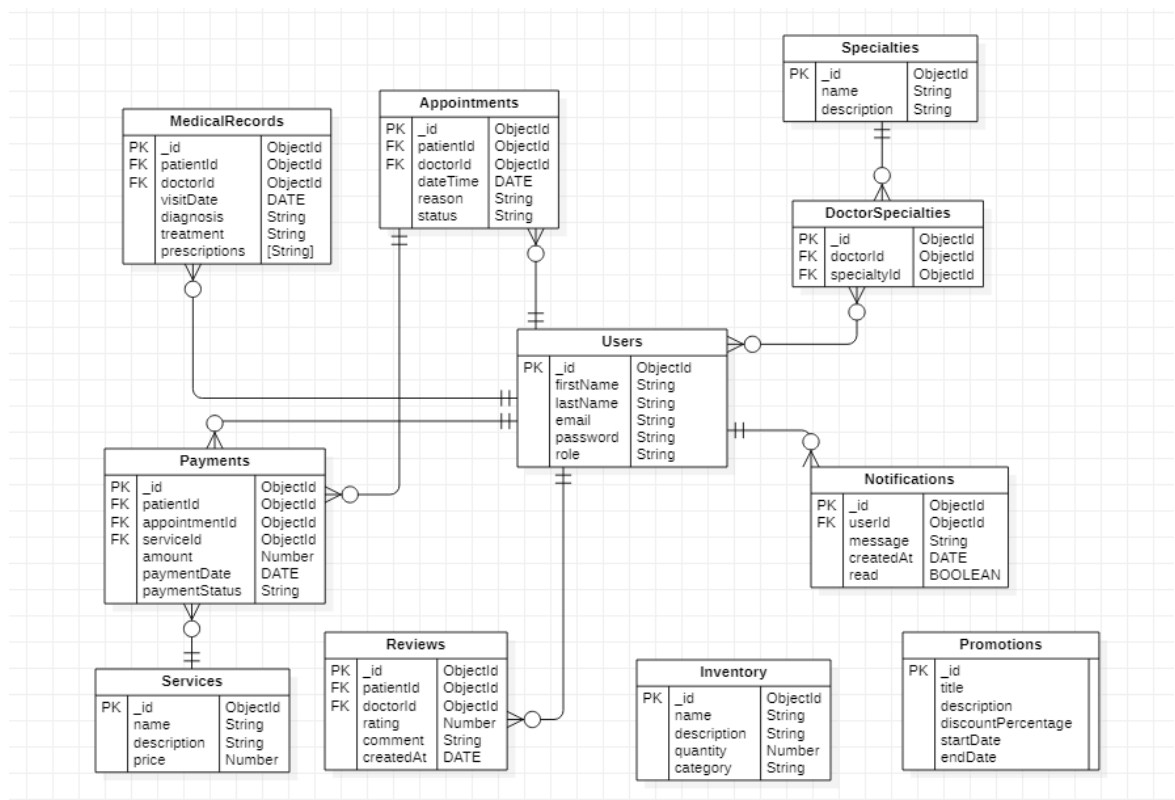


Рисунок 2.2 – Схематичне зображення бази даних

2.3 Проектування та реалізація алгоритмів роботи системи

Система містить всі основні функції для роботи і є простою як у засвоєнні так і у використанні.

Для роботи з системою було створено сутність «Роль», яка надає сценарії використання можливостей системи в залежності від типу ролі. На даному етапі створені такі типи ролей у системі: адміністратор, гість, пацієнт, лікар.

Доки користувач не пройшов авторизацію, він за замовчуванням має статус - гість. При даному сценарії використання для такого типу користувача система має обмежений сценарій використання - перегляд тільки загальної та рекламної інформації.

Сценарій «Адміністратор» має максимальні можливості використання системи у створенні та адмініструванні користувачів, графіків прийомів, тощо, але має обмежений доступ до конфіденційної інформації пацієнтів та співробітників яка зберігається у відповідних електронних кабінетах.

Сценарій «Лікар» передбачає роботу з графіками прийомів пацієнтів, обмеженою службовою інформацією з карти пацієнта, та інформацією, що доступна у відповідному електронному кабінеті лікаря.

Після реєстрації чи авторизації користувача у якості пацієнта йому надається доступ до можливостей системи за сценарієм «Пацієнт».

Трохи детальніше опишемо даний сценарій. При реєстрації користувача у якості пацієнта створюється та надається доступ до електронного кабінету пацієнта в якому зберігається загальна інформація про пацієнта, та з якого надається додатковий доступ до інших можливостей системи таких як самозапис до лікаря, перегляд та участь у промоакціях, перегляд та додавання коментарів пацієнтів тощо. До ідентифікатору даного кабінету прив’язуються інші типи документів таких як електронна карта пацієнта та інші. Система передбачає можливість записатись пацієнтам на прийом. Після авторизації користувача та перехід на відповідну сторінку, система відображає графік прийомів та надає можливість вибрати зручний час та дату для запису. Після того, як пацієнт вибрав потрібні дані відбувається перевірка на можливість створення нового запису, у разі не можливості створити, система повідомляє про це користувача. При успішній перевірці вибраної дати та часу, система записує пацієнта на прийом та оновлює графік прийомів лікаря, де з’являється новий запис.

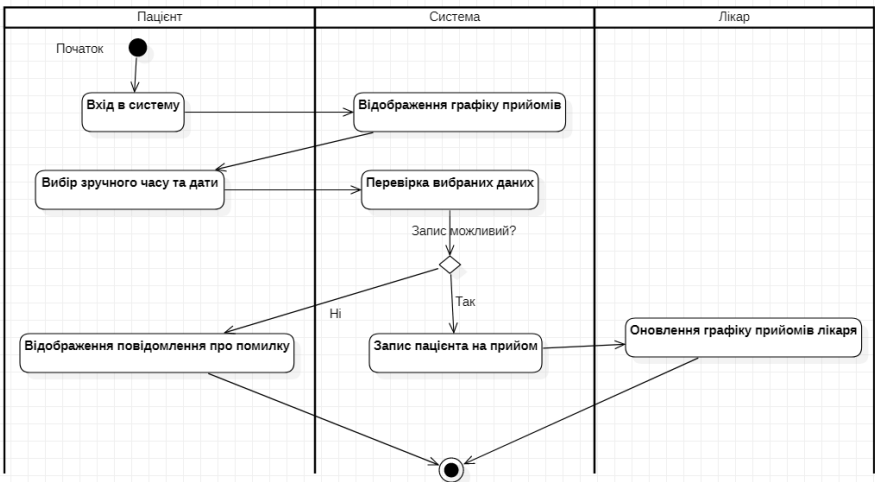


Рисунок 2.3 – Діаграма діяльності здійснення запису на прийом до лікаря

2.4 Реалізація функціоналу

Для доступу до використання повного функціоналу платформи користувачеві незалежно від його статусу потрібно зареєструватися чи авторизуватися в системі. Для проходження реєстрації потрібна електронна пошта, яка надалі використовується для входу (авторизації) в систему, отримання повідомлень та документів від клініки, можливість відновлення паролю тощо.

При розробці бекенду на Kotlin використовується патерн Repository. Патерн Repository - це спосіб організації доступу до даних в об'єктно-орієнтованому програмуванні. Основна ідея полягає в тому, щоб відокремити логіку доступу до даних від бізнес-логіки застосунку.

Основні принципи патерну Repository:

1. Інкапсуляція: Repository приховує деталі реалізації доступу до даних і представляє їх у вигляді колекції об'єктів предметної області. Це дозволяє ізолювати код застосунку від змін, пов'язаних з джерелом даних.
2. Абстракція: Repository надає уніфікований інтерфейс для доступу до даних, незалежно від їх джерела (база даних, API, файлова система тощо). Це дозволяє легко замінювати або модифікувати джерело даних без впливу на код застосунку.
3. Тестування: Патерн Repository спрощує тестування, оскільки він дозволяє замінювати реальні джерела даних на фейкові об'єкти (mock) під час тестування.

Реалізацію даного патерну у цьому випадку продемонстровано на двох прикладах - для User(користувач) Review(відгук) наступна є об'явлення необхідних функцій в інтерфейсі MongoRepository:

```
interface MongoRepository {  
    suspend fun addUser(user: User): Boolean  
    suspend fun getAllUsers(): List<User>  
    suspend fun addReview(review: Review): Boolean  
    suspend fun updateReview(review: Review): Boolean
```

```

suspend fun readMyReviews(skip: Int, author: String): List<Review>
suspend fun readAllReviews(): List<Review>
suspend fun checkUserExistence(user: User): User?
suspend fun checkUserId(id: String): Boolean
...
}

```

Об’явлення функцій для роботи з сутностями (Entities) є реалізація (імплементування) його функціонала в класі MongoDB:

```

@InitApi
fun initMongoDB(ctx: InitApiContext) {
    System.setProperty(
        "org.litote.mongo.test.mapping.service",
        "org.litote.kmongo.serialization.SerializationClassMappingTypeService"
    )
    ctx.data.add(MongoDB(ctx))
}

```

Розглянемо реалізацію створення конфігурації для ініціалізації бази даних:

```

class MongoDB(private val context: InitApiContext) : MongoRepository {
    private val client = MongoClient.create(MONGODB_URI)
    private val database = client.getDatabase(DATABASE_NAME)
    private val userCollection = database.getCollection<User>("user")
    private val reviewCollection = database.getCollection<Review>("review")
}

```

Розглянемо реалізацію створення класу підключення до БД та для роботи з сутностями:

```

override suspend fun addUser(user: User): Boolean {
    return userCollection.insertOne(user).wasAcknowledged()
}

```

Розглянемо реалізацію створення нового користувача:

```

override suspend fun getAllUsers(): List<User> {
    return userCollection.find().toList()
}

```

Розглянемо реалізацію отримання списку всіх користувачів користувача:

```

override suspend fun addReview(review: Review): Boolean {
    return reviewCollection
        .insertOne(review)
        .wasAcknowledged()
}

```

Далі було створено алгоритм для API за допомогою якого будуть надходити запити з фронтенду. В якості прикладу наведено код для створення нового користувача:

```

@Api(routeOverride = "adduser")
suspend fun addUser(context: ApiContext) {
    try {
        val user = context.req.body?.decodeToString()?.let {
            Json.decodeFromString<User>(it)
        }
        val newUser = user?.copy(
            _id = ObjectIdGenerator().generate().toString()
        )
        var responseBody = Json.encodeToString(
            newUser?.let {
                context.data.getValue<MongoDB>().addUser(it)
            }
        )
        context.res.setBodyText(responseBody)
    } catch (e: Exception) {
        context.res.setBodyText(Json.encodeToString(e))
    }
}

```

Наступним кроком була реалізація запиту зі сторони фронтенду на API нашого сервера, для чого було створено файл ApiCalls.kt, в якому знаходяться необхідні для цього функції, розглянемо реалізацію створення нового користувача:

```

suspend fun addUser(user: User): String {
    return window.api.tryPost(
        apiPath = "adduser",

```

```

        body = Json.encodeToString(user).encodeToByteArray()
    )?.decodeToString().toString()
}

```

Розглянемо реалізацію створення нового відгуку:

```

suspend fun addReview(review: Review): String {
    return window.api.tryPost(
        apiPath = "addreview",
        body = Json.encodeToString(review).encodeToByteArray()
    )?.decodeToString().toString()
}

```

На основі вище зазначеного була створена загальна діаграма компонентів системи (рис. 2.13).

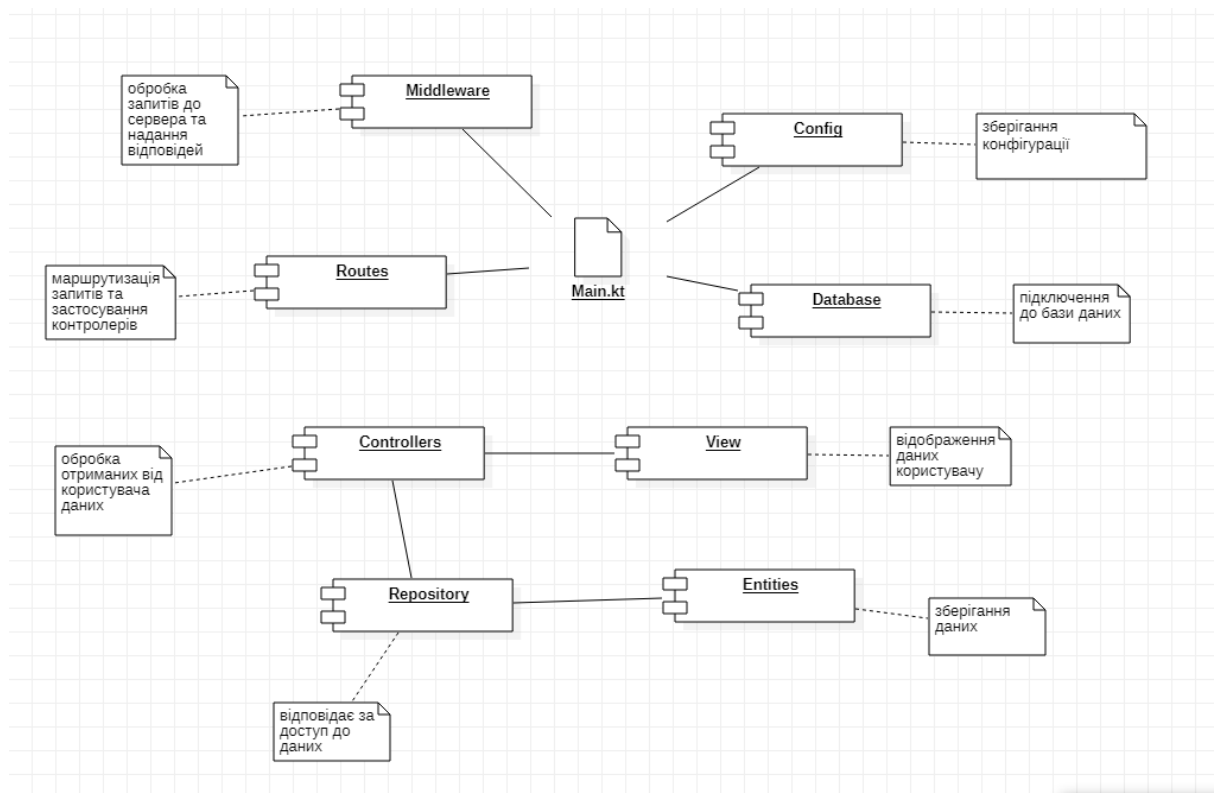


Рисунок 2.13 – Діаграма компонентів

Висновки до другого розділу

Проведено аналіз і визначено варіанти використання системи та створено об'єктно-орієнтовану структуру. Спроектowano базу даних для зберігання

інформації про пацієнтів, графіки прийомів та інші важливі дані. Реалізовано алгоритми роботи системи, що включають процес реєстрації, бронювання прийомів, управління записами. Таким чином, була розроблена комплексна система, яка забезпечує автоматизацію та зручність у роботі стоматологічної клініки.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З САЙТОМ

3.1 Порядок встановлення та налаштування параметрів сайту

Встановлення та налаштування платформи вимагає розгортання всіх його компонентів та налаштування як кожного елементу окремо, так і загальної системи в цілому.

Для опису архітектури платформи було створено діаграму розгортання (рис. 3.1).

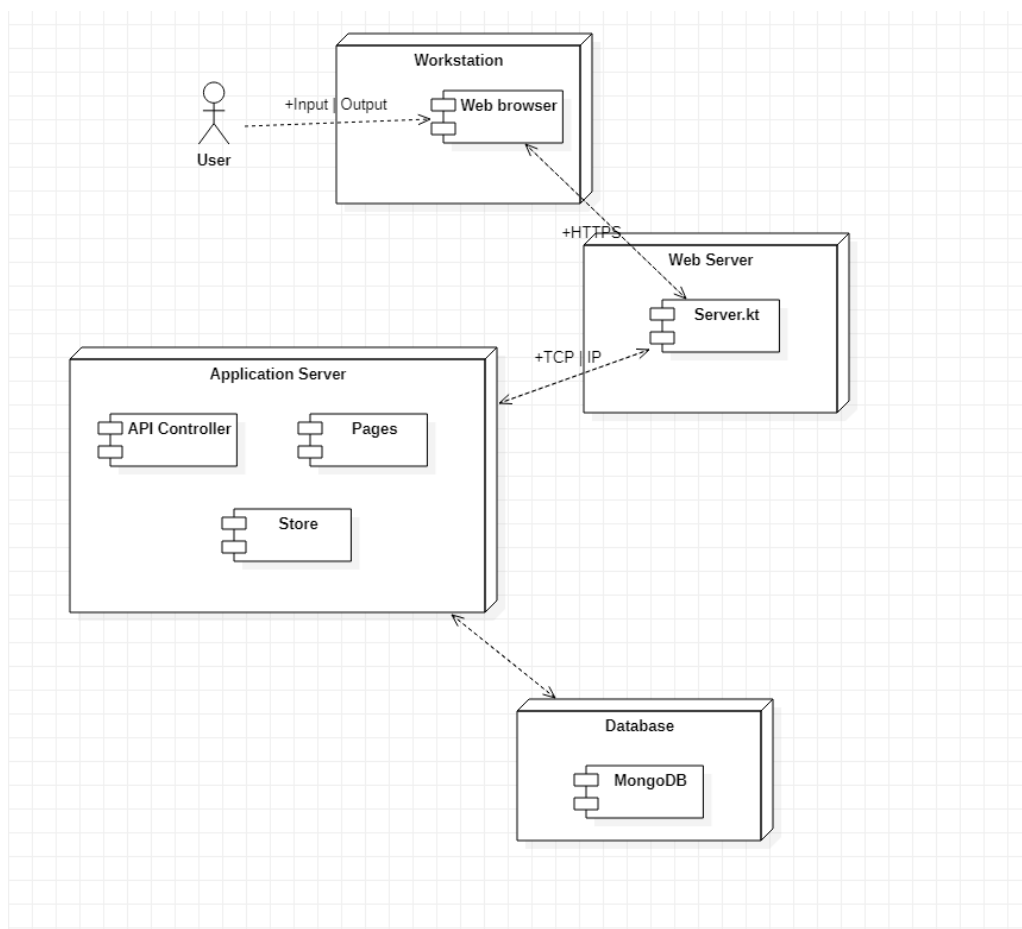


Рисунок 3.1 – Діаграма розгортання

На діаграмі розгортання зображена система, яка складається з Workstation (Веб-браузер), Web Server (Server.kt), Application Server (API Controller, Pages, Store), Database (MongoDB).

Workstation (Веб-браузер) виконує наступні функції:

- переглядає веб-сторінки, взаємодіє з веб-додатком;
- відправляє HTTP-запити до Веб-сервера;

- отримує та обробляє відповіді від Веб-сервера.

Для виконання цих функцій використовується система відповідей чи запитів, що базується на протоколі HTTP/HTTPS. В якості інструменту, для данного елементу можливо використати Google Chrome, Mozilla Firefox, Microsoft Edge, Safari та інші.

Web Server (Server.kt) виконує наступні функції:

- приймає HTTP-запити від Workstation;
- перенаправляє запити до Application Server;
- надає статичний вміст (HTML, CSS, JS) до Workstation;
- може здійснювати певні обчислення та обробку запитів перед

передачею їх до Application Server.

Для з'єднання з Workstation і/або Application Server можуть використовуватися різні протоколи (як HTTP/HTTPS так і TCP/IP) в залежності від того яка буде вибрана система – внутрішня мережа та власний сервер чи хмарні технології.

Application Server (API Controller, Pages, Store) можна розбити також на підкомпоненти:

- api Controller: приймає запити від Web Server, обробляє їх, взаємодіє з Pages та Store;
- store: зберігає дані, може бути інтегрований з Database;
- обробляє ділову логіку веб-додатку;
- взаємодіє з Database;
- отримує та обробляє дані від Web Server;
- створює відповіді для Web Server.

Взаємодія з Web Server за допомогою внутрішньої мережі, а Database - через API (хоча можливо встановити на власний сервер MongoDB).

Database (MongoDB), на даний елемент покладені наступні функції:

- зберігання даних, необхідних для веб-додатку;
- доступ через API для Application Server;
- надання можливості пошуку, сортування та фільтрації даних.

Діаграма показує архітектуру веб-додатку, де різні компоненти працюють разом для обробки запитів користувача та надання даних. Взаємодія між компонентами здійснюється за допомогою HTTPS/HTTP-протоколу та API.

3.2 Структура інтерфейсу веб-додатку та порядок роботи

У веб-сайті користувачі можуть заходити з комп'ютера чи планшета. Сайт є адаптованим для стандартних пристроїв.

Перейшовши за посиланням на сайт, користувач першим побачить головну сторінку (рис. 3.2).

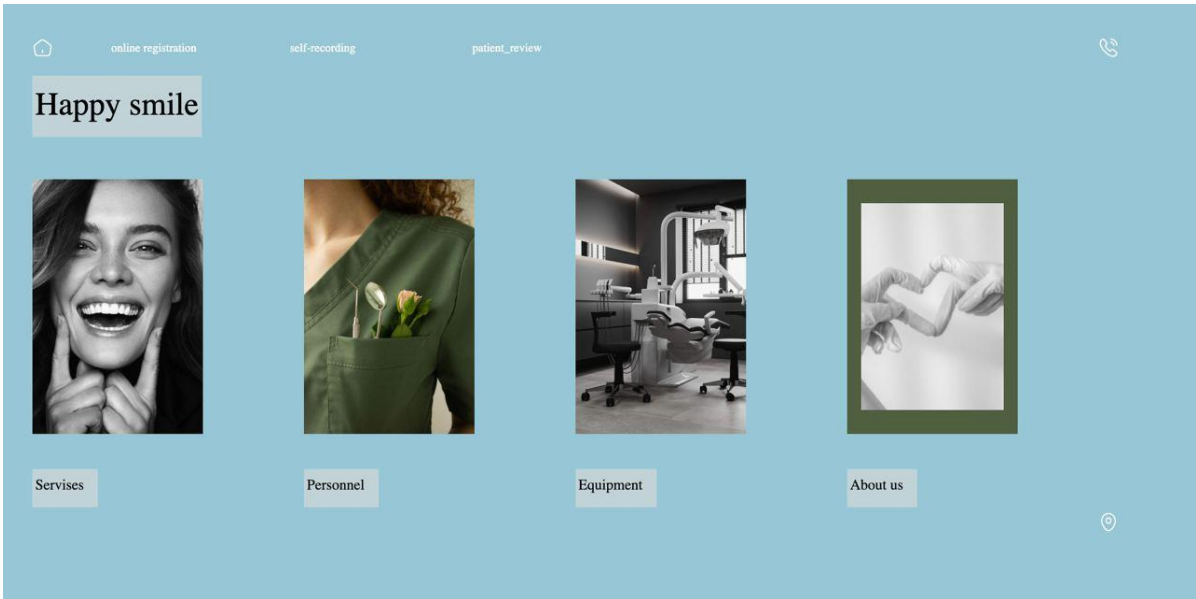


Рисунок 3.2 – Головна сторінка

На кожній сторінці розташована кнопка (рис. 3.3), при натисканні на неї користувач переходить на головну сторінку.



Рисунок 3.3 – Кнопка для переходу на головну сторінку

На деяких сторінках розташована кнопка (рис. 3.4), при натисканні на неї користувач переходить на сторінку контакти (рис. 3.5).



Рисунок 3.4 – Кнопка для переходу на сторінку з контактами

На сторінці з контактами є посилання на соціальну мережу, WhatsApp, пошта та номер телефону (рис. 3.5).



Рисунок 3.5 – Сторінка з контактами

При натисканні користувачем на посилання Instagram, відбувається перехід по посиланню (рис. 3.7).



Рисунок 3.6 – Кнопка для переходу на сторінку з геолокацією

При натисканні користувачем на кнопку (рис 3.6), відбувається перехід на гугл-карти з місцезнаходженням (рис. 3.8).



happysmail0

Стежити



0 дописів

Читачі: 0

Стежить: 0

ДОПИСИ

ПОЗНАЧЕНО

**Ще немає дописів**

Рисунок 3.7 – Інстаграм клініки

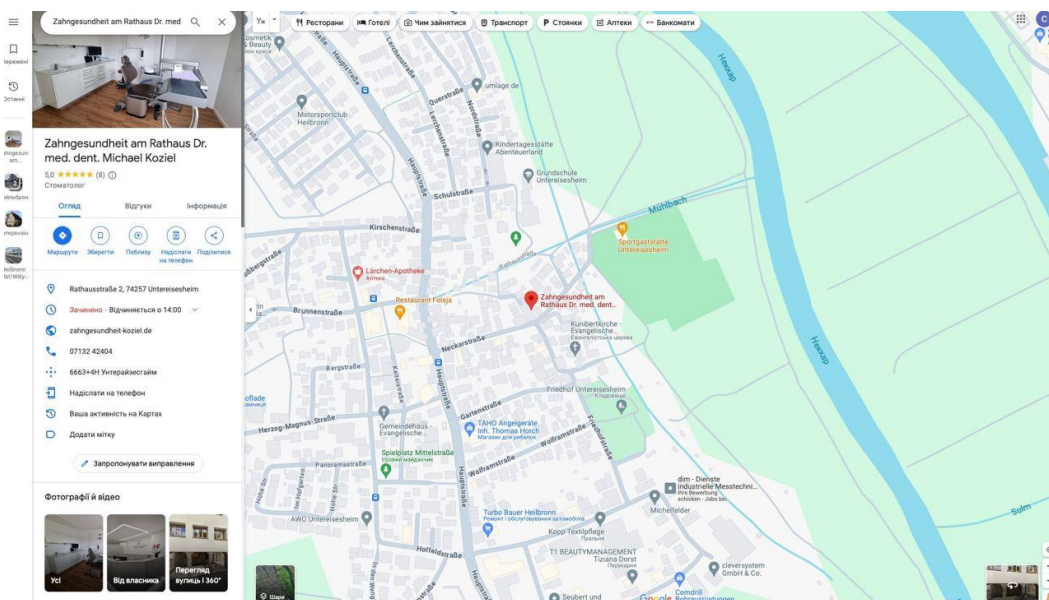


Рисунок 3.8 – Сторінка з геолокацією

Натискаючи на текст на головній сторінці online registration ,користувач переходить на сторіну Online registration (рис. 3.9).На сторінці розташована форма для реєстрації пацієнта та кнопка зберегти та відмінити дії.

Натискаючи на текст на головній сторінці self-recording ,користувач переходить на сторіну Self-recording (рис. 3.10). На сторінці розташована форма для самозапису пацієнта та кнопка зберегти та відмінити дії.

Натискаючи на текст на головній сторінці patient_review , користувач переходить на сторіну Patient_reveiw (рис. 3.11, рис. 3.12). На сторінці

розташовані відгуки пацієнтів клініки та кнопка (рис. 3.13), натискаючи на яку користувач переходить на сторінку Patient_reveiw з формою для додавання відгука (рис. 3.14).

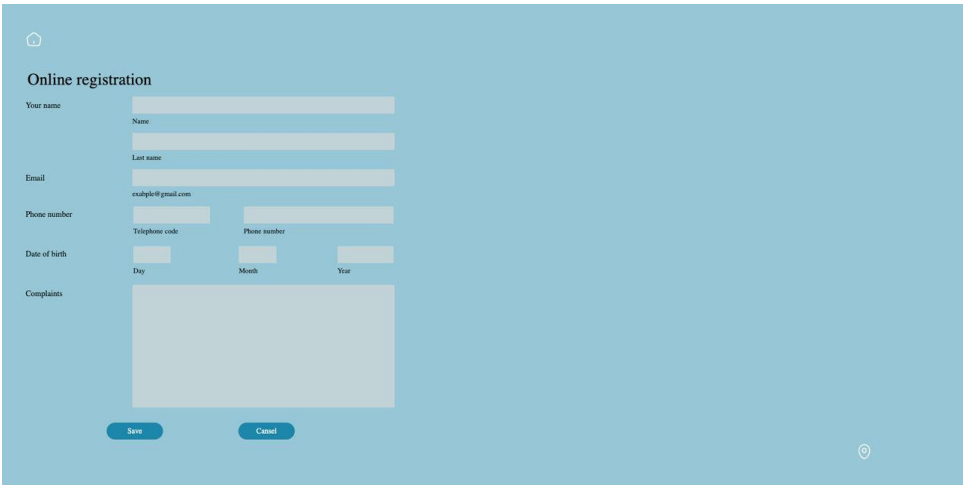
A screenshot of a web form titled "Online registration" on a light blue background. The form includes input fields for "Your name" (with sub-fields for "Name" and "Last name"), "Email" (with a placeholder "example@gmail.com"), "Phone number" (with sub-fields for "Telephone code" and "Phone number"), "Date of birth" (with sub-fields for "Day", "Month", and "Year"), and a "Complaints" text area. At the bottom are "Save" and "Cancel" buttons. A home icon is in the top left, and a user profile icon is in the bottom right.

Рисунок 3.9 – Сторінка з формою для реєстрації пацієнта

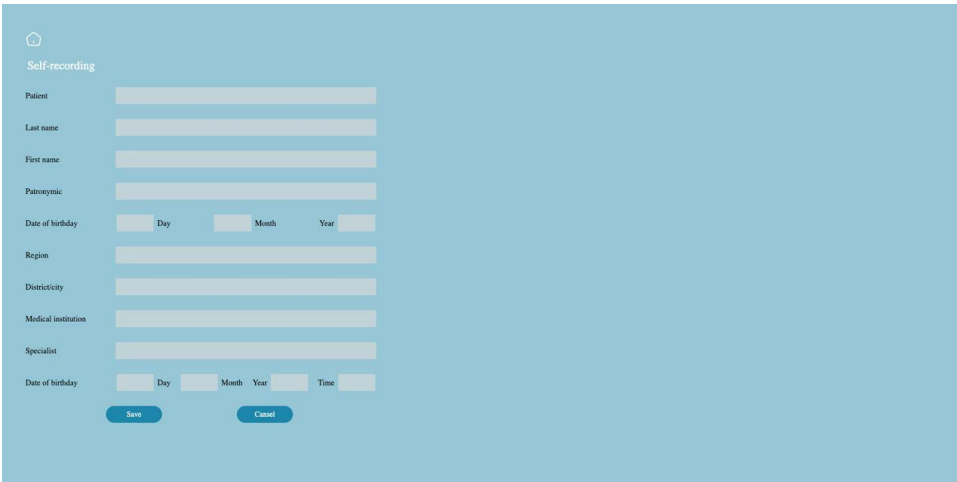
A screenshot of a web form titled "Self-recording" on a light blue background. The form includes input fields for "Patient", "Last name", "First name", "Pseudonym", "Date of birthday" (with sub-fields for "Day", "Month", and "Year"), "Region", "District/city", "Medical institution", "Specialist", and "Date of birthday" (with sub-fields for "Day", "Month", "Year", and "Time"). At the bottom are "Save" and "Cancel" buttons. A home icon is in the top left.

Рисунок 3.10 – Сторінка з формою для самозапису пацієнта

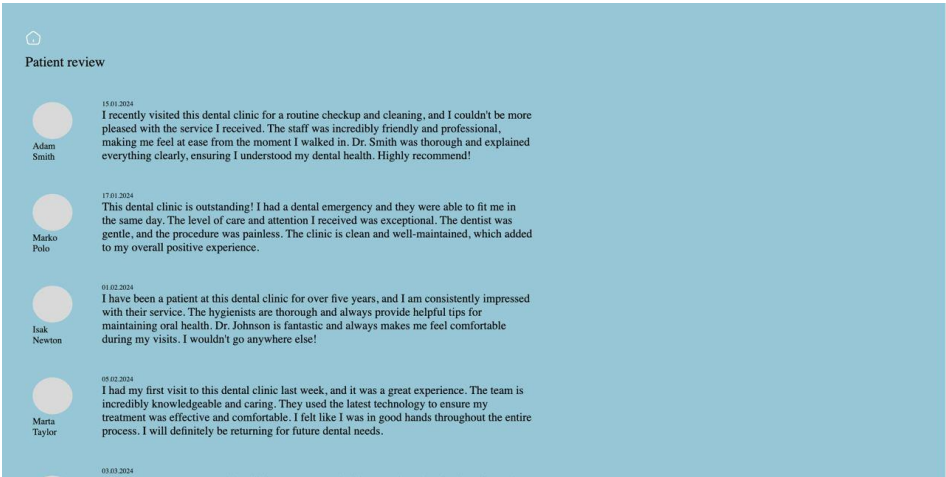
A screenshot of a web page titled "Patient review" on a light blue background. It displays four reviews, each with a circular profile picture, a name, a date, and a text review. The reviews are from Adam Smith (15.01.2024), Marko Polo (17.01.2024), Isak Newton (01.02.2024), and Marta Taylor (05.02.2024). A date "03.03.2024" is shown at the bottom left. A home icon is in the top left.

Рисунок 3.11 – Сторінка з відгуками пацієнтів

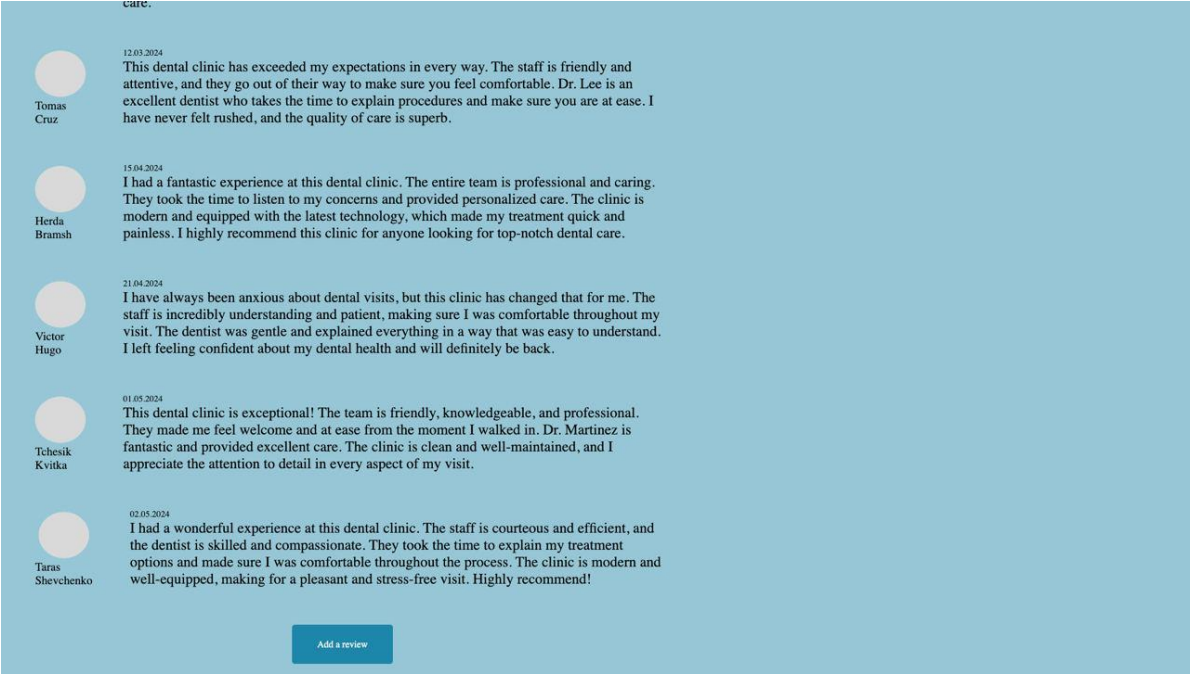


Рисунок 3.12 – Сторінка з відгуками пацієнтів



Рисунок 3.13 – Кнопка для додавання відгука

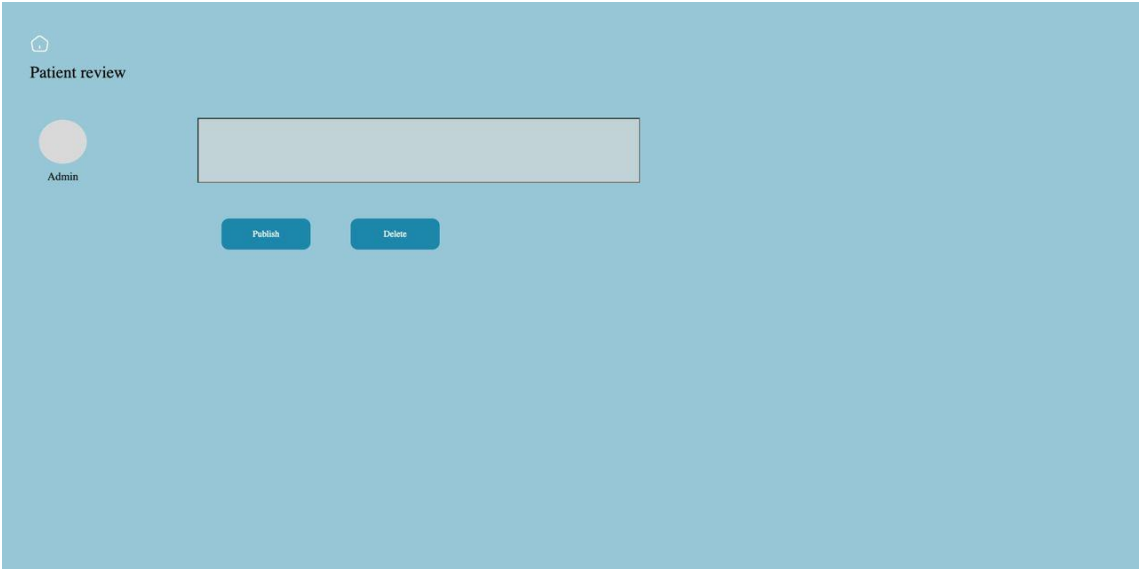


Рисунок 3.14 – Сторінка з формою для додавання відгуків

На головній сторінці при натисканні на текст Personel або картинку (рис. 3.15), користувач переходить на сторінку Personel (рис. 3.16). На сторінці розташована інформація про персонал.



Рисунок 3.15 – Картинка та текст для переходу на сторінку Personel

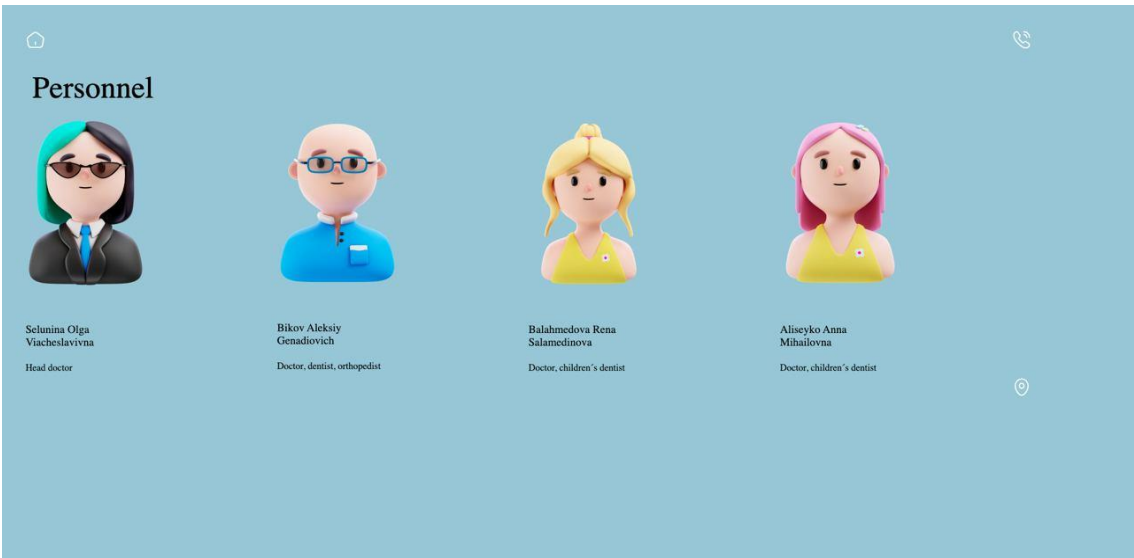


Рисунок 3.16 – Сторінка з інформацією про персонал

На головній сторінці при натисканні на текст Equipment або картинку (рис. 3.17), Користувач переходить на сторінку Equipment (рис. 3.18). На сторінці розташовані фото та інформація про обладнання.

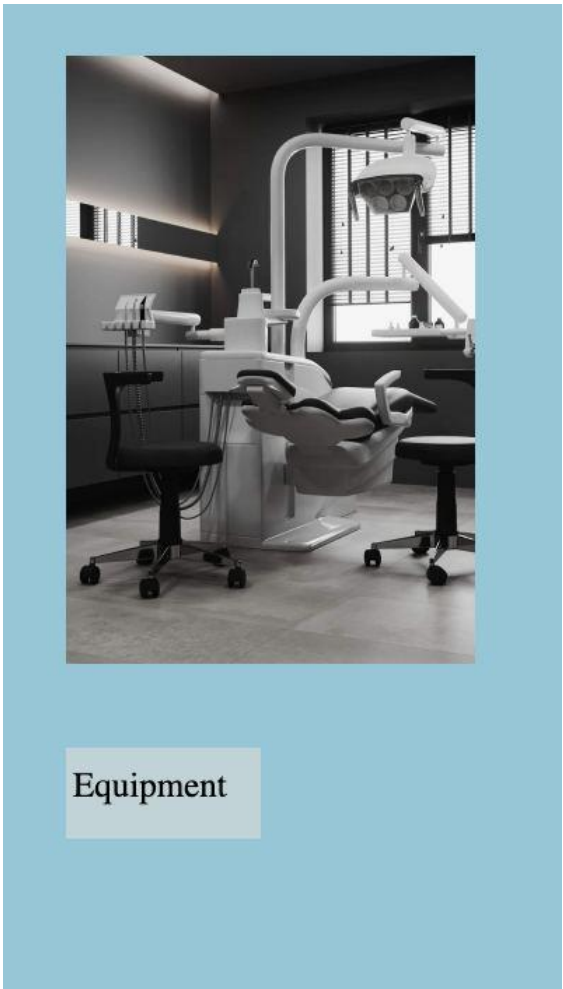


Рисунок 3.17 – Картинка та текст для переходу на сторінку Equipment

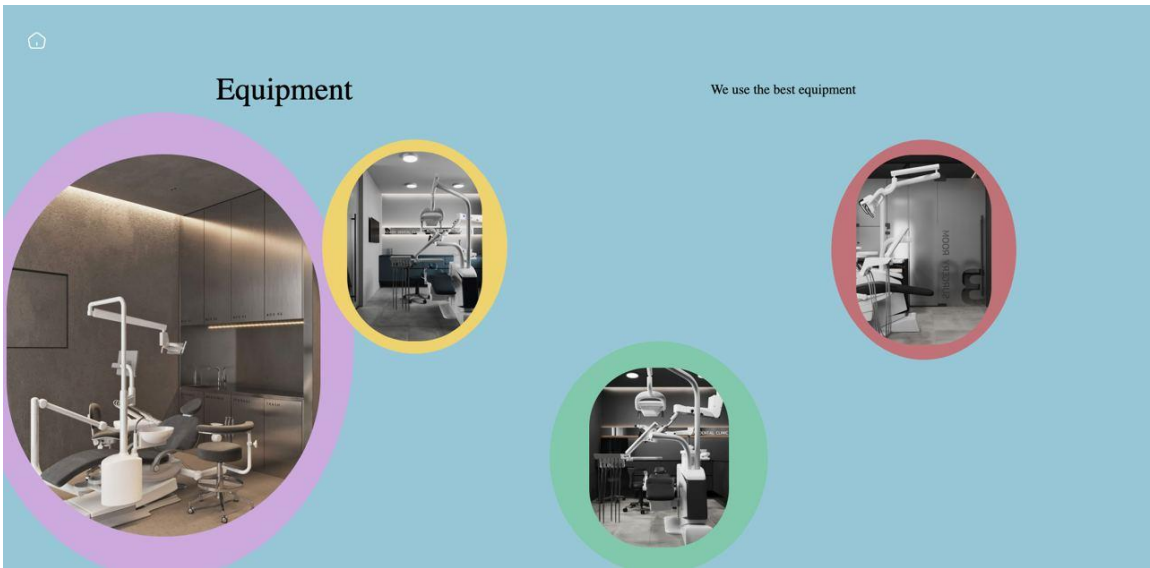


Рисунок 3.18 – Сторінка з фото та інформацією про обладнання

На головній сторінці при натисканні на текст Servises або картинку (рис. 3.19), користувач переходить на сторінку Servises (рис. 3.20). На сторінці розташована інформація про послуги.

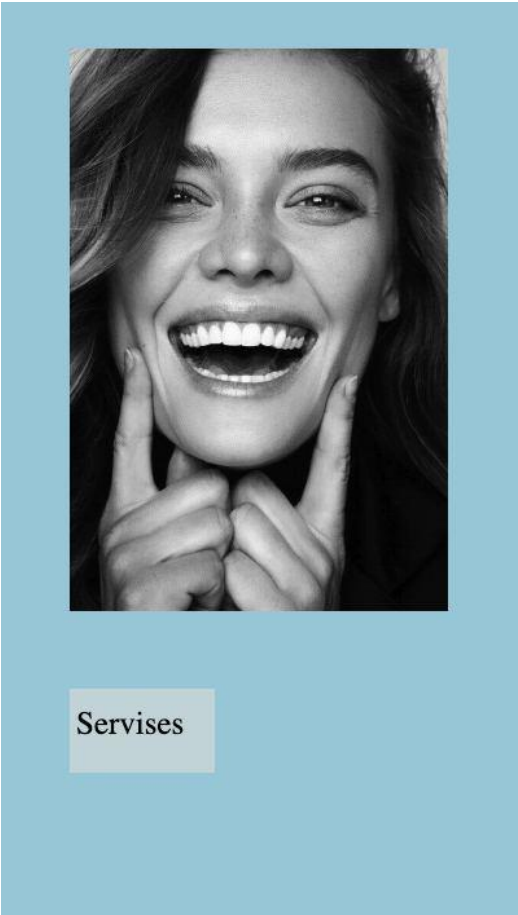


Рисунок 3.19 – Картинка та текст для переходу на сторінку Servises



Рисунок 3.20 – Сторінка з описом про послуги

На головній сторінці при натисканні на текст About us або картинку (рис. 3.21), Користувач переходить на сторінку about us (рис. 3.22). На сторінці розташована інформація про клініку.



Рисунок 3.21 – Фото та текст для переходу на сторінку About us

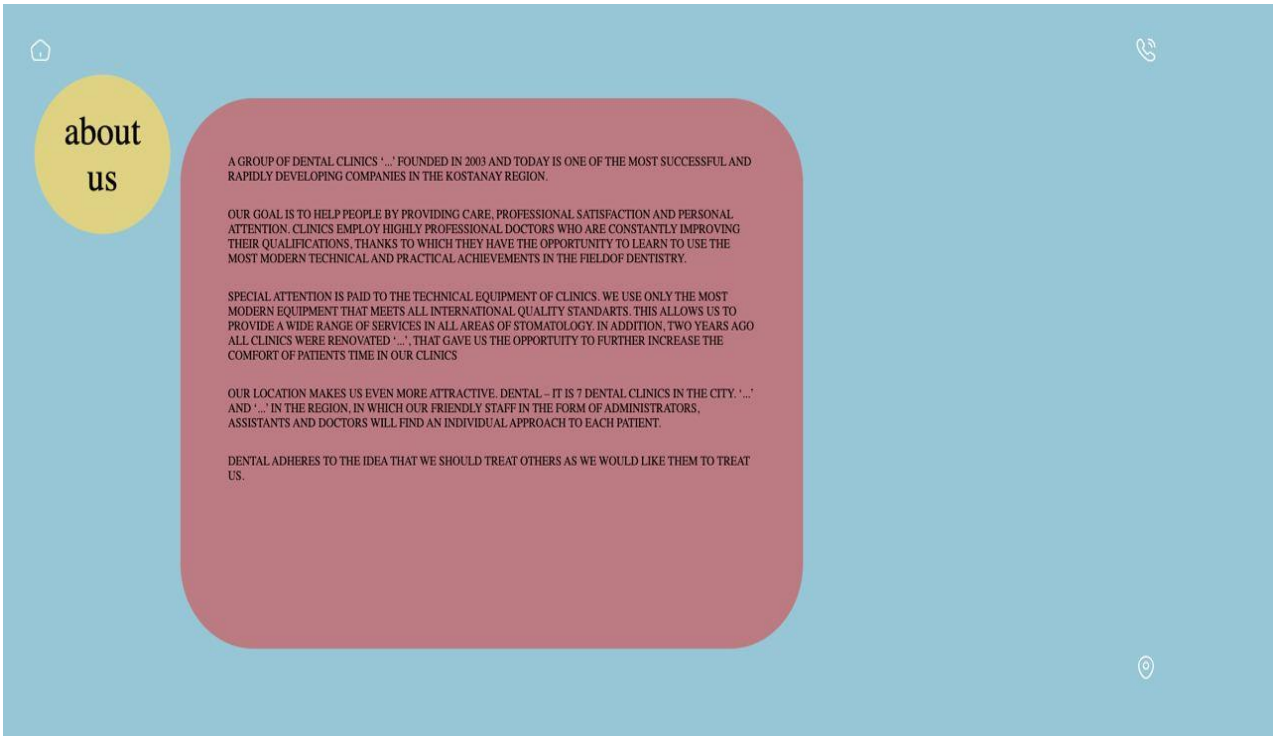


Рисунок 3.22 – Сторінка з описом про клініку

3.3 Тестування роботи веб-додатку

Тестування важливий етап у багатьох сферах діяльності, включаючи програмне забезпечення, веб-сайти, медичне обладнання, розробці продуктів, який допомагає забезпечити їх високу якість та надійність. Цей процес дозволяє виявити та виправити помилки, знизити ризики для користувачів та бізнесу, а також підтримати відповідність вимогам. Існує кілька типів тестування, таких як функціональне, навантажувальне, автоматизоване, безпекове та тестування сумісності, які сприяють підвищенню якості продукту та забезпеченню його успішної роботи.

При розробці сайту було проведено тестування як на етапі створення, так і після завершення процесу розробки. По виявленні та усуненні помилок було проведено повторне тестування продукту, щоб переконатися у його стабільності та правильному функціонуванні.

Наведемо приклад тестування при валідації реєстрації нового пацієнта.

При введенні імені менше 2 символів з'являється попередження (рис. 3.23).

The screenshot shows a web form titled "Online registration" with a dark blue background. The form fields include: "Your name" (with a sub-label "Name" and value "Martinez"), "Password" (with a sub-label "password" and value "123456"), "Email" (with value "l.martinez@example.com"), "Phone number" (split into "Telephone code" with value "+1" and "Phone number" with value "567-890-1234"), and "Date of birth" (split into "Day" with value "10", "Month" with value "12", and "Year" with value "1993"). There is a "Complaints" section with a large text area. A white error message box on the right states: "Name is not valid. Name must have 2 or more char". At the bottom are "Save" and "Cancel" buttons.

Рисунок 3.23 – Повідомлення про помилку «Невірно введено ім'я»

Теж саме попередження з’являється якщо і прізвище буде містити менше 2 символів (рис. 3.24).

The screenshot shows an 'online registration' form with the following fields: 'Your name' (with sub-fields 'Name' and 'Last name'), 'Password', 'Email', 'Phone number' (with sub-fields 'Telephone code' and 'Phone number'), and 'Date of birth' (with sub-fields 'Day', 'Month', and 'Year'). A 'Complaints' text area is at the bottom. The 'Last name' field contains 'Laura', which triggers a validation error: 'Last name is not valid. Last name must have 2 or more char'. The 'Save' and 'Cancel' buttons are at the bottom right.

Рисунок 3.24 – Повідомлення про помилку «Невірно введено прізвище»

Наступне поле яке також повинне пройти валідації це пароль, він не має бути менше за 6 симолів (рис. 3.25).

The screenshot shows the same 'online registration' form as Figure 3.24, but with the 'Password' field containing '12345'. This triggers a validation error: 'Password is not valid. Password must have 6 or more char'. The 'Save' and 'Cancel' buttons are at the bottom right.

Рисунок 3.25 – Повідомлення про помилку «Невірно введено пароль»

Остання валідація для цієї форми – перевіряє електронну адресу (рис. 3.26).

online registration

Online registration

Your name

Laura

Name

Martinez

Last name

Password

123456

password

Email

l.martinez@example

exable@gmail.com

Phone number

+1

567-890-1234

Telephone code

Phone number

Date of birth

10

12

1993

Day

Month

Year

Complaints

Email Address is not valid.

Save

Cancel

Рисунок 3.26 – Повідомлення про помилку «Невірно введена електронна пошта»

Після того виправлено всі з’являється повідомлення про успішне створення нового користувача-пацієнта (рис. 3.27).

online registration

Online registration

Your name

Laura

Name

Martinez

Last name

Password

123456

password

Email

l.martinez@example.com

exable@gmail.com

Phone number

+1

567-890-1234

Telephone code

Phone number

Date of birth

10

12

1993

Day

Month

Year

Complaints

User are created.

Save

Cancel

Рисунок 3.27 – Повідомлення про створеного пацієнта

І як результат успішної реєстрації система нас переадресує на сторінку профіля пацієнта (рис. 3.28).

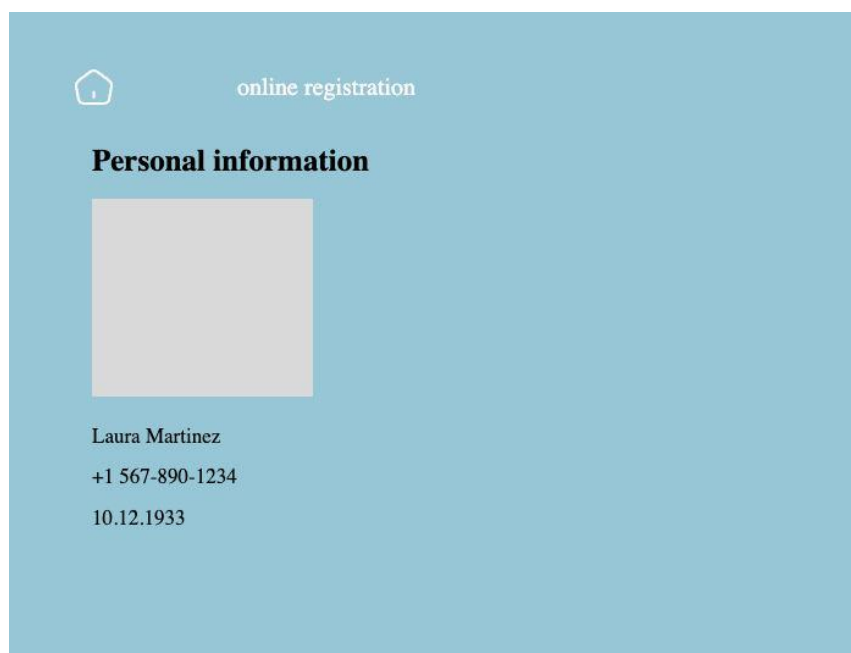


Рисунок 3.28 – Профіль пацієнта

Висновки до розділу 3

В даному розділі було описано інтерфейс, порядок роботи з веб-сайтом та діаграма, яка показує архітектуру веб-додатку, де різні компоненти працюють разом для обробки запитів користувача та надання даних. Взаємодія між компонентами здійснюється за допомогою HTTPS/HTTP-протоколу та API.

Розглянуто процес встановлення та налаштування платформи, включаючи розгортання всіх компонентів системи. Описано структуру інтерфейсу веб-додатку, де кожен елемент має чітко визначену функцію. Проведено тестування роботи веб-додатку, що підтвердило його відповідність встановленим вимогам та показало високу зручність і ефективність для користувачів.

Представлені результати тестування функціоналу веб-сайту для стоматологічної клініки. Розглянуті вимоги до тестування, проведені етапи тестування і оцінка його результатів.

ВИСНОВКИ

У ході бакалаврської роботи було поставлено амбітну мету – розробити веб-додаток для стоматологічної клініки, який би дозволив пацієнтам самостійно записуватися на прийом. Це завдання вимагало комплексного підходу, який охоплював не тільки створення привабливого інтерфейсу, але й розробку надійної системи обробки даних.

Першим кроком було ретельне дослідження галузі стоматології та аналіз вже існуючих онлайн-систем запису. Хоча на ринку було багато схожих рішень, жодне з них не відповідало вимогам щодо функціональності та простоти використання.

Щоб створити оптимальну систему, було обрано модульну архітектуру як для серверної, так і для клієнтської частини. Серцем цієї архітектури став фреймворк Kobweb, заснований на мові Kotlin Jetpack Compose. Вибір Kotlin Multi Platform як середовища виконання дозволив забезпечити високу продуктивність та ефективність системи.

Для зберігання інформації про пацієнтів, їхні записи та іншу важливу інформацію було використано документо-орієнтовану базу даних MongoDB, яка вирізняється швидкістю роботи та гнучкістю.

Наступним кроком було формулювання та аналіз функціональних та нефункціональних вимог до системи. Проаналізовано можливі сценарії використання, розроблено структуру системи та виявлено зв'язки між її компонентами. Зрештою, було сформульовано сутності системи, які представляють дані, що зберігаються в базі даних.

Результат роботи було розроблено систему, яка спрощує процес запису на прийом до стоматолога та інших спеціалістів клініки. Система має інтуїтивний інтерфейс та надає можливість пацієнтам зручно планувати візити, вибираючи зручний час та спеціаліста.

Крім розробки системи також було створено інструкції щодо її встановлення та налаштування. Dodatok пройшов тестування, що дозволило виявити та виправити потенційні проблеми.

					ІПЗ.КР.Б-121-24-ПЗ	
						58

Розроблений веб-додаток має потенціал і можливість на удосконалення надалі. Він не тільки спрощує роботу клініки, але й збільшує її привабливість для пацієнтів. За допомогою додатка клініка може ефективно інформувати пацієнтів про свої послуги, проводити маркетингові кампанії та надавати інформаційну підтримку.

Ця система стане цінним інструментом для стоматологічної клініки, сприяючи зростанню її ефективності та задоволеності пацієнтів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Аллен Б. Веб-розробка для початківців: від нуля до професіонала / Б. Аллен. – Київ: Техніка, 2022.
2. Дмитро К. Застосування Kotlin для розробки веб-додатків / К. Дмитро. – Львів: Академія, 2021.
3. Керніган Б. Мова програмування C / Б. Керніган, Д. Рітчі. – Київ: Вища школа, 2020.
4. Найджел Р. Ефективне програмування на Kotlin / Р. Найджел. – Харків: Фоліо, 2021.
5. Пітер К. MongoDB: Посібник розробника / К. Пітер. – Київ: Освіта, 2022.
6. Стюарт Ч. Бази даних: теорія та практика / Ч. Стюарт. – Львів: Львівська політехніка, 2021.
7. Козлов А. Kobweb: Новий фреймворк для розробки веб-додатків на Kotlin / А. Козлов, 2023.
8. Іванов С. Розробка веб-додатків з Kobweb: порівняння з іншими фреймворками / С. Іванов. – Київ, 2022.
9. Сидоренко О. Kotlin Multi Platform: переваги та недоліки / О. Сидоренко, 2022.
10. Петренко В. Розробка кросплатформених додатків на Kotlin / В. Петренко. – Одеса: Розробка мобільних додатків, 2023.
11. Офіційний сайт Kobweb [Електронний ресурс] - Режим доступу до ресурсу: <https://kobweb.io/>
12. Документація Kobweb [Електронний ресурс] - Режим доступу до ресурсу: <https://kobweb.io/docs/>
13. Офіційний сайт MongoDB [Електронний ресурс] - Режим доступу до ресурсу: <https://www.mongodb.com/>
14. Документація MongoDB [Електронний ресурс] - Режим доступу до ресурсу: <https://www.mongodb.com/docs/>

15. Kotlin Multi Platform документація [Електронний ресурс] - Режим доступу до ресурсу:

<https://kotlinlang.org/docs/multiplatform.html>

16. Kotlin Jetpack Compose документація [Електронний ресурс] - Режим доступу до ресурсу: <https://developer.android.com/jetpack/compose>

17. Документація JetBrains IntelliJ IDEA [Електронний ресурс] - Режим доступу до ресурсу: <https://www.jetbrains.com/help/idea/>

18. Документація Git [Електронний ресурс] - Режим доступу до ресурсу:

<https://git-scm.com/doc>

19. Інструкції з встановлення та налаштування Kobweb Server [Електронний ресурс] - Режим доступу до ресурсу:

<https://kobweb.io/docs/getting-started/>

20. Інструкції з встановлення та налаштування MongoDB [Електронний ресурс] - Режим доступу до ресурсу:

<https://www.mongodb.com/docs/manual/installation/>

21. Сайт HB Clinic [Електронний ресурс] - Режим доступу до ресурсу:

<https://hbclinic.com.ua/>

22. Сайт Familia [Електронний ресурс] - Режим доступу до ресурсу:

<https://familia.ua/>

23. Сайт KES Clinic [Електронний ресурс] - Режим доступу до ресурсу:

<https://kes.com.ua/ua/>

ДОДАТКИ

					ІІЗ.КР.Б-121-24-ІІЗ	
						62

Реалізація `data class` `OnlineRegistrationPageUiState`

```

data class OnlineRegistrationPageUiState(
    var id: String = "",
    var name: String = "",
    var lastName: String = "",
    var password: String = "",
    var email: String = "",
    var phoneCod: String = "",
    var phoneNumber: String = "",
    var dayBirth: String = "",
    var monthBirth: String = "",
    var yearBirth: String = "",
) {
    fun reset() = this.copy(
        id = "",
        name = "",
        lastName = "",
        password = "",
        email = "",
        phoneCod = "",
        phoneNumber = "",
        dayBirth = "",
        monthBirth = "",
        yearBirth = "",
    )
}

@Page(routeOverride = "/online_registration")
@Composable
fun OnlineRegistrationPage() {
    val context = rememberPageContext()
    val scope = rememberCoroutineScope()
    var uiState by remember{
mutableStateOf(OnlineRegistrationPageUiState()) }
    var response by remember{
mutableStateOf<ApiUserListResponse>(ApiUserListResponse.Loading)
}

    var invalidEmailPopup by remember{ mutableStateOf(false) }
    val breakpoint = rememberBreakpoint()
    var invalidNamePopup by remember{ mutableStateOf(false) }
}

    var invalidLastNamePopup by remember{
mutableStateOf(false) }
    var invalidPasswordPopup by remember{
mutableStateOf(false) }
    LaunchedEffect(Unit) {
        getAllUsers(

```

```

        onSuccess = {
            if (it is ApiUserListResponse.Success) {
                response = it
            }
        },
        onError = {
            println("Error")
            response =
                ApiUserListResponse.Error(it.message.toString())
        }
    )
}
if (invalidNamePopup) {
    MessagePopup(
        message = "Name is not valid. Name must have
2 or more char",
        onDialogDismiss = {
            invalidNamePopup = false
        }
    )
}
if (invalidLastNamePopup) {
    MessagePopup(
        message = "Last name is not valid. Last name
must have 2 or more char",
        onDialogDismiss = {
            invalidLastNamePopup = false
        }
    )
}
if (invalidPasswordPopup) {
    MessagePopup(
        message = "Password is not valid. Password
must have 6 or more char",
        onDialogDismiss = {
            invalidPasswordPopup = false
        }
    )
}
if (invalidEmailPopup) {
    MessagePopup(
        message = "Email Address is not valid.",
        onDialogDismiss = {
            invalidEmailPopup = false
        }
    )
}
}
}

```

```

Column (
    modifier = Modifier
        .fillMaxSize ()
        .backgroundColor (Theme.Primary.rgb)
) {
Column (
    modifier = Modifier
        .fillMaxSize ()
        .maxWidth (PAGE_WIDTH.px)
        .padding (top = 60.px, leftRight = 60.px)
        .background (Theme.Primary.rgb),
    verticalArrangement = Arrangement.Top,
    horizontalAlignment = Alignment.Start
) {
HeaderSection (
    breakpoint = breakpoint,
    trailing = Res.Icon.calling,
    isTrailing = true,
    isOnlineRecord = true,
    onOnlineRecord = {},
    onCalling = {}
)

MainOnlineRecordSection (
    onSave = {
},
        onCancel = {
uiState.reset ()
        },
        onInvalidName = {
scope.launch {
            invalidNamePopup = true
            delay (2000)
            invalidNamePopup = false
        }
},
        onInvalidLastName = {
scope.launch {
            invalidLastNamePopup = true
            delay (2000)
            invalidLastNamePopup = false
        }
},
        onInvalidPassword = {
scope.launch {
            invalidPasswordPopup = true
            delay (2000)

```

```

invalidPasswordPopup = false
    }
    },
    onInvalidEmail = {
scope.launch {
    invalidEmailPopup = true
        delay(2000)
            invalidEmailPopup = false
        }
    }
)
    GetAllUsers(response = response)
    FooterSection ()
}
}
}

```