**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА**

**ФАКУЛЬТЕТ УПРАВЛІННЯ ФІНАНСАМИ ТА БІЗНЕСУ**

**Кафедра цифрової економіки та бізнес-аналітики**

**КУРСОВА РОБОТА**

**з дисциплін  професійної та практичної підготовки**

на тему:

Інформаційна система приватної клініки

**спеціальність:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**051 «Економіка»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

                              (код та найменування спеціальності)

**спеціалізація:** \_\_\_\_\_\_\_\_\_«Інформаційні технології в бізнесі» \_\_\_\_\_\_\_\_\_\_\_

                             (найменування спеціалізації)

**освітній ступінь: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**бакалавр**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

              (бакалавр/магістр)

|  |  |  |
| --- | --- | --- |
| **Науковий керівник:**  к.ф.-м.н.доц. кафедри **Депутат Б.Я**.    (науковий ступінь, посада, прізвище, ініціали)  \_\_\_\_\_\_\_\_\_\_  **“\_\_\_\_”** \_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р.       (підпис) |  | **Виконавець:**  студентка групи УФЕ-31с Данильців Софія                         (прізвище, ініціали)  \_\_\_\_\_\_\_\_\_\_\_\_  **“\_\_\_\_”** \_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р.      (підпис) |

**Загальна**

**кількість балів**  \_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

                                                                            (підпис, ПІП членів комісії)

**ЛЬВІВ 2023**

ЗМІСТ

[ВСТУП 3](#_Toc137846736)

[1. АНАЛІЗ ВИМОГ 5](#_Toc137846737)

[**2.1** **Постановка завдання** 5](#_Toc137846738)

[**2.2** **Розробка моделі варіантів використання веб-сайту** 6](#_Toc137846739)

[**2.3** **Аналіз засобів реалізації (техніко-економічне обґрунтування вибору)** 7](#_Toc137846740)

[2. РОЗРОБКА БАЗИ ДАННИХ 14](#_Toc137846741)

[**2.1** **Модель даних** 14](#_Toc137846742)

[2.2 **Нормалізація бази даних** 16](#_Toc137846743)

[**2.3** **Типи даних** 18](#_Toc137846744)

[**2.4** **Реалізація цілісності даних** 20](#_Toc137846746)

[3. РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ ТА ВЕБ-ІНТЕРФЕЙСУ 22](#_Toc137846748)

[**3.1** **Структура проекту** 22](#_Toc137846749)

[**3.2** **Розробка системи** 22](#_Toc137846750)

[**3.3** **Розгортання розробленої системи на локальному сервері** 37](#_Toc137846751)

[4. ВИСНОВОК 39](#_Toc137846752)

[5. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 41](#_Toc137846753)

### ВСТУП

Медична індустрія є важливою складовою суспільства, а функціонування приватних клінік відіграє важливу роль у наданні медичних послуг. За останні роки спостерігається зростання кількості приватних клінік, що вимагає від них не тільки високої якості медичної допомоги, але й ефективної інформаційної системи для управління клінікою та покращення задоволення пацієнтів.

***Предметом дослідження*** є теоретичні та практичні аспекти розробки інформаційної системи приватної клініки за допомогою:

* мови розмітки HTML та CSS;
* мови програмування JavaScript, Java та її фрейворку Spring;
* мови запитів.

Інформаційна система приватної клініки повинна надавати користувачу можливість зареєструватись, записатись на прийом до лікаря та скасувати запис. Адміністратор повинен мати права доступу до всіх даних інформаційної системи та можливість редагування цих даних.

***Метою*** даної курсової роботи є розробка інформаційної системи для приватної клініки, яка допоможе підвищити ефективність роботи медичного закладу та покращити якість надання медичних послуг. Основним завданням є створення програмного забезпечення, яке включатиме в себе інтерфейс користувача, базу даних пацієнтів та медичну інформацію, засоби планування та керування прийомами, а також інші функціональні можливості, необхідні для забезпечення ефективної роботи клініки.

***Об'єктом дослідження*** є інформаційна система приватної клініки, яка включає в себе комплекс програмних та апаратних засобів, необхідних для збору, обробки та зберігання інформації про пацієнтів, лікарів, адміністраторів та записи на прийом до лікарів. Дослідження включає розробку програмного забезпечення з використанням сучасних технологій та архітектурних стилів розробки програмного забезпечення, зокрема мов програмування Java, веб-технологій (HTML, CSS, JavaScript) та бази даних MySQL. Крім того, будуть використані методи проектування бази даних та розробки веб-інтерфейсу.

***Отримані результати дослідження*** щодо інформаційної системи приватної клініки можуть бути корисні для покращення організації та управління медичним закладом. Завдяки впровадженню рекомендацій, клініки зможуть покращити якість надання медичних послуг, забезпечити ефективне планування роботи лікарів та підвищити задоволення пацієнтів.

***Використане програмне забезпечення***: для виконання роботи було використано такі технології як Java Spring Framework - для програмування серверної частини, MySQL - для створення бази даних, HTML, CSS та JavaScript для веб-інтрефейсу інформаційної системи. Використання вказаного програмного забезпечення і технічного стеку дозволить розробити функціональну та зручну інформаційну систему, що відповідає потребам приватної клініки та сприятиме поліпшенню роботи медичного заклад.

***Структруа роботи:*** Обсяг курсової роботи складатиме XX сторінок і буде містити розділи про аналіз вимог, розробку бази даних, розробку веб-додатку, висновки, список використаних джерел та додатки.

### АНАЛІЗ ВИМОГ

## **Постановка завдання**

Завдання курсової роботи на тему "Розробка інформаційної системи продажу автомобілів" включає наступні пункти:

1. Проаналізувати переваги та недоліки існуючих інформаційних систем приватних клінік.
2. Сформулювати вимоги для розробки майбутньої інформаційної системи приватної клініки.
3. Розробити базу даних для зберігання інформації про пацієнтів, лікарів, та записи на прийом.
4. Розробити веб-інтерфейс для зручної взаємодії з даними інформаційної системи для пацієнтів та адміністраторів.
5. Провести тестування розробленої інформаційної системи для виявлення та усунення можливих дефектів та оцінки ефективності системи.

Для даного проекту було надане наступне технічне завдання:

• Реєстрація нового клієнта.

• Перегляд усіх записів на прийом для певного клієнта.

• Перегляд та скасування запису на прийом до лікаря.

• Перегляд та редагування інформації про лікарів, клієнтів та їх записів за допомогою облікового запису адміністратора.

• Додавання та видалення інформації про лікарів за допомогою облікового запису адміністратора.

• Видалення інформації про клієнтів та скасування їх записів до лікарів за допомогою облікового запису адміністратора.

## **Розробка моделі варіантів використання веб-сайту**

Розроблена інформаційна система приватної клініки має 2 типи користувачів:

* Клієнт – людина яка звернулась за послугами до приватної клініки. Основними сценаріями використання інтерфейсу приватної клініки для клієнта є:
  + Реєстрація та Вхід в систему;
  + Створення записусу на прийом до лікаря на запланований час;
  + Скасування запису до лікаря.
* Адміністрато – людина яка має доступ до всіх даних інформаційної системи приватної клініки та можливість редагування цих даних. Основними сценаріями використання інтерфейсу приватної клініки для   
  адміністратора є:
  + Вхід в систему;
  + Додавання, редагування та видалення інформації про лікарів;
  + Перегляд, редагування та видалення інформації про записи на прийом;
  + Додавання, редагування та видалення інформації про клієнтів;

На рисунку 1 наведена модель використання інформаційної системи приватної клініки у вигляді Use Case діаграми:

Зображення, що містить текст, схема

Автоматично згенерований опис

Рис. 1. Use-Case діаграма основних сценаріїв використання інформаційної системи

## **Аналіз засобів реалізації (техніко-економічне обґрунтування вибору)**

Для розробки інформаційної системи приватної клініки були використані наступні технології:

* **Bootstrap**:
  + ***Готові компоненти***: Bootstrap надає багато готових компонентів, таких як кнопки, форми, таблиці, навігаційні панелі і багато інших. Це дозволяє швидко створювати стильний та сучасний вигляд веб-сторінок.
  + ***Адаптивний дизайн***: Bootstrap має вбудовану підтримку адаптивного дизайну, що дозволяє автоматично пристосовувати веб-сторінки до різних розмірів екранів, включаючи мобільні пристрої. Це забезпечує кращий досвід користувача на будь-якому пристрої.
  + ***Легкість використання***: Bootstrap має зрозумілу та просту до використання документацію. Це дозволяє розробникам швидко освоїти фреймворк і використовувати його для швидкої розробки веб-інтерфейсу.
  + ***Підтримка браузерів***: Bootstrap підтримує більшість сучасних браузерів, забезпечуючи сумісність інтерфейсу на різних платформах.
* **HTML (HyperText Markup Language):**
  + ***Стандарт веб-розробки:*** HTML є основним мовою розмітки для побудови структури веб-сторінок. Використання HTML дозволяє визначити заголовки, параграфи, списки, посилання та інші елементи, що становлять скелет веб-сторінки.
  + ***Сумісність з браузерами:*** HTML є широко підтримуваною мовою і працює на більшості сучасних браузерів. Це забезпечує однаковий вигляд веб-сторінок для користувачів незалежно від використовуваного браузера.
  + ***Легкість використання:*** HTML має простий синтаксис, що дозволяє розробникам швидко створювати і редагувати веб-сторінки. Багато редакторів коду надають підсвічування синтаксису та автодоповнення, що полегшує процес розробки.
* **CSS (Cascading Style Sheets):**
  + ***Стилізація веб-сторінок:*** CSS дозволяє визначати зовнішній вигляд елементів HTML, таких як кольори, розташування, розміри, шрифти і т.д. Завдяки CSS можна надати веб-сторінкам привабливий та професійний вигляд.
  + ***Розширюваність та підтримка:*** CSS дозволяє розробникам створювати власні стилі та шаблони, що сприяє гнучкості в дизайні веб-сторінок. Крім того, CSS має широку підтримку в сучасних браузерах, що дозволяє однаково відображати стилі на різних платформах.
  + ***Легкість управління:*** CSS дозволяє відокремити стиль від структури веб-сторінки, що робить код більш організованим і легким для редагування та підтримки.
* **JS (JavaScript):**
  + ***Динамічність:*** *JavaScript додає інтерактивність до веб-сторінок, дозволяючи взаємодіяти з користувачем, виконувати анімації, обробляти події та багато іншого. Це робить веб-інтерфейс більш живим та привабливим для користувачів.*
  + ***Широкі можливості:*** JavaScript має багато функцій та бібліотек, що дозволяють розробникам створювати складні функціональність, таку як валідація форм, асинхронні запити на сервер, маніпуляції з DOM елементами і багато іншого.
  + ***Кросс-браузерна підтримка:*** JavaScript підтримується більшістю сучасних браузерів, що забезпечує його відпрацювання на різних платформах.
* **JSP (JavaServer Pages):**
  + ***Інтеграція з Java:*** JSP дозволяє розробникам використовувати Java-код для генерації динамічного вмісту веб-сторінок. Це забезпечує широкі можливості в розробці складних інформаційних систем, включаючи доступ до баз даних, обробку форм, роботу з сесіями та інші функції Java.
  + ***Масштабованість:*** JSP дозволяє розробникам створювати багаторівневі архітектури за допомогою Java-компонентів, що сприяє зручній підтримці та розширенню інформаційної системи.
  + ***Безпека:*** JSP надає вбудовані механізми безпеки, такі як обмеження доступу до ресурсів, аутентифікація та авторизація, що робить його популярним в розробці захищених веб-додатків.

Використання цих технологій в розробці веб-інтерфейсу інформаційної системи дозволяє отримати гнучкий, стильний та функціональний дизайн, забезпечувати зручний користувацький досвід, підтримувати сумісність з різними браузерами та забезпечувати безпеку та масштабованість системи. Комбінація цих технологій дає можливість розробникам створювати потужні та ефективні інформаційні системи для різних галузей і ділянок бізнесу.

* **Java:**
  + ***Кросс-платформеність:*** Java є платформонезалежною мовою програмування, що означає, що програми, написані на Java, можуть працювати на різних операційних системах, таких як Windows, macOS та Linux. Це робить її універсальною технологією для розробки серверних додатків, які мають широке коло аудиторії.
  + ***Надійність та безпека:*** Java має вбудовану систему обробки помилок та винятків, що дозволяє розробникам легко виявляти та виправляти проблеми. Крім того, Java має високий рівень безпеки, зокрема забезпечує контроль над доступом до ресурсів, захист від вразливостей і шифрування даних.
  + ***Велика спільнота розробників:*** Java має велику та активну спільноту розробників, що означає наявність великої кількості бібліотек, фреймворків та інструментів, які полегшують розробку. Також, завдяки широкій спільноті, можна знайти відповіді на багато запитань та проблем у процесі розробки.
  + ***Масштабованість:*** Java пропонує широкий набір інструментів для розробки масштабованих додатків. Вона підтримує розподілені архітектури, многопотоковість, кластеризацію та інші механізми, що дозволяють розширювати та оптимізувати роботу системи під великі навантаження.
* **Spring Framework:**
  + ***Інверсія керування:*** Spring Framework базується на принципі Інверсії керування (IoC), що дозволяє розділити компоненти додатку і забезпечити їх легку заміну та тестування. Це сприяє зменшенню залежностей та полегшує підтримку та розширення коду.
  + ***Вбудована підтримка великої кількості функціональності:*** Spring Framework надає широкий спектр модулів та бібліотек для реалізації різних аспектів розробки, таких як веб-розробка, робота з базами даних, безпека, тестування та інші. Наприклад, Spring MVC дозволяє швидко створювати веб-додатки, а Spring Data спрощує взаємодію з реляційними базами даних.
  + ***Аспектно-орієнтоване програмування (AOP):*** Spring Framework підтримує парадигму AOP, яка дозволяє виділяти спільний функціонал (логування, транзакції, безпека) та застосовувати його до різних частин додатку без необхідності дублювання коду. Це спрощує утримання та розширення системи.
  + ***Тестування:*** Spring Framework надає розширену підтримку для тестування, включаючи можливість проведення модульних та інтеграційних тестів, підробку залежностей, мокування об'єктів та інші інструменти для забезпечення якості розробки.

Використання Java та Spring Framework для розробки серверної частини інформаційних систем дозволяє отримати надійний, масштабований та підтримуваний код. Вони забезпечують велику кількість інструментів та розширень, що спрощують розробку та підтримку системи. Крім того, Java та Spring Framework мають велику спільноту розробників, що дозволяє швидко знайти відповіді на запитання та проблеми, а також забезпечує постійний розвиток та оновлення фреймворку.

* **MySQL** є однією з найпопулярніших систем управління базами даних (СУБД) і має багато переваг для використання в інформаційних системах. Перевагами використання MySQL є:
  + ***Надійність та стабільність:*** MySQL відомий своєю високою надійністю та стабільністю. Він має добре розроблену архітектуру, яка забезпечує стійку роботу бази даних навіть при великому навантаженні. Багато великих компаній та організацій використовують MySQL для критичних застосунків, оскільки він забезпечує надійну зберігання та доступ до даних.
  + ***Простота використання:*** MySQL має простий та зрозумілий синтаксис мови запитів SQL, що полегшує розробку та оптимізацію запитів до бази даних. Розробники швидко освоюють MySQL і можуть ефективно працювати з ним для створення та управління базами даних.
  + ***Швидкодія:*** MySQL є високопродуктивною базою даних, яка забезпечує швидку обробку запитів та доступ до даних. Він має оптимізовані алгоритми та структури даних, що дозволяють ефективно опрацьовувати великі обсяги інформації. Багато розробників вибирають MySQL через його високу продуктивність та швидкодію.
  + ***Масштабованість: M***ySQL підтримує масштабованість, що дозволяє легко розширювати базу даних зростанням обсягу даних або навантаження на систему. Він може бути використаний як для невеликих проектів, так і для великих підприємств з великими обсягами даних.
  + ***Гнучкість:*** MySQL підтримує різні типи даних, включаючи текстові, числові, дата-часові, географічні та багато інших. Це дозволяє розробникам ефективно зберігати та обробляти різноманітну інформацію. MySQL також підтримує різні операції бази даних, такі як з'єднання таблиць, сортування, групування, фільтрація тощо.
  + ***Зручність управління:*** MySQL має вбудовані інструменти для адміністрування та управління базами даних. Це включає графічні інтерфейси, командний рядок, API для програмного забезпечення сторонніх розробників тощо. Ці інструменти полегшують налаштування, моніторинг та оптимізацію бази даних.
  + ***Сумісність:*** MySQL підтримує стандартні мови та протоколи, такі як SQL, JDBC, ODBC, що дозволяють його використовувати з різними мовами програмування та платформами розробки. Він може легко інтегруватись з іншими інструментами та технологіями, що робить його універсальним рішенням для багатьох проектів.

Враховуючи ці переваги, MySQL є популярним вибором для багатьох розробників та організацій, що потребують надійного та ефективного зберігання та управління даними в їх інформаційних системах. Він надає широкий спектр можливостей для створення потужних та масштабованих баз даних.

### РОЗРОБКА БАЗИ ДАННИХ

## **Модель даних**

Модель даних (також відома як схема даних або модель інформації) - це формальне представлення структури та взаємозв'язків даних в певній області знань або системі. Вона описує, як дані організовані, зберігаються і обробляються в рамках певної системи або додатку.

Модель даних може включати такі елементи, як сутності (об'єкти або поняття, про які збираються дані), атрибути (властивості або характеристики сутностей) та відношення (зв'язки між сутностями). Існує кілька типів моделей даних, включаючи ієрархічні моделі, мережеві моделі, реляційні моделі та об'єктно-орієнтовані моделі.

Реляційна модель даних є однією з найпоширеніших моделей і описує дані у вигляді таблиць, відомих як "реляції". Кожна реляція складається з набору кортежів (рядків), а кожен кортеж містить значення атрибутів (стовпців). Реляційна модель використовуєся в реляційних базах даних, які широко застосовуються у сфері розробки програмного забезпечення та управління даними.

Модель даних визначає структуру і організацію даних, а також відношення між ними. Вона може бути використана для розробки бази даних, проектування схеми бази даних, розробки програмного забезпечення, створення запитів та аналізу даних. Правильно спроектована модель даних може полегшити роботу з даними і забезпечити ефективне зберігання та обробку інформації.

Для реалізації технічного завдання було створено наступні таблиці в базі данихMySQL:

* DOCTORS: в даній таблиці зберігаються дані про лікарів які надають послуги в клініці. Сутності даної талиці мають наступні атрибути:
  + id – унікальний ідентифікатор запису(кортежу) в базі даних
  + first\_name – ім’я лікаря
  + last\_name – прізвище лікаря
  + email – робоча електронна пошта лікаря
  + phone\_number – робочий телефон лікаря
  + specialization – спеціалізація лікаря
* USERS: в даній таблиці зберігаються дані про користувачів системи. Сутності даної талиці мають наступні атрибути:
  + id – унікальний ідентифікатор запису(кортежу) в базі даних
  + first\_name – ім’я користувача
  + last\_name – прізвище користувача
  + email –електронна користувача яка використовується для входу
  + phone\_number –телефон користувача
  + pwd – пароль користувача (зберігається в базі даних в зашифрованому вигляді)
  + date\_of\_birth – дата народження
  + blood\_group – група крові
  + RH – резус фактор
  + allergy - алергія
  + user\_role – роль користувача в системі
* APPOINTMENTS: в даній таблиці зберігаються дані про записи до лікарів. Сутності даної талиці мають наступні атрибути:
  + id – унікальний ідентифікатор запису(кортежу) в базі даних
  + client\_id – унікальний ідентифвкатор клієнта
  + doctor\_id – унікальний ідентифікатор лікаря
  + date\_time – дата та час запису
  + summary –опис скарги

Зображення, що містить текст, знімок екрана, програмне забезпечення, Мультимедійне програмне забезпечення

Автоматично згенерований опис

Рис.2.1. UML-діаграма бази даних

## **Нормалізація бази даних**

Нормалізація бази даних (БД) - це процес проектування і структурування бази даних з метою усунення аномалій і неоднозначностей, що виникають при збереженні та оновленні даних. Метою нормалізації є зменшення дублювання даних, забезпечення цілісності даних та полегшення змін в структурі бази даних.

Форми нормалізації:

* Перша нормальна форма (1НФ): Ця форма вимагає, щоб кожна комірка в таблиці містила тільки атомарні значення (недільні дані), тобто значення, які не можна поділити на більш малі частини. Також вона вимагає, щоб кожен рядок у таблиці мав унікальний ідентифікатор, який можна використовувати для ідентифікації окремих записів.
* Друга нормальна форма (2НФ): Ця форма вимагає виконання усіх вимог 1НФ і додатково вимагає, щоб кожен атрибут (стовпець) у таблиці пов'язаних даних пов'язувався тільки з повною ключовою частиною первинного ключа. Загалом кажучи, ця форма спрощує структуру таблиці та забезпечує залежності між атрибутами.
* Третя нормальна форма (3НФ): Ця форма вимагає виконання усіх вимог 2НФ та додатково вимагає, щоб кожен атрибут у таблиці не залежав від інших неключових атрибутів, але тільки від первинного ключа або атрибутів, які напряму залежать від первинного ключа. Це допомагає уникнути транзитивних залежностей між атрибутами.
* Четверта нормальна форма (4НФ): Ця форма вимагає, щоб відношення в БД були вільні від многозначних залежностей. Многозначні залежності виникають, коли один набір атрибутів може мати кілька значень для одного і того ж ключа. 4НФ допомагає уникнути таких аномалій та розщеплювати таблиці для видалення зайвих множин значень.
* П'ята нормальна форма (5НФ): Ця форма також відома як нормалізація проектування баз даних (Database Design Normalization, DKNF). Вона вимагає розщеплення відношень, які мають взаємозалежні атрибути, у відношення, які містять тільки недільні дані. 5НФ спрямована на усунення зайвої складності в структурі даних та забезпечення максимальної незалежності атрибутів.

Існують також розширені форми нормалізації, такі як шоста нормальна форма (6НФ), сьома нормальна форма (7НФ) та інші, які застосовуються у випадках дуже складних або спеціалізованих баз даних. Ці форми нормалізації допомагають уникнути різних видів аномалій та забезпечують більш гнучке та ефективне управління даними.

На практиці четверту і фище форми нормалізації використовують надзвичайно рідко і для реалізації поставленого технічного завдання базу даних було приведено до 1НФ, 2НФ та 3 НФ оскільки вони сприяють ефективнішому управлінню даними, зменшують ризик втрати даних та полегшують проведення змін в структурі бази даних.

## **Типи даних**

## MySQL підтримує різноманітні типи даних, які можуть бути використані для зберігання різних видів інформації. Ось кілька основних типів даних в MySQL:

* Числові типи даних:
  + INTEGER: цілі числа.
  + FLOAT: десяткові числа з плаваючою крапкою.
  + DECIMAL: десяткові числа з фіксованою точністю.
* Рядкові типи даних:
  + CHAR: фіксована довжина рядка.
  + VARCHAR: змінна довжина рядка.
  + TEXT: для довгих текстових значень.
* Дата та час:
  + DATE: дата (рік, місяць, день).
  + TIME: час (години, хвилини, секунди).
  + DATETIME: дата та час.
* Булеві типи даних:
  + BOOLEAN або BOOL: значення true або false.

Це лише деякі з основних типів даних, які підтримуються в MySQL. Крім цього, існують інші типи даних, такі як ENUM для створення списку значень, SET для зберігання набору значень та інші. Коректний вибір типів даних дуже важливий для ефективного збереження та обробки даних в MySQL.

Для розробленої бази даних були висористані наступні типи даних для відповідних таблиць:

* DOCTORS:
  + id – int
  + first\_name – varchar(255)
  + last\_name – varchar(255)
  + email – varchar(100)
  + phone\_number – varchar(50)
  + specialization – varchar(255)
* USERS:
  + id – int
  + first\_name – varchar(255)
  + last\_name – varchar(255)
  + email – varchar(100)
  + phone\_number – varchar(50)
  + pwd – varchar(255)
  + date\_of\_birth – date
  + blood\_group – int
  + RH – char
  + allergy - varchar(255)
  + user\_role – varchar(50)
* APPOINTMENTS:
  + id – int
  + client\_id – int
  + doctor\_id – int
  + date\_time – datetime
  + summary – varchar(1000)

## **Реалізація цілісності даних**

## MySQL надає різні механізми для забезпечення цілісності даних в базі даних. Ось кілька способів реалізації цілісності даних в MySQL:

* Первинний ключ (Primary Key): Ми можемо визначити один або кілька стовпців у таблиці як первинний ключ. Первинний ключ гарантує унікальність значень у цих стовпцях і запобігає дублюванню даних.
* Зовнішній ключ (Foreign Key): Ми можемо використовувати зовнішні ключі для встановлення зв'язків між таблицями. Зовнішній ключ забезпечує зв'язок між стовпцем у одній таблиці і первинним ключем у іншій таблиці, що дозволяє контролювати цілісність даних та забезпечує відношення між ними.
* Обмеження (Constraints): MySQL дозволяє визначати різні обмеження для стовпців у таблицях. Наприклад, обмеження NOT NULL вимагає, щоб значення стовпця не було NULL, обмеження UNIQUE забезпечує унікальність значень у стовпці, а обмеження CHECK дозволяє встановлювати умови для значень стовпців.
* Транзакції (Transactions): MySQL підтримує транзакції, які дозволяють групувати кілька операцій бази даних в один блок. Це дозволяє виконувати групу операцій як єдину атомарну операцію, що забезпечує цілісність даних під час одночасних доступів та забезпечує відновлення у випадку відмови.

Ці механізми дозволяють контролювати та забезпечувати цілісність даних в MySQL. Правильне використання цих засобів допомагає уникнути аномалій, забезпечує консистентність та надійність бази даних.

Для реалізації цілісності даних в розробленій системі були використані наступні механізми у відповідних таблицях:

* DOCTORS:
  + id – PRIMARY KEY
  + first\_name – NOT NULL
  + last\_name – NOT NULL
  + email – NOT NULL UNIQUE
  + phone\_number – NOT NULL UNIQUE
  + specialization – NOT NULL
* USERS:
  + id – PRIMARY KEY
  + first\_name – NOT NULL
  + last\_name – NOT NULL
  + email – NOT NULL UNIQUE
  + phone\_number – NOT NULL UNIQUE
  + pwd – NOT NULL
  + date\_of\_birth – NULL
  + blood\_group – NULL
  + RH – NULL
  + allergy - NULL
  + user\_role – NOT NULL
* APPOINTMENTS:
  + id – PRIMARY KEY
  + client\_id – FOREIGN KEY NOT NULL
  + doctor\_id – FOREIGN KEY NOT NULL
  + date\_time – NOT NULL
  + summary – NULL

### РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ ТА ВЕБ-ІНТЕРФЕЙСУ

## **Структура проекту**

Розроблена система поділена на три основні модулі: клієнт (користувацький інтерфейс), сервер для комунікації з базою даних та клієнтом, та сама база даних.

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

Рис.3.1 Загальна структра проекту

Весь проект поділений на три основні директорії, як видно на рисунку 3.1:

* java – директорія для серверного інтерфейсу;
* resources – директорія в якій знаходиться файл application.properties. В цьому файлі вказана інформація для з’єднання з базою даних;
* webapp – директорія для файлів користувацьго інтерфейсу

## **Розробка системи**

Для швидкого функціонування системи та створення інтерфейсу для кінцевого користувача система була поділена на наступні модулі, кожен з яких відповідає за виконання певного завдання:

* серверний сервіс для комунікації та взаємодії з базою даних написаний на мові програмування Java з використанням фреймворку Spring Boot;
* реляційна база даних MySQL, яка буде зберігати інформацію користувачів обробляти запити від сервера;
* клієнт, який буде виконувати комунікацію з серверними компонентами додатку розроблений на мові HTML, CSS, JavaScript, за допомогою бібліотеки React.

Далі детальніше про реалізацію кожного з модулів додатку

**Веб-інтерфейс**

За допомогою бібліотеки Bootstrap можна створювати гнучкі та функціональні веб-аплікації. Даний сервіс системи забезпечує простий та зрозумілий інтерфейс для кінцевого користувача.

Використання Bootstrap, HTML, CSS та JavaScript у спрінгових веб-додатках має декілька переваг:

* Респонсивний дизайн: Bootstrap надає потужні інструменти для розробки респонсивного дизайну, що дозволяє створювати веб-сторінки, які працюють та виглядають добре на різних пристроях та розмірах екранів. Це особливо корисно в епоху мобільних пристроїв, де користувачі отримують доступ до веб-додатків з різних пристроїв.
* Готові компоненти: Bootstrap надає широкий набір готових компонентів, таких як кнопки, форми, меню, таблиці, модальні вікна та багато інших. Це дозволяє розробникам ефективно використовувати ці компоненти для швидкого створення інтерфейсу користувача без необхідності вирішення деталей стилізації та розміщення.
* Підтримка кросс-браузерності: Bootstrap вирішує проблеми кросс-браузерності, забезпечуючи єдність вигляду та поведінки компонентів на різних веб-браузерах. Це дозволяє розробникам писати код, який працює однаково добре на різних браузерах без необхідності вирішення специфічних проблем браузера.
* Можливість кастомізації: Bootstrap надає широкі можливості кастомізації. Розробники можуть налаштовувати вигляд компонентів, колірну схему, шрифти та інші параметри згідно з власними потребами та брендуванням. Це дозволяє створювати унікальний вигляд та ідентичність для веб-додатків.
* Підтримка мультимовності: HTML та CSS забезпечують підтримку різних мов та локалізацію. Це дозволяє розробникам створювати багатомовні веб-додатки, які можуть бути доступні для користувачів з різних країн та мовних груп.
* Можливості інтерактивності: JavaScript використовується для створення інтерактивних функцій на веб-сторінках. Він дозволяє додавати анімацію, валідацію форм, розкриваючіся панелі, обробники подій та багато іншого. JavaScript є потужним інструментом для покращення користувацького досвіду та функціональності веб-додатків.

Загалом, використання Bootstrap, HTML, CSS та JavaScript у спрінгових веб-додатках сприяє швидкому процесу розробки, респонсивному дизайну, кросс-браузерності, кастомізації та покращенню інтерактивності. Вони спільно допомагають розробникам створювати функціональні веб-додатки зі зручним інтерфейсом для користувачів.

Оскільки додаток має бути захищений від сторонніх осіб, то при спробі зайти на сайт з’являється форма для авторизації користувача. Дані для авторизації знаходяться в базі даних в таблиці USERS. Форма авторизації показана на рисунку нижче.

Зображення, що містить текст, знімок екрана, Шрифт, Бренд

Автоматично згенерований опис

Рис.3.2. Вигляд меню для авторизації користувача

Після введення логіну і пароля користувача, клієнт відправляє введені дані на сервер і в разі успішної аутентифікації відкриває головну сторінку сайту залежно від прав користувача (user/admin), яка показана на наступному рисунку.

Зображення, що містить текст, знімок екрана, Шрифт, Бренд

Автоматично згенерований опис

Рис.3.3. Вигляд головного меню адміністратора

Зображення, що містить текст, знімок екрана, Шрифт, Бренд

Автоматично згенерований опис

Рис.3.4. Вигляд головного меню користувача

Як видно на рисунку вище, при успішній авторизації відкривається доступ до відповідних функцій сайту. При невдалій спробі авторизації, буде виведене повідомлення про помилку та не дасть доступу до функцій сайту. Скріншоти сторінки сайту при хибному введенні даних показані нижче.

Зображення, що містить текст, знімок екрана, Шрифт, Бренд

Автоматично згенерований опис

Рис.3.5. Вигляд сотрінки авторизації при введенні хибного паролю

Зображення, що містить текст, знімок екрана, Шрифт, Бренд

Автоматично згенерований опис

Рис.3.6. Вигляд сотрінки авторизації при введенні хибного логіну

Також в розробленій інформаційній системі реалізована можливість реєстрації користувачів з перевіркою унікальності електроноої пошти та номеру телефоную Також реалізована перевірка паролю на відповідність патерну. Скріншоти сторінки реєстрациї наведені нижче.

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Рис.3.7. Вигляд сотрінки реєсрації при введенні існуючої електронної пошти

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Рис.3.8. Вигляд сотрінки реєсрації при введенні слабкого паролю

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Рис.3.9. Вигляд сотрінки реєсрації при введенні існуючого номеру телефону

Для клієнтів клініки реалізована можливість запису до лікарів, перегляд та скасування записів. Відповідні сторінки показані на скріншотах нижче

Зображення, що містить текст, знімок екрана, Шрифт, Веб-сторінка

Автоматично згенерований опис

Рис.3.10. Вигляд сотрінки для запису на прийом

На сторінці перегляду наявних записів реалізована можливість скасування відповідного запису до лікаря

Зображення, що містить текст, знімок екрана, Шрифт, Бренд

Автоматично згенерований опис

Рис.3.11. Вигляд сотрінки для перегляду та скасування записів до лікарів

Для адміністраторів клініки реалізовані наступні функції:

* функція додавання, видалення та редагування даних про лікарів;
* функція реадгування та видалення інформації про клієнтів з валідацією введених даних як для форми реєстрації;
* функція видалення та редагування створених записів до лікарів.

Скріншоти відповідних сторінок наведені нижче:

Зображення, що містить текст, знімок екрана, Шрифт, Бренд

Автоматично згенерований опис

Рис.3.12. Вигляд сотрінки для перегляду, редагування та видалення інформації про лікарів

Зображення, що містить текст, знімок екрана, Шрифт, Бренд

Автоматично згенерований опис

Рис.3.13. Вигляд сотрінки для додавання нових лікарів

Зображення, що містить текст, знімок екрана, програмне забезпечення, Веб-сторінка

Автоматично згенерований опис

Рис.3.14. Вигляд сотрінки для перегляду та редагування інформації про клієнтів

Зображення, що містить текст, знімок екрана, Шрифт, Веб-сторінка

Автоматично згенерований опис

Рис.3.15. Вигляд сотрінки для перегляду, редагування та видалення записів до лікарів

**Серверна частина**

Далі розгянемо «мозок» додатку, а саме серверну частину яка взаємодіє з базою даних. Серверна частина написана на об’єктно орієнтованій мові програмування Java з використанням популярного фреймворку Spring, та його підмодулів: Spring Boot, Spring Data, Spring MVC та Spring Security.

Головним завданням даного сервісу є забезпечення API для комунікації з базою даних, та виконання CRUD (Сreate, Read, Update, Delete) операцій з базою даних MySQL.

Структура серверної частини показана на рисунку нижче:

Зображення, що містить текст, знімок екрана, Шрифт, дизайн

Автоматично згенерований опис

Рис.3.16. Структура класів серверної частини проекту

Серверний інтерфейс складається з декількох частин:

* константи системи (пакет “consts”). Це пакет в якому розміщений інтерфейс який містить константні значення які використовуються в розробленій системі;
* модель бази даних (пакет “entity”). В цьому пакеті створені три класи, які визначають як будуть виглядати таблиці в базі даних. Кожен клас це окрема таблиця. Клас “Person” пов’язаний з головною таблицею бази даних, саме в ній зберігаються дані про користувачів сервісу. Клас “Doctor” пов’язаний з таблицею лікарів. Клас “Appointment” пов’язаний з таблицею записів на прийом до лікарів. В кожному з цих класів є внутрішній клас який реалізує патерн Builder. Це зроблено для спрощення та більш гнучкого підходу до процесу створення об’єктів класів.
* репозиторій (пакет repository). Цей пакет відповідає за зв'язок з базою даних. В даному пакеті є батьківський клас «DatabaseAccess» від якого наслідуються класи «AppointmentDao», «DoctorDao» та «UserDao». Дані «Dao» класи помічені анотацією @Repository. Анотація @Repository використовується в спрінгових веб-додатках для позначення класу, який виконує функції репозиторію. Репозиторій відповідає за взаємодію з базою даних або іншим джерелом даних.Основна роль класу, позначеного анотацією @Repository, полягає у виконанні операцій збереження, отримання, оновлення та видалення даних з бази даних або іншого джерела даних.@Repository є одним зі спеціальних позначень (анотацій) у Spring Framework, що дозволяє автоматичне впровадження (dependency injection) іншим компонентам додатку, які потребують репозиторіїв для доступу до даних. Використовуючи цю анотацію, спрінговий контейнер може автоматично створити екземпляр класу репозиторію та інтегрувати його в інші частини додатку;
* сервіс (пакет service). В цьому пакеті містяться всі класи, що мають   
  бізнес-логіку роботи з даними які cервер отримує з бази даних та   
  від веб-інтерфейсу;
* утиліти (пакет util). Цей пакет містить два класи. Клас «PasswordUtil» відповідає за обробку паролів отриманих як з бази даних, так і від веб-інтерфейсу. Клас «ValidationUtil» відповідає за валідацю даних отриманих з веб-інтерфейсу;
* контролер (пакет controller). Це пакет, що містить класи які приймають та відправляють запити клієнта.

В корені проекту серверної частини знаходиться клас «Application». Це клас завдяки якому запускається серверна частина та веб-інтерфейс системи.

Безпосередньо з веб-інтефейнсом взаємодіют класи в пакеті controller. Завантаження інформації з бази даних реалізоване за допомогою Dao класів, а передавання цієї інформації до веб-інтерфейсу реалізовани за допомогою методів класів контроллерів помічених анотацією @GetMapping. Коли клієнт відправляє GET-запит до сервера, спрінговий веб-додаток використовує анотацію @GetMapping для визначення методу контролера, який буде обробляти цей запит. Метод, позначений @GetMapping, повинен повертати відповідь на запит у вигляді даних або представлення, які будуть відправлені назад клієнту. Узагальнивши, можна сказати що методи помічені анотацією @GetMapping відповідають за READ операції з базою даних. Приклад методу поміченого анотацією @GetMapping наведено нижче:

Зображення, що містить текст, знімок екрана, програмне забезпечення, Мультимедійне програмне забезпечення

Автоматично згенерований опис

Рис.3.17. Приклад реалізації обробки GET запиту   
в класі «RegistrationController»

У даному випадку, GET-метод registration в RegistrationController виконує обробку запиту на отримання сторінки реєстрації.

* GET-запит до /registration викликає метод registration() у контролері.
* Метод створює об'єкт ModelAndView, який використовується для передачі даних на сторінку та відображення.
* Назва перегляду (view name) встановлюється на REGISTRATION\_VIEW, яка вказує на сторінку реєстрації.
* Об'єкт ModelAndView повертається як відповідь на запит, і це дозволяє рендерити відповідну сторінку реєстрації.

Отже, GET-метод registration просто повертає сторінку реєстрації, без обробки вхідних даних або виконання додаткової логіки. Його призначення - відображення форми реєстрації та надання користувачу можливості заповнити необхідні поля для реєстрації.

При натисканні кнопок у веб-інтерфейсі системи відправляються POST запити до серверної частини. За допомогою методів помічених анотацією @PostMapping реалізовані CREATE, UPDATE, DELETE операцій з базою даних.

Анотація @PostMapping використовується в Java для позначення методу у контролері (клас, що обробляє HTTP-запити), який має бути викликаний при HTTP POST-запиті до відповідного URL-шляху.

Коли клієнт відправляє POST-запит до сервера, спрінговий веб-додаток використовує анотацію @PostMapping для визначення методу контролера, який буде обробляти цей запит. Метод, позначений @PostMapping, повинен приймати вхідні дані з запиту і виконувати відповідні дії, такі як збереження даних у базі даних або обробка запиту.

Приклад методу поміченого анотацією @PostMapping наведено нижче::

Зображення, що містить текст, знімок екрана, меню

Автоматично згенерований опис

Рис.3.18. Приклад реалізації обробки POST запиту   
в класі «RegistrationController»

У даному випадку, POST-метод registration в RegistrationController виконує обробку запиту на реєстрацію користувача.

* Вхідні дані з запиту отримуються з об'єкту HttpServletRequest. Значення параметрів запиту, такі як "firstName", "lastName", "email" і т.д., використовуються для створення об'єкта типу Person.
* Об'єкт Person створюється за допомогою методу builder() та набору методів setFirstName(), setLastName(), і так далі, для заповнення полів об'єкта з вхідними даними.
* Після створення об'єкта Person, проводяться перевірки валідності пароля та обробка результатів реєстрації користувача.
  + Якщо пароль співпадає з повтореним паролем, викликається метод registerClient() сервісу userService для реєстрації клієнта з об'єктом Person. Отриманої відповіді (ResponseEntity) використовується для подальшої обробки.
  + Якщо реєстрація успішна (отриманий HTTP статус CREATED), повідомлення про успішну реєстрацію додається до моделі (Model) за допомогою model.addAttribute(). Крім того, метод populateModel() викликається для заповнення моделі з даними користувача.
  + Якщо реєстрація неуспішна (отриманий HTTP статус BAD\_REQUEST), повідомлення про помилку, яке міститься в тілі відповіді (responseEntity.getBody()), додається до моделі. Метод populateModel() також викликається для заповнення моделі з даними користувача.
* Якщо пароль не співпадає з повтореним паролем, повідомлення про помилку PASSWORDS\_NOT\_EQUAL додається до моделі. Метод populateModel() викликається для заповнення моделі з даними користувача.
* Після обробки запиту та оновлення моделі, повертається назва перегляду (view name) REGISTRATION\_VIEW, що вказує, яку сторінку шаблону відображати клієнту.

Отже, POST-метод registration отримує вхідні дані з запиту, створює об'єкт Person, реєструє користувача через userService, обробляє результати реєстрації та оновлює модель перед поверненням сторінки реєстрації.

## **Розгортання розробленої системи на локальному сервері**

Для розгортання системи на локальному середовищі було використано фреймворк Spring Boot оскільки він має значну кількість переваг переваг, і ось деякі з них:

* Спрощене налаштування: Spring Boot надає зручні інструменти для налаштування і автоматичного конфігурування веб-додатків. Завдяки цьому, нам не потрібно проводити важкі та складні налаштування для розгортання додатку на Tomcat. Spring Boot просто збирає наш додаток разом з вбудованим контейнером Tomcat, що полегшує його запуск.
* Вбудований контейнер сервлетів: Spring Boot включає в себе вбудований контейнер сервлетів, яким є Tomcat. Це означає, що нам не потрібно окремо встановлювати та налаштовувати Tomcat на своєму локальному комп'ютері. Ми можемо просто додати залежність в проекті Maven і Spring Boot автоматично додасть вбудований Tomcat до додатку.
* Зручний розгортання: Завдяки вбудованому контейнеру Tomcat, вам не потрібно вручну розгортати ваш додаток на сервері Tomcat. Просто запустіть ваш Spring Boot додаток, і вбудований Tomcat автоматично розгорне його на локальному сервері.

Також необхідно розгорнути базу даних MySQL для використання її в нашій системі. Для цього знадобиться виконати наступні кроки:

* Встановити MySQL: Завантажити та встановити MySQL на комп’ютер або віртуальну машину. Для цього потрібно скористатися офіційним сайтом MySQL (https://dev.mysql.com/downloads/) для завантаження відповідної версії MySQ.
* Створити базу даних: За допомогою графічного інтерфейсу (наприклад, MySQL Workbench) можна створити нову базу даних, в якій наша система буде зберігати дані.
* Налаштувати доступ до бази даних: Створити користувача бази даних та надати йому права доступу до бази даних.
* Підключіться до бази даних у вашому додатку: У нашому Spring Boot додатку підключення до бази даних налаштоване у файлі application.properties:

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

Рис.3.19. Параметри для підключення серверної частини системи до бази даних

### ВИСНОВОК

У сучасному суспільстві медична індустрія відіграє надзвичайно важливу роль у забезпеченні населення якісною медичною допомогою. Приватні клініки, зростання кількості яких спостерігається в останні роки, відіграють ключову функцію у наданні медичних послуг. Для ефективної роботи таких клінік необхідна інформаційна система, яка допоможе управляти клінікою та забезпечити задоволення пацієнтів.

Метою даної курсової роботи було розробити інформаційну систему для приватної клініки, яка підвищить ефективність роботи медичного закладу та поліпшить якість надання медичних послуг. Для досягнення цієї мети було використано такі технології, як HTML, CSS, JavaScript, Java та фреймворк Spring.

Аналізуючи функціональні вимоги до інформаційної системи, було розроблено інтерфейс користувача, базу даних пацієнтів та медичну інформацію, а також інструменти для планування та керування прийомами. Це дозволить клініці оптимізувати роботу з пацієнтами, спростити процес запису на прийом та забезпечити зручний доступ до медичної інформації.

Основною перевагою використання таких технологій, як HTML, CSS, JavaScript, Java та Spring, є їх потужність і гнучкість. Ці інструменти дозволяють розробникам створювати функціональні та естетично привабливі інтерфейси, забезпечуючи зручне взаємодію користувачів з системою. Крім того, вони забезпечують надійність, безпеку та швидкодію роботи системи.

Використання бази даних MySQL дозволяє зберігати та управляти великим обсягом даних про пацієнтів, лікарів та записи на прийоми. Це забезпечує зручний доступ до інформації, швидкий пошук та обробку даних, а також забезпечує високий рівень безпеки.

Отримані результати дослідження стосовно інформаційної системи приватної клініки можуть бути корисними для організації та управління медичним закладом. Впровадження рекомендацій, що випливають із даної роботи, дозволить поліпшити якість надання медичних послуг, забезпечити ефективне планування роботи лікарів та підвищити задоволення пацієнтів.

У ході роботи були використані такі технології, як Java Spring Framework для програмування серверної частини, MySQL для створення бази даних, HTML, CSS та JavaScript для розробки веб-інтерфейсу інформаційної системи. Застосування цього технічного стеку дозволить розробити функціональну та зручну інформаційну систему, яка повністю відповідає потребам приватної клініки та сприятиме поліпшенню роботи медичного закладу.

У підсумку, розробка інформаційної системи приватної клініки є актуальною та важливою задачею для поліпшення медичної допомоги та ефективного управління клінікою. Результати даної курсової роботи демонструють, що використання сучасних технологій програмування та баз даних є невід'ємною складовою процесу розробки інформаційних систем для медичних закладів.

### СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Віддалений репозиторій на GitHub: https://github.com/Sofia1107/private\_clinic
2. Офіційний веб-сайт Spring Boot: https://spring.io/projects/spring-boot
3. DB Designer: https://www.dbdesigner.net
4. "Database Normalization Tutorial" - My Server Name: https://uk.myservername.com/database-normalization-tutorial
5. "Data Types (Transact-SQL)" - Microsoft Docs: https://learn.microsoft.com/ru-ru/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver16
6. "Цілісність та безпека даних" - Конспект лекції   
   з організації баз даних та знань (Тема 8): https://elearning.sumdu.edu.ua/free\_content/lectured:89b3d175c06a6b137e410cb14821d0e94549ad5a/20151030211833/44700/index.html
7. W3Schools SQL Tutorial: https://www.w3schools.com/sql
8. Bootstrap Get Started - W3Schools: https://www.w3schools.com/bootstrap/bootstrap\_get\_started.asp