**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНТЕЛЕКТУАЛЬНИХ**

**ТЕХНОЛОГІЙ І ЗВ’ЯЗКУ**

Навчально-науковий інститут інфокомунікацій та програмної інженерії

Кафедра інформаційних технологій

**Курсовий проект**

з дисципліни «Організація баз даних та знань»

на тему: **«СТВОРЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ПОЛІКЛІНІКИ»**

Виконав: студент 3 курсу, групи ІПЗ-3.04 спеціальності

121 Інженерія програмного забезпечення

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Бухта М.М.

Національна шкала\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів\_\_\_\_\_\_\_Оцінка ECTS\_\_\_\_\_\_\_

**Одеса  2024**

ЗМІСТ

|  |  |
| --- | --- |
| ВСТУП |  |
| 1. ПОСТАНОВКА ЗАВДАННЯ |  |
| 2. ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ |  |
| 3. АНАЛІЗ ФУНКЦІЙНИХ ВИМОГ ТА USE-CASE ДІАГРАМИ |  |
| 4. ДІАГРАМИ ПОСЛІДОВНОСТІ ДОДАТКА |  |
| 5. ІНФОРМАЦІЙНА МОДЕЛЬ ПРЕДМЕТНОЇ ОБЛАСТІ |  |
| 6. ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ |  |
| 7. МОДЕЛЬ ПРОГРАМНОГО ДОДАТКУ |  |
| 8. СТВОРЕННЯ БАЗИ ДАНИХ |  |
| 9. ЗАПИТИ ДО БАЗИ ДАННИХ ДЛЯ РОЗВ’ЯЗАННЯ ПОСТАВЛЕНИХ ЗАДАЧ |  |
| 10. ПРОГРАМНА РЕАЛІЗАЦІЯ КЛІЄНТСЬКОГО ДОДАТКУ |  |
| 11. ПРОГРАМНА РЕАЛІЗАЦІЯ СЕРВЕРНОЇ ЧАСТИНИ |  |
| 12. ІНСТРУКЦІЯ КОРИСТУВАЧА З ІЛЮСТРАЦІЯМИ |  |
|  |  |
|  |  |
|  |  |
|  |  |

**ВСТУП**

В сучасних поліклініках електронна картотека пацієнтів є однією з найважливіших складових інформаційної системи. Ця система забезпечує збереження та організацію медичних даних пацієнтів, а також дозволяє зручно керувати інформацією про прийоми, лікування та інші медичні процедури.

Метою даного проекту є розроблена інформаційна система - електронна картотека поліклініки. Дані пацієнтів будуть зберігатися в базі даних та зчитуватися під час роботи програми.

Задачами проекту буде розробка інформаційної моделі предметної області та створення бази даних, яка забезпечить потрібною інформацією електронну картотеку

Метою курсового проекту є проектування та створення інформаційної системи для автоматизації повсякденних процесів у електронній картотеці пацієнтів поліклініки. Для досягнення цієї мети потрібно розв'язати наступні задачі:

* Проаналізувати предметну область, відокремити користувачів системи;
* виконати проектування бази даних;
* Обґрунтувати вибір засобів та технологій розробки;
* Розробити базу даних на основі системи керування базами даних PostgreSQL;
* Розробити Front-End частину застосунку;
* Розробити Back-End частину застосунку;
* Протестувати програмний застосунок, переконавшись у його коректному функціонуванні

**1. ПОСТАНОВКА ЗАВДАННЯ**

В інформаційній системі передбачено три типи користувачів:

* **Адміністратор** — відповідая за підтримку працездатності, змінює ролі, слідкує за базою, куреє співробітниками та їх розкладом;
* **Доктор** — приймає записи пацієнтів, виписує рецепти, результати прийому. Оновлює мед картку пацієнта;
* **Пацієнт** — перегляд розкладу лікарів, запис до лікаря, робить запроси у онлайн консультації.

У табл. 1.1. наведено перелік задач для кожного з користувачів із зазначенням вхідної та вихідної інформації. Схема бази даних знаходиться в додатку А.

Таблиця 1.1 — задачі користувачів.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Задача | Вхідна інформація | | | Вихідна інформація |
| 1. **Пацієнт** | | | | |
| 1.1. Перегляд лікарів | ПІБ лікаря  Спеціальність | | | ПІБ лікаря  Спеціальність  Графік роботи  Номер телефону |
| 1.2. Реєстрація | Пошта  Пароль | | |  |
| 1.3. Перегляд результатів прийомів | Ідентифікатор пацієнта | | | ПІБ лікаря  Скарги  Діагнози  Рецепт (назва ліків)  ПІБ пацієнта  Дата прийому  Статус прийому |
| 1.4. Перегляд виписаних рецептів | Ідентифікатор пацієнта | | | Назва препарату  Виробник  Рекомендації прийому |
| 1.5 Запис на прийом | Ідентифікатор співробітника  Скарги  Ідентифікатор пацієнта  Дата прийома Час прийома | | | Прийом |
| Задача | | Вхідна інформація | Вихідна інформація | |
| 1. **Лікар** | | | | |
| 2.1. Перегляд даних про пацієнта | ПІБ пацієнта | | | ПІБ пацієнта  Дата народження  Адреса  Номер телефону |
| 2.2 Перегляд результатів прийому | Ідентифікатор співробітника | | | Ідентифікатор прийому  ПІБ лікаря  Скарга  Діагноз  Рецепт  ПІБ пацієнта  Дата  Статус прийома |
| 2.3 Перегляд рецептів | Ідентифікатор рецепту | | | Номер препарату  Назва препарату  Рекомендації прийому |
| 2.4 Додавання нового рецепту | Номер препарату  Назва препарату  Рекомендації по прийому  Ідентифікатор прийому | | | Рецепт |
| 2.5 Перегляд свого робочого графіку | Ідентифікатор співробітника | | | Час початку  Час закінчення  День тижня |
| 2.6 Редагування прийому | Ідентифікатор прийому  Діагноз | | | Прийом |
| 2.7 Зміна статусу прийому | Статус = «Открыт /Закрыт/Отменен»  Діагноз  Ідентифікатор рецепту | | | Прийом |
| 1. **Адміністратор** | | | | |
| 3.1. Пошук зареєстрованих користувачів | Роль  Пошта | | | Пошта  Роль  Дата створення |
| 3.2 Редагування користувачів | Ідентифікатор користувача  Роль | | | Користувач |
| 3.3. Пошук зареєстрованого пацієнта | ПІБ пацієнта | | | ПІБ пацієнта  Дата народження  Адреса  Номер телефону |
| 3.4 Створення медичної картки | ПІБ пацієнта  Дата народження  Адреса  Номер телефонy  Ідентифікатор користувача | | | Медична картка пацієнта |
| Задача | Вхідна інформація | | | Вихідна інформація |
| 3.5 Редагування медичної картки | Ідентифікатор пацієнта  ПІБ пацієнта  Дата народження  Адреса | | | Медична картка пацієнта |
| 3.6. Редагування препаратів | Ідентифікатор препарату  Номер перпарату  Назва препарату  Виробник | | | Препарат |
| 3.7 Додавання нового співробітника | ПІБ співробітника  Спеціальність  Номер телефону  Ідентифікатор користувача | | | Медпрацівник |
| 3.8 Видалення препаратів з бази | Ідентифікатор препарату | | |  |
| 3.9 Створювати графік роботи лікарів | Лікар  День тижня прийому  Початку прийому  Кінець прийому | | | Графік |

**2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ**

Для вирішення поставлених задач була обрана **трьохланкова клієнт-серверна архітектура**, яка складається з сервера бази даних (рівень управління ресурсами), сервера додатків (рівень прикладного компоненту) та клієнтів (рівень представлення даних). Взаємодія між цими компонентами відбувається таким чином: клієнт надсилає запит на перегляд сторінки, який обробляється сервером додатків. При необхідності, сервер додатків звертається до сервера бази даних та отримує відповідь у вигляді запитуваної сторінки, яку відправляє клієнту. Схема цієї взаємодії наведена на рисунку 2.1.

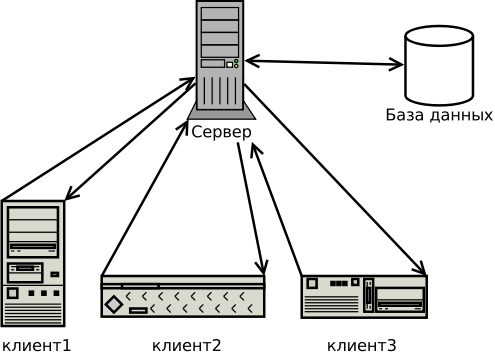


Рисунок 2.1 — схема триланкової архітектури.

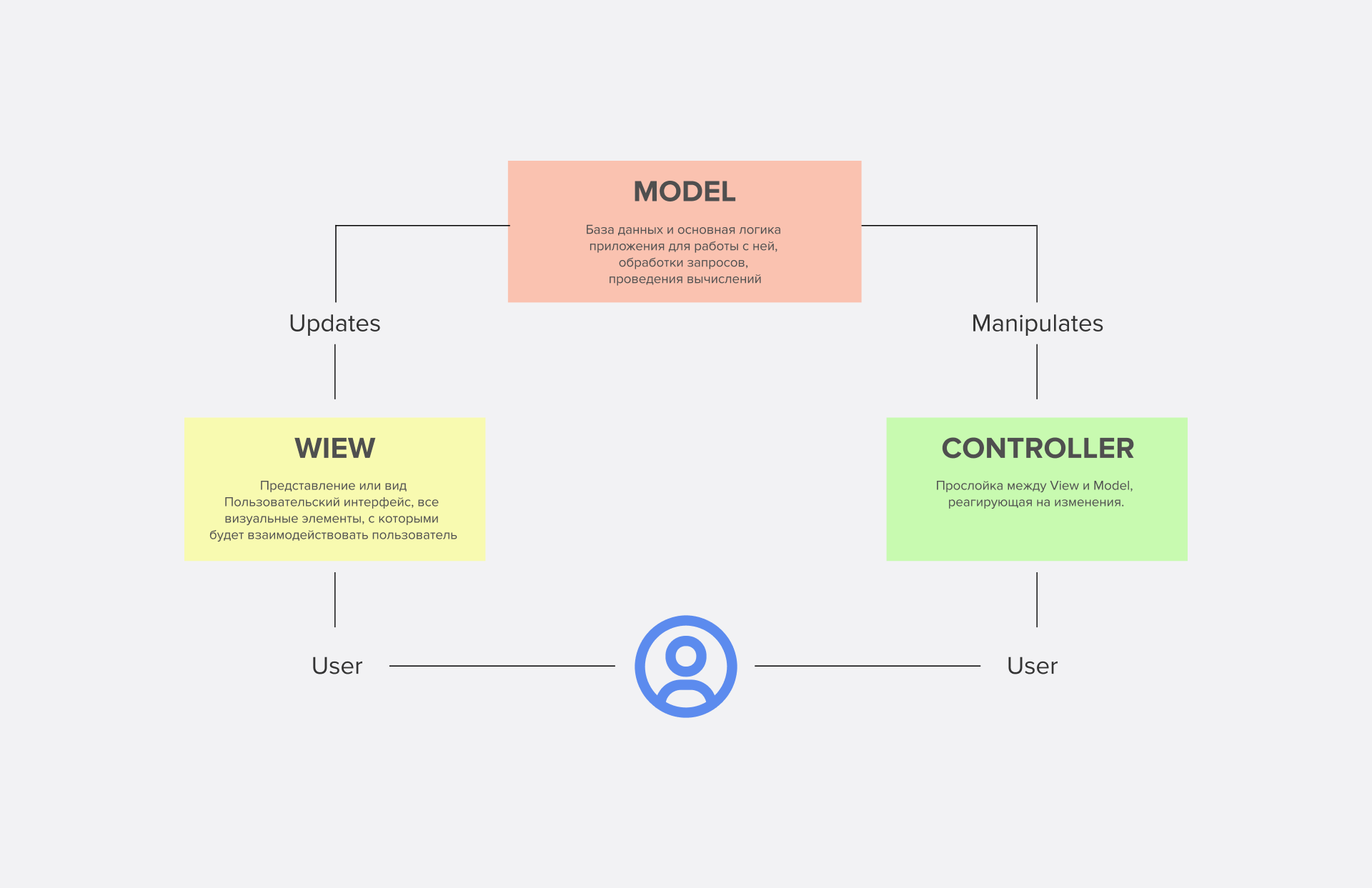
Для проектування логічної структури додатку був обраний шаблон **MVC (Model-View-Controller)**. У цьому архітектурному патерну йде переросподіл обов’язків між трьома компонентами, що дозволяє робити розробки більш гнучно та підтриємаємо.

**Model (модель)** — відповідає за управління даними. Зберегає усю мета інформацію.

**View (представлення)** — відповадіє за відображення даних користувачу. Якщо це графічна частина, то відображає графічно, якщо ми говоримо про програмну, то це, частіше усього, інтерфейс взаємодії, який можно використовувати для комунікації між іншими компонентами програми.

**Controller (контролер)** — приймає вхідні дани та обробляє їх. Тут описана основна логіка додатка та бізнес-логіка.

Схема архітектурного шаблону наведена на рисунку 2.2.

Рисунок 2.2 — схема архітектурного патерна MVC.

**Переваги MVC:**

* Розділення обов’язків — розділення на три компоненти робить код більш організованим та підтримуємим.
* Модульність — усі компаненти незалежні та можуть бути заменіми.
* Тестованість — усі компаненти можуть бути легко протестовані без залежності один від одного
* Гнучність — архітектура легко адаптується до змін вимог

Завдяки цим перевагам, шаблон MVC дозволяє швидко побудувати прототип додатку та легко вносити зміни без значних змін у вихідний код.

**3. ІНФОРМАЦІЙНА МОДЕЛЬ ПРЕДМЕТНОЇ ОБЛАСТІ**

Сутінсть та їх властивості з описом обмежень, що потрібні для розв’язання поставлених задач, наведені в таблиці 3.1.

Таблиця 3.1 — опис сутностей та їх властивостей

|  |  |  |
| --- | --- | --- |
| **Властивість** | **Опис** | **Обмеження** |
| **Об’єкт Users/Користувач** | | |
| id | Ідентифакатор | PRIMARY KEY (PK) |
| name | Ім’я | До 50 симв. NOT NULL |
| surname | Прізвище | До 50 симв. NOT NULL |
| middle\_name | По-батькові | До 50 симв. |
| email | Пошта | custom\_email, NOT NULL, UNIQUE |
| **Об’єкт Passwords/Пароль** | | |
| user\_id | Ідентифікатор користувача | PRIMARY KEY (PK), FOREIGN KEY (FK) |
| pass | Пароль | TEXT, NOT NULL |
| **Об’єкт PhoneNumber/НомерТелефону** | | |
| id | Ідентифікатор | PRIMARY KEY (PK) |
| number | Номер телефону | VARCHAR(10), NOT NULL, UNIQUE, CHECK (CHAR\_LENGTH(number) = 10) |
| country\_code | Код країни | VARCHAR(3), NOT NULL |
| user\_id | Ідентифікатор користувача | FOREIGN KEY (FK) |
| **Об’єкт Roles/Ролі** | | |
| id | Ідентифікатор | PRIMARY KEY (PK) |
| name | Назва ролі | VARCHAR(50), NOT NULL, UNIQUE |
| salary | Зарплата | DECIMAL(8, 2), CHECK (salary >= 0) |
| **Властивість** | **Опис** | **Обмеження** |
| **Об’єкт UsersRoles/РолиКористувачів** | | |
| id | Ідентифікатор | PRIMARY KEY (PK) |
| user\_id | Ідентифікатор користувача | FOREIGN KEY (FK), UNIQUE (user\_id, role\_id) |
| role\_id | Ідентифікатор ролі | FOREIGN KEY (FK) |
| **Об’єкт Messages/Повідомлення** | | |
| id | Ідентифікатор | PRIMARY KEY (PK) |
| user\_1\_id | Ідентифікатор користувача 1 | FOREIGN KEY (FK) |
| user\_2\_id | Ідентифікатор користувача 2 | FOREIGN KEY (FK) |
| destination\_time | Час відправки | TIMESTAMPTZ, DEFAULT CURRENT\_TIMESTAMP, NOT NULL |
| data | Повідомлення | До 2048 симв., NOT NULL |
| **Об’єкт Allergens/Алергени** | | |
| id | Ідентифікатор | PRIMARY KEY (PK) |
| name | Назва алергену | До 50 симв. NOT NULL |
| **Об’єкт UsersAllergens/АлергениКористувача** | | |
| id | Ідентифікатор | PRIMARY KEY (PK) |
| user\_id | Ідентифікатор користувача | FOREIGN KEY (FK) |
| allergen\_id | Ідентифікатор алергену | FOREIGN KEY (FK) |
| **Об’єкт BookedDoctors/БронюванняДоктора** | | |
| id | Ідентифікатор | PRIMARY KEY (PK) |
| patient\_id | Ідентифікатор пацієнта | FOREIGN KEY (FK), CONSTRAINT valid\_patient\_role CHECK (is\_valid\_patient\_role(patient\_id)) |
| doctor\_id | Ідентифікатор лікаря | FOREIGN KEY (FK), CONSTRAINT valid\_doctor\_role CHECK (is\_valid\_doctor\_role(doctor\_id)) |
| **Властивість** | **Опис** | **Обмеження** |
| book\_time | Час бронювання | TIMESTAMPTZ, DEFAULT CURRENT\_TIMESTAMP, NOT NULL |
| appointment\_time | Час прийому | TIMESTAMPTZ, NOT NULL |
| **Об’єкт DoctorAppointments/ПрийомУЛікаря** | | |
| id | Ідентифікатор | PRIMARY KEY (PK) |
| booked\_doctor\_id | Ідентифікатор бронювання лікаря | FOREIGN KEY (FK) |
| complaint | Скарга | До 2048 симв. |
| **Об’єкт Analyses/Аналізи** | | |
| id | Ідентифікатор | PRIMARY KEY (PK) |
| name | Назва аналізу | До 128 симв., NOT NULL UNIQUE |
| **Об’єкт AnalyseAppointments/НазначеніАналізи** | | |
| id | Ідентифікатор | PRIMARY KEY (PK) |
| analyse\_id | Ідентифікатор аналізу | FOREIGN KEY (FK) |
| doctor\_appintment\_id | Ідентифікатор прийому лікаря | FOREIGN KEY (FK) |
| is\_completed | Виконано | BOOLEAN |
| **Об'єкт PatientSickLeave/Больнічні** | | |
| id | Ідентифікатор | PRIMARY KEY (PK) |
| doctor\_appointment\_id | Ідентифікатор прийому лікаря | FOREIGN KEY (FK) |
| start\_date | Дата початку | DATE, DEFAULT CURRENT\_DATE, NOT NULL |
| end\_date | Дата закінчення | DATE, DEFAULT CURRENT\_DATE, NOT NULL |
| **Об'єкт Drugs/Ліки** | | |
| id | Ідентифікатор | PRIMARY KEY (PK) |
| **Властивість** | **Опис** | **Обмеження** |
| name | Назва препарату | До 128 симв., NOT NULL UNIQUE |
| description | Опис | До 2048 симв., NOT NULL |
| **Об’єкт PrescriptionDrugs/ВиписаніЛіки** | | |
| id | Ідентифікатор | PRIMARY KEY (PK) |
| drug\_id | Ідентифікатор препарату | FOREIGN KEY (FK) |
| doctor\_appointment\_id | Ідентифікатор прийому лікаря | FOREIGN KEY (FK) |

Між сутностямин наявні 2 типи зв’язків:

* 1:N — “один-до-бататьох”. Наявні в таблицях:
  + phone\_numbers та users
  + messages та users (user\_1\_id, user\_2\_id)
  + users\_allergen та users
  + users\_allergen та allergens
  + booked\_doctors та users (patient\_id, doctor\_id)
  + analyse\_appointments та analyses
  + prescription\_drugs та drugs
* 1:1 — “один-до-одного”. Наявні у таблицях:
  + users\_roles та users
  + doctor\_appointments та booked\_doctors
  + analyse\_appointments та doctor\_appointments
  + passwords та users
  + patient\_sick\_leave та doctor\_appointments
  + prescription\_drugs та doctor\_appointments

Для формалізації даного зв’язку первинний ключ однозв’язної сутності додається до схеми N-зв’язної сутності у якості завнішнього ключа. Схема бази даних наведена в додатку А.

**6. ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

Для роботи з базою даних обрано СКБД **PostgreSQL**, для Back-End частини використано мову програмування **C++** і бібліотеки **POSIX**, **jsoncpp**, **gtests**, **plog**, **cmake**, **pqxx** та стандартну бібліотеку шаблонів (**STL**). Для Front-End використовувалися нативні JavaScript, HTML та CSS. Весь процес розробки виконувався у текстовому редакторі **Visual Studio Code** з використанням системи контролю версій **Git**.

Сьогодні існує багато систем керування базами даних (Oracle, MongoDB, PostgreSQL та інші). Для **реалізації бази даних** цієї системи обрано діалект PostgreSQL з наступних причин:

* Об'єктно-реляційна модель даних, що використовується в PostgreSQL, є зрозумілою для кінцевого користувача. Це дозволяє зручно працювати з даними і зробити систему більш доступною для користувачів.
* Має гнучкий механізм управління правами користувачів бази даних за допомогою ролей. Це дозволяє точно налаштувати доступ до даних для різних користувачів і забезпечити безпеку інформації.
* Мова програмування plpgSQL, яка є розширенням стандартного SQL, підтримується в PostgreSQL. Вона дозволяє створювати ефективні збережені процедури і спрощує програмування логіки обробки даних.
* Постійно підтримується розробниками, що гарантує наявність оновлень, виправлення помилок та підтримку у майбутньому.

Back-End частина була розроблена з використанням мови програмування C++ і бібліотек POSIX, jsoncpp, gtests, plog, cmake, pqxx та стандартної бібліотеки шаблонів (STL). Це дозволяє створювати високопродуктивні та надійні серверні додатки.

Основні можливості та бібліотеки **Back-End:**

* POSIX: Працює на UNIX системах і забезпечує прямий доступ до команд системи. Це дозволяє значно економити ресурси і підвищує швидкість обробки та передачі даних.
* jsoncpp: Використовується для роботи з форматом даних JSON, що дозволяє легко парсити та генерувати JSON-дані.
* gtests: Бібліотека для створення та виконання модульних тестів, що забезпечує високу якість коду.
* plog: Легковага бібліотека для логування, яка дозволяє відстежувати події та помилки під час виконання програми. Являється базою для собственого логера.
* cmake: Система автоматичного складання, що дозволяє легко керувати процесом компіляції та налаштуванням проекту.
* pqxx: C++ бібліотека для роботи з PostgreSQL, що забезпечує зручний інтерфейс для виконання SQL-запитів та роботи з базою даних.

**Front-End** частина була розроблена з використанням нативних JavaScript, HTML та CSS, що дозволяє створювати інтерактивні і привабливі веб-інтерфейси. HTML використовується для структурування веб-сторінок, CSS для стилізації елементів, а JavaScript для додавання динамічної поведінки та взаємодії з сервером.

Для розробки використовувався текстовий редактор Visual Studio Code, що забезпечує зручне середовище для написання коду з підтримкою великої кількості плагінів та розширень. Система контролю версій Git використовувалася для управління змінами коду, що дозволяє відстежувати історію змін, співпрацювати з іншими розробниками і забезпечувати резервне копіювання коду.