**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ТОРГОВЕЛЬНО-ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ**

**Кафедра інженерії програмного забезпечення та кібербезпеки**

**Практична підготовка**

**ЗВІТ  
з індивідуального завдання**на тему: «Розробка мобільного додатку для клієнтів спортзалу «YOurGym»»

**Виконала:** студентка групи 3-4 ФІТ

Авєріна Наталія Ігорівна

**Перевірили:** Криворучко О.В. та

Десятко А.М

**Київ – 2023**

**Мобільний додаток для клієнтів спортзалу «YOurGym»**

1. ***Загальні відомості***

Тема: «Мобільний додаток для клієнтів спортзалу»

Предметна область: тренажерний зал.

1. ***Мета та призначення створення програми***
   1. Призначення програми

Спростити та оптимізувати взаємодію клієнтів із спортзалом, надавши їм інструмент для управління особистими даними та доступом до тренажерного залу.

* 1. Мета створення програми

Забезпечити користувачам зручний доступ до інформації та сервісів спортзалу через мобільний інтерфейс, підвищити задоволеність клієнтів і збільшити їх лояльність до клубу.

* 1. Визначення потреб майбутніх користувачів

Основними потребами користувачів є:

* Доступ до спортзалу за допомогою QR-коду в додатку
* Можливість призупинення дії абонементу
* Відстеження активності
* Доступ до інформації про тренерів та абонементи
* Доступність та зручність додатку

1. ***Вимоги до програми***
   1. Вимоги до програми в цілому

Функціональні вимоги:

* Авторизація користувачів
* Інтерактивний календар з історією відвідувань та занять
* Управління тривалістю абонемента
* Безпека даних та конфіденційність інформації

Нефункціональні вимоги:

* Висока продуктивність та швидкість відгуку додатку.
* Легкість використання і доступність інтерфейсу.
* Масштабованість та можливість адаптації до зростання кількості користувачів.
  1. Вимоги до структури та функціонування програми

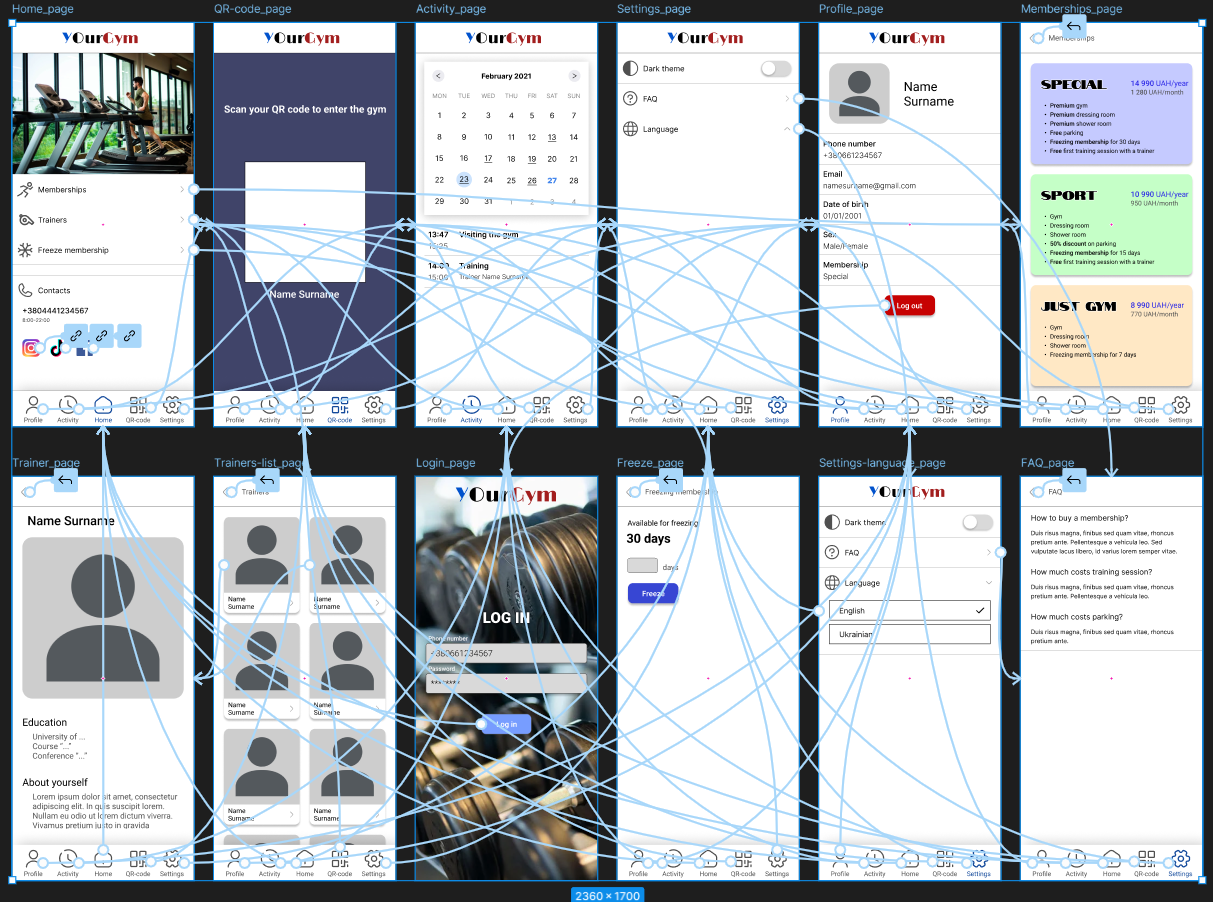
Макет мобільного додатку

На головну сторінку (*Home*) користувач потрапляє після входу в систему (*Login\_page*) за допомогою номеру телефону та паролю.

З домашньої сторінки можна потрапити на сторінку зі списком абонементів (*Memberships*), заморозити абонемент (*Freeze membership*) та переглянути список тренерів (*Trainers*).

Мобільний додаток, окрім домашньої, має ще 4 основних сторінки:

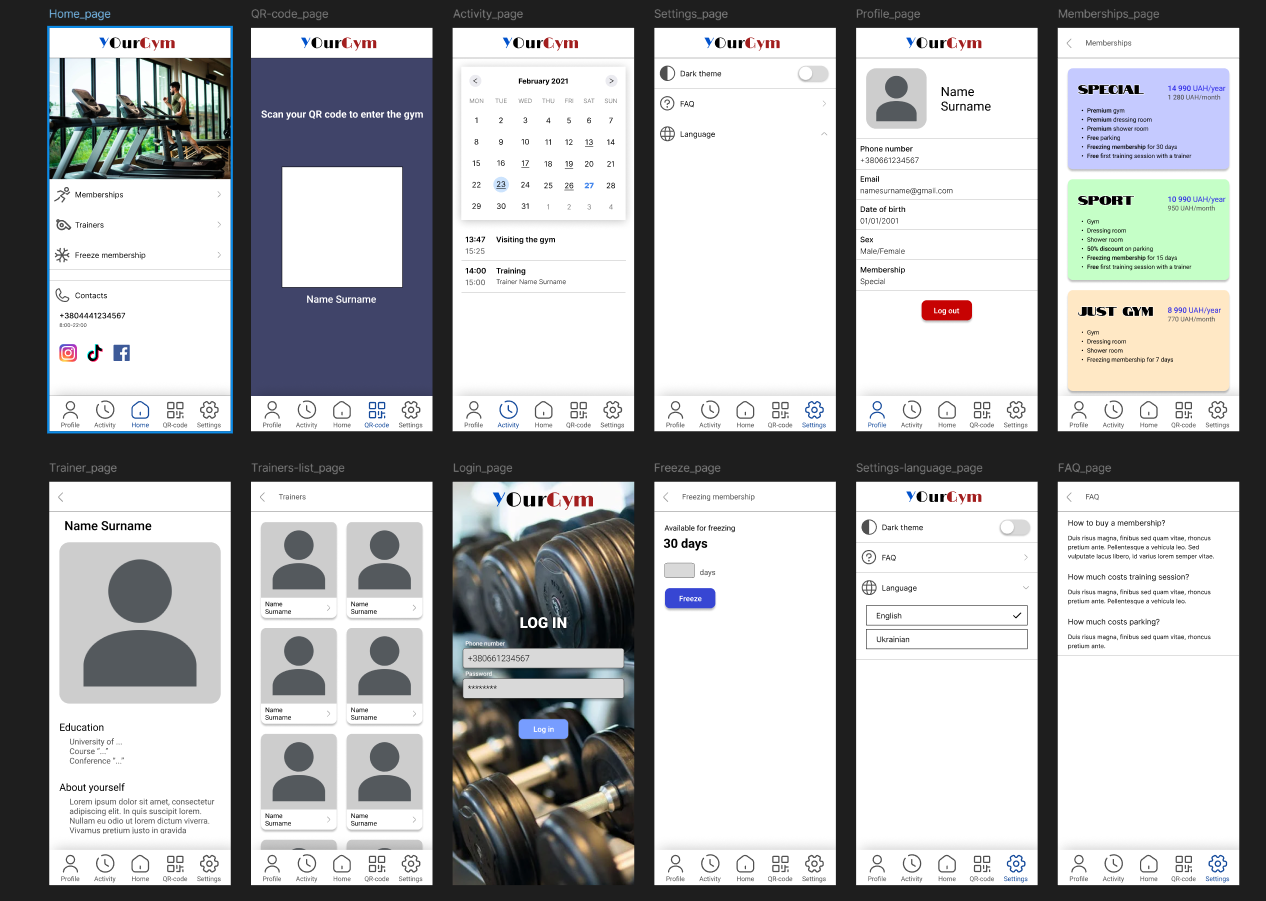
1. сторінка активності користувача (*Activity*);
2. профіль користувача (*Profile*);
3. налаштування додатку (*Settings*);
4. сторінка з QR-кодом (*QR-code*).

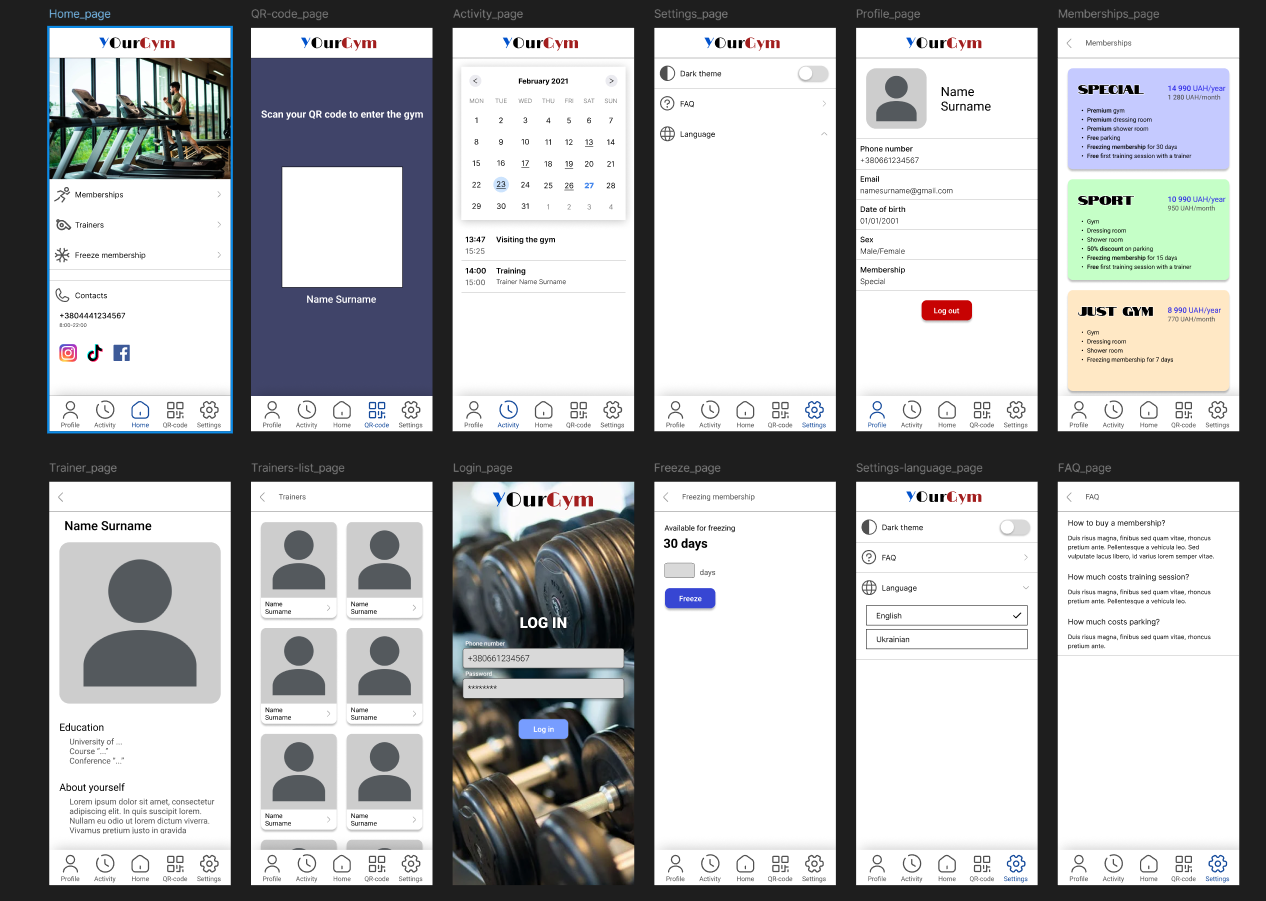


Дизайн мобільного додатку

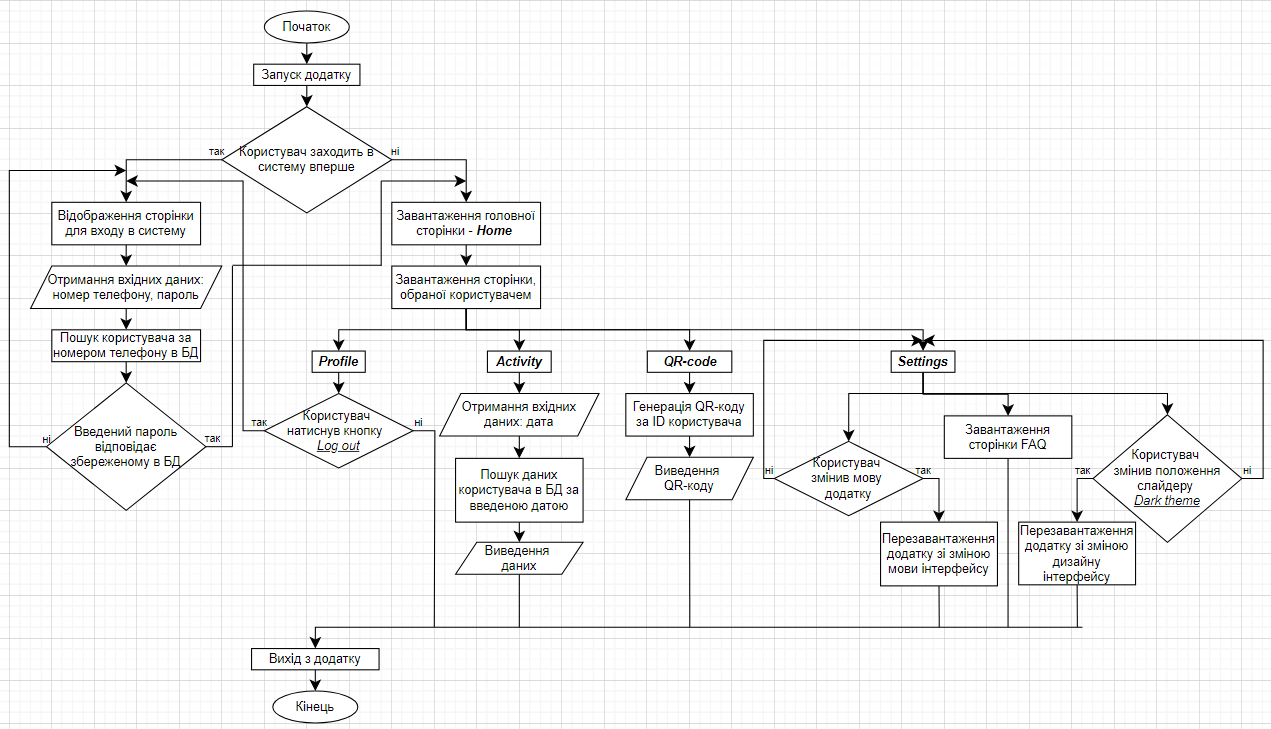
Додаток має мінімалістичний дизайн з використанням спокійних кольорів та чіткого шрифту (*Roboto*), що має забезпечити легкість сприйняття користувачем.

Інтерфейс інтуїтивно зрозумілий, кожна кнопка має стандартні іконки, які звичні користувачам будь-якої вікової категорії.

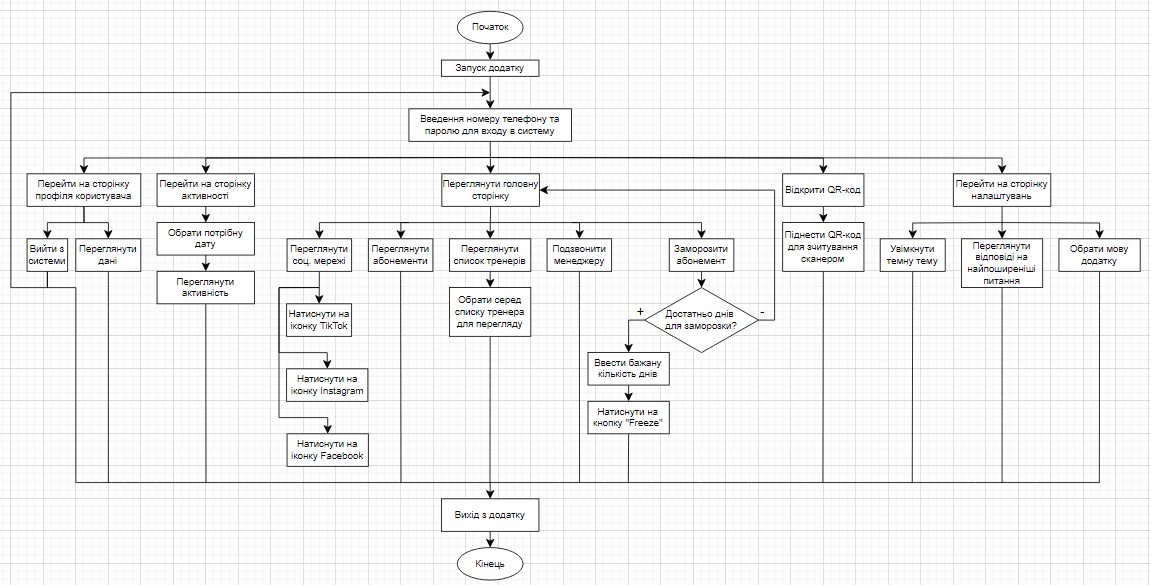




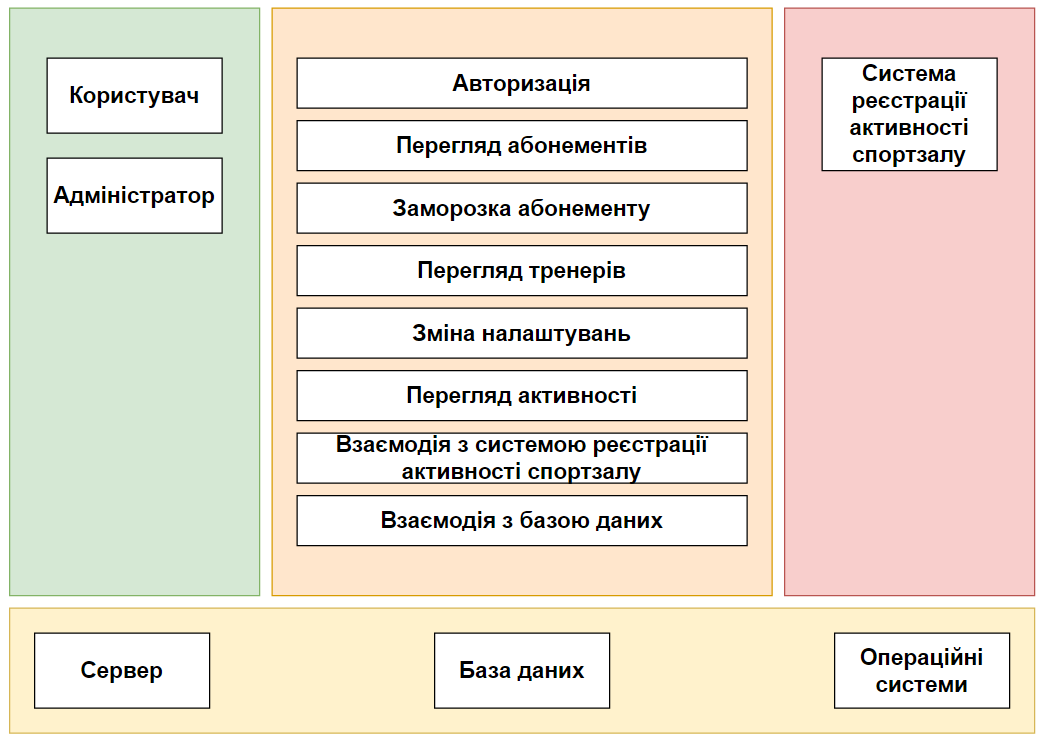
Блок-схема алгоритму роботи програми з точки зору внутрішніх процесів



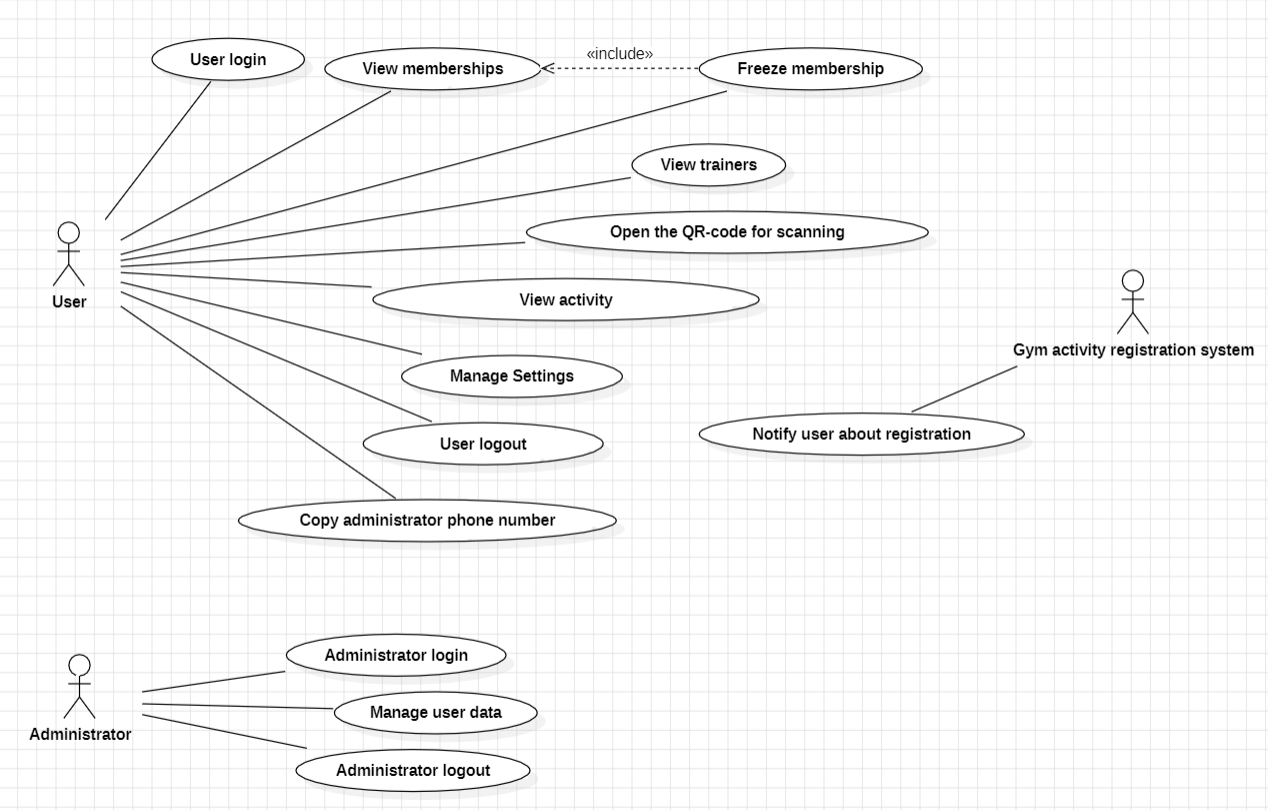
Блок-схема алгоритму роботи програми з точки зору користувача



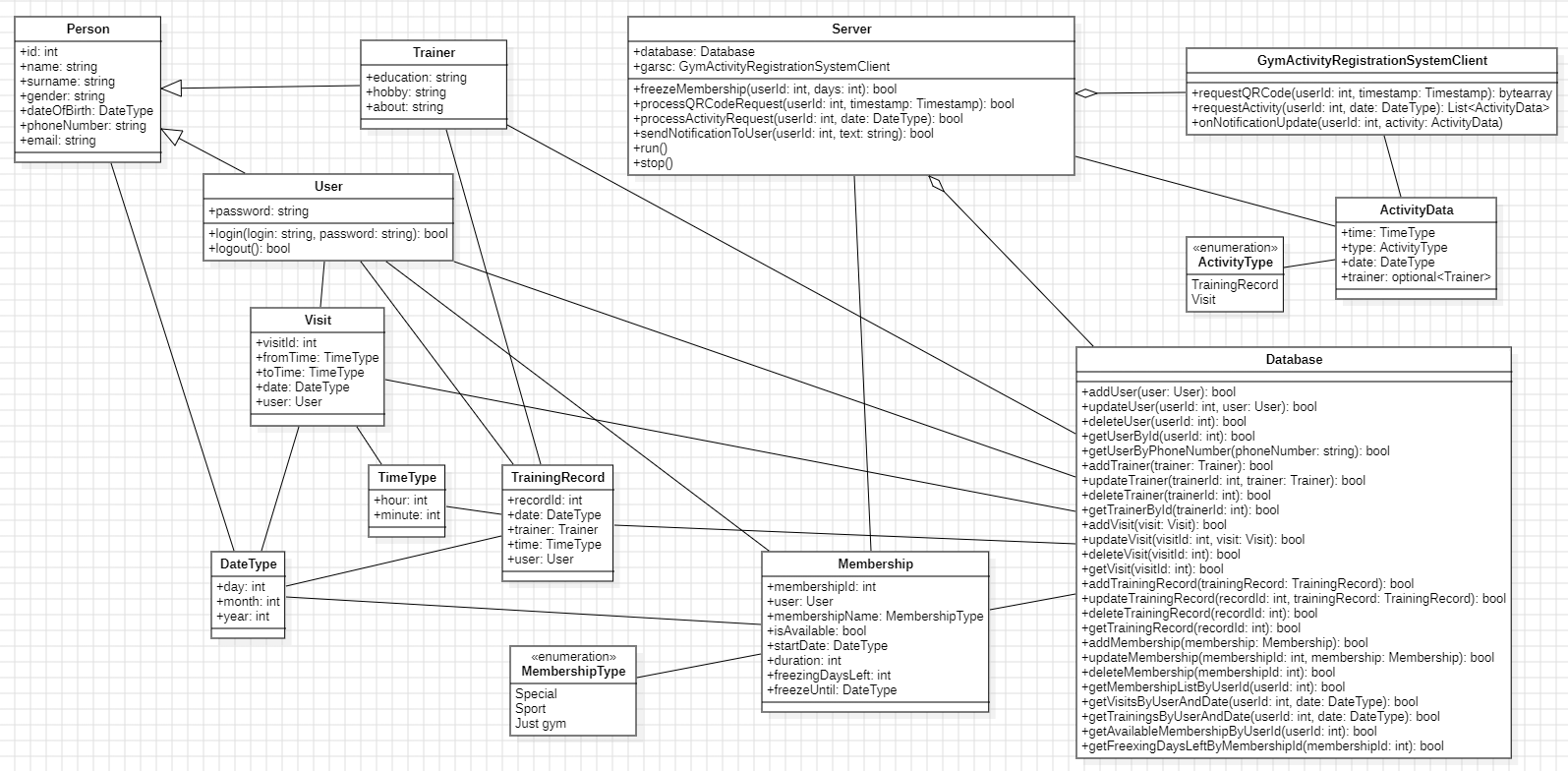
Концептуальна модель програми



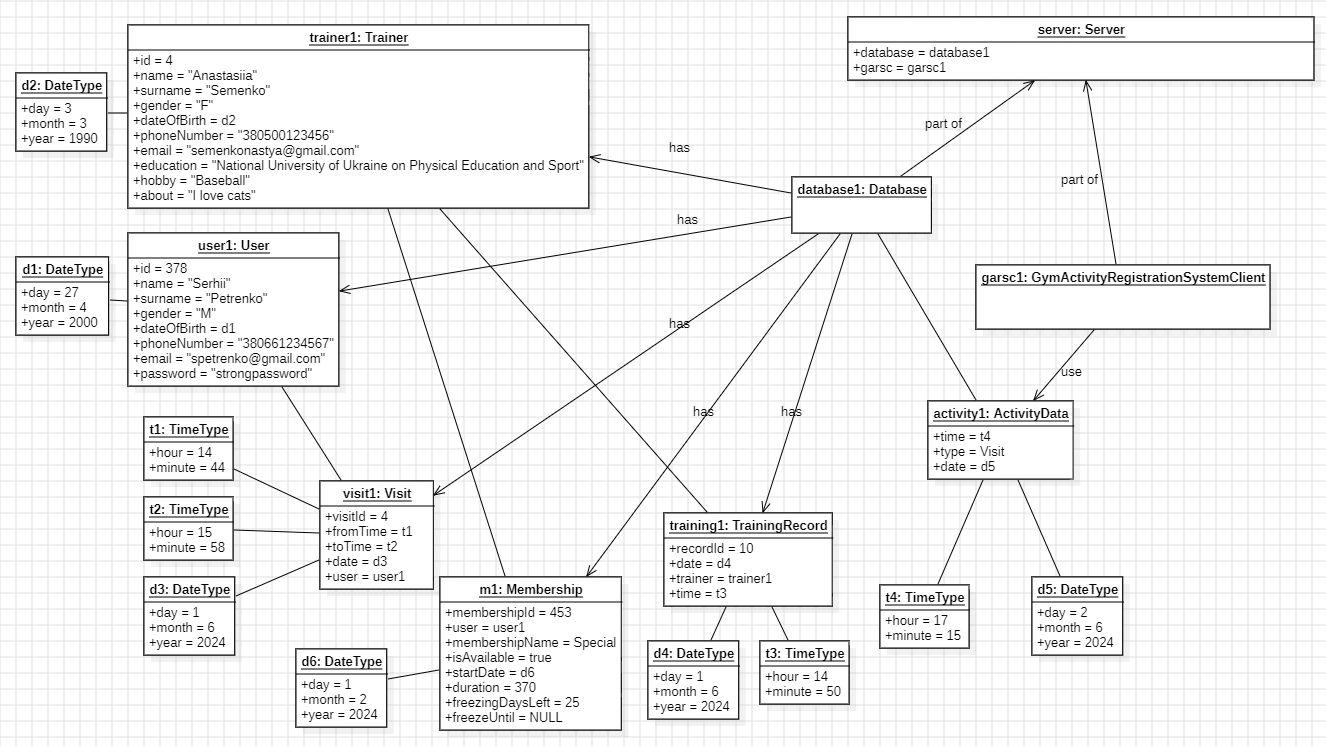
Діаграма варіантів використання



Діаграма класів



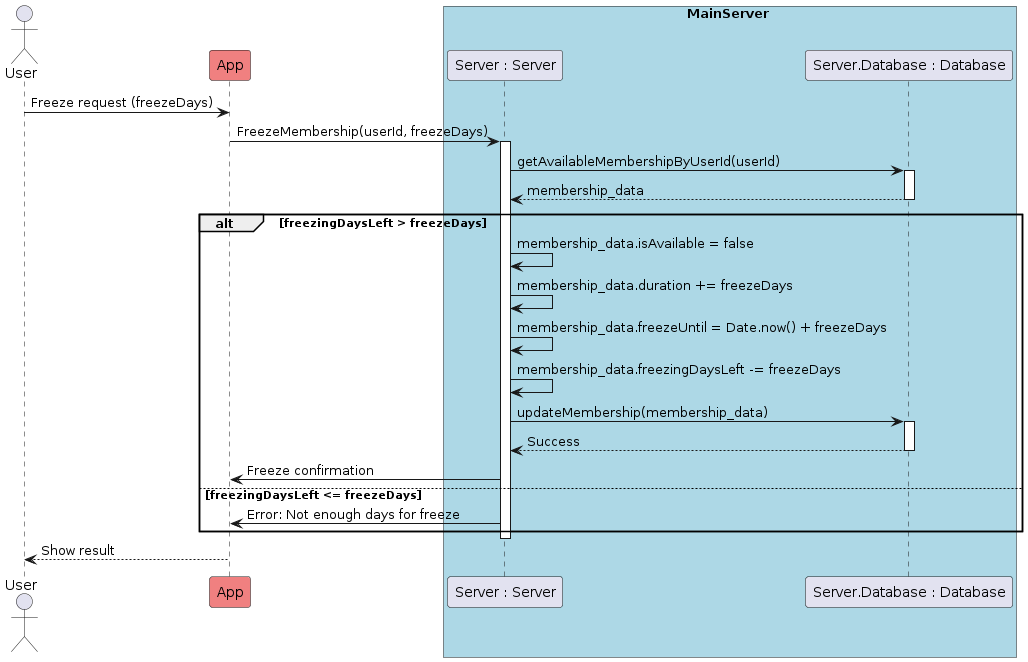
Діаграма об’єктів



Діаграми послідовностей

Для того щоб зобразити часовий графік «життя» об’єктів, зв’язаних між собою для реалізації варіанту використання чи виконання завдання (в нашому випадку – реалізації методів серверу), потрібно створити для кожного діаграму послідовностей.

Наступна діаграма ілюструє процес заморожування абонементу (метод класу Server – *freezeMembership(userId: int, days: int): bool*)

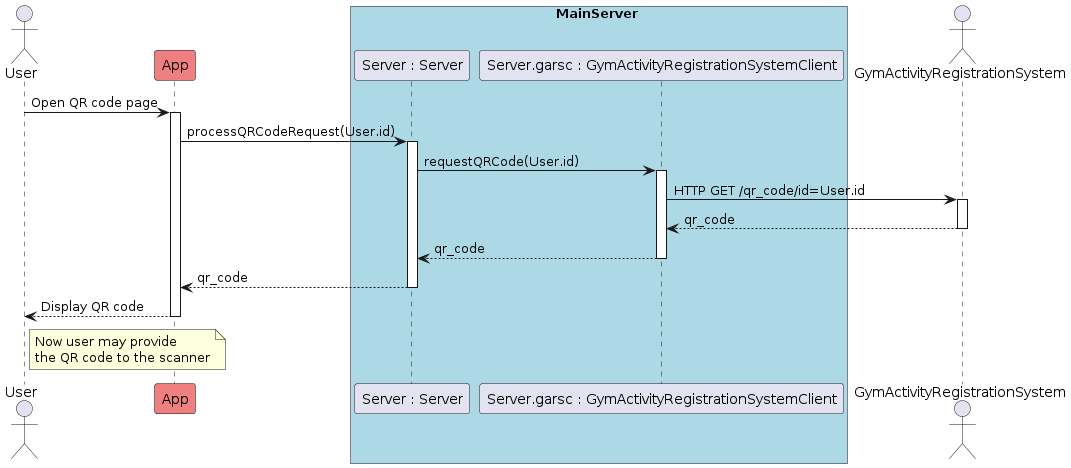


Користувач ініціює запит на заморожування абонементу через мобільний додаток, передаючи кількість днів, на які він бажає заморозити свій абонемент. Мобільний додаток відправляє запит FreezeMembership з ID користувача та кількістю днів для заморозки до cервера (Server). Сервер запитує у бази даних (Database) інформацію про абонемент користувача. База даних повертає дані абонементу (membership\_data) назад до сервера.

Сервер аналізує отримані дані. Якщо доступних днів для заморозки більше, ніж запитана кількість днів, то значення доступності абонементу стає false, тривалість абонементу збільшується на вказану кількість днів, зменшується доступна кількість днів для заморозки та встановлюється дата завершення заморозки. Сервер відсилає оновлені дані в БД для збереження, отримує відповідь про успіх і надсилає підтвердження заморозки в додаток.

Якщо доступних днів недостатньо, сервер надсилає в додаток повідомлення про помилку. Результат передається користувачу.

Діаграма послідовностей виконання методу *processQRCodeRequest(userId)* виглядає так:

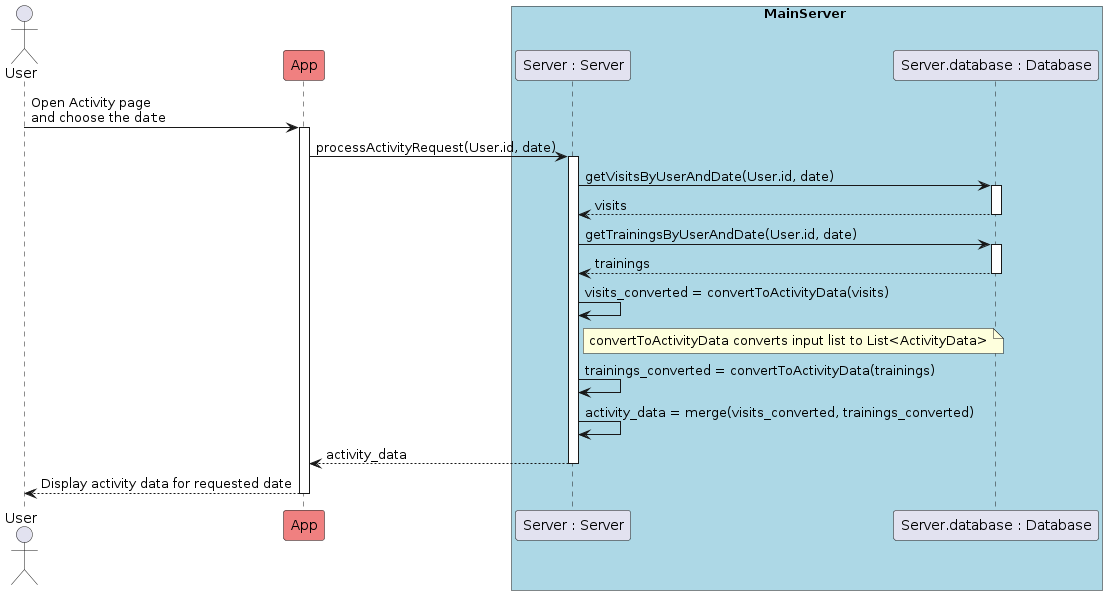


Користувач (User) запускає сторінку QR-коду в мобільному додатку (App). Мобільний додаток відправляє запит *processQRCodeRequest(userId)* до сервера (Server). Сервер перенаправляє запит на отримання QR-коду до компонента GymActivityRegistrationSystemClient(gars\_client) через метод *requestQRCode(userId)*. GymActivityRegistrationSystemClient (gars\_client) виконує HTTP GET запит до зовнішньої системи GymActivityRegistrationSystem(gars), використовуючи URL */qr\_code/id=userId.*

GymActivityRegistrationSystem (gars) обробляє запит та повертає QR-код до GymActivityRegistrationSystemClient (gars\_client). GymActivityRegistrationSystemClient (gars\_client) передає QR-код назад до Server, який, у свою чергу, передає його мобільному додатку.

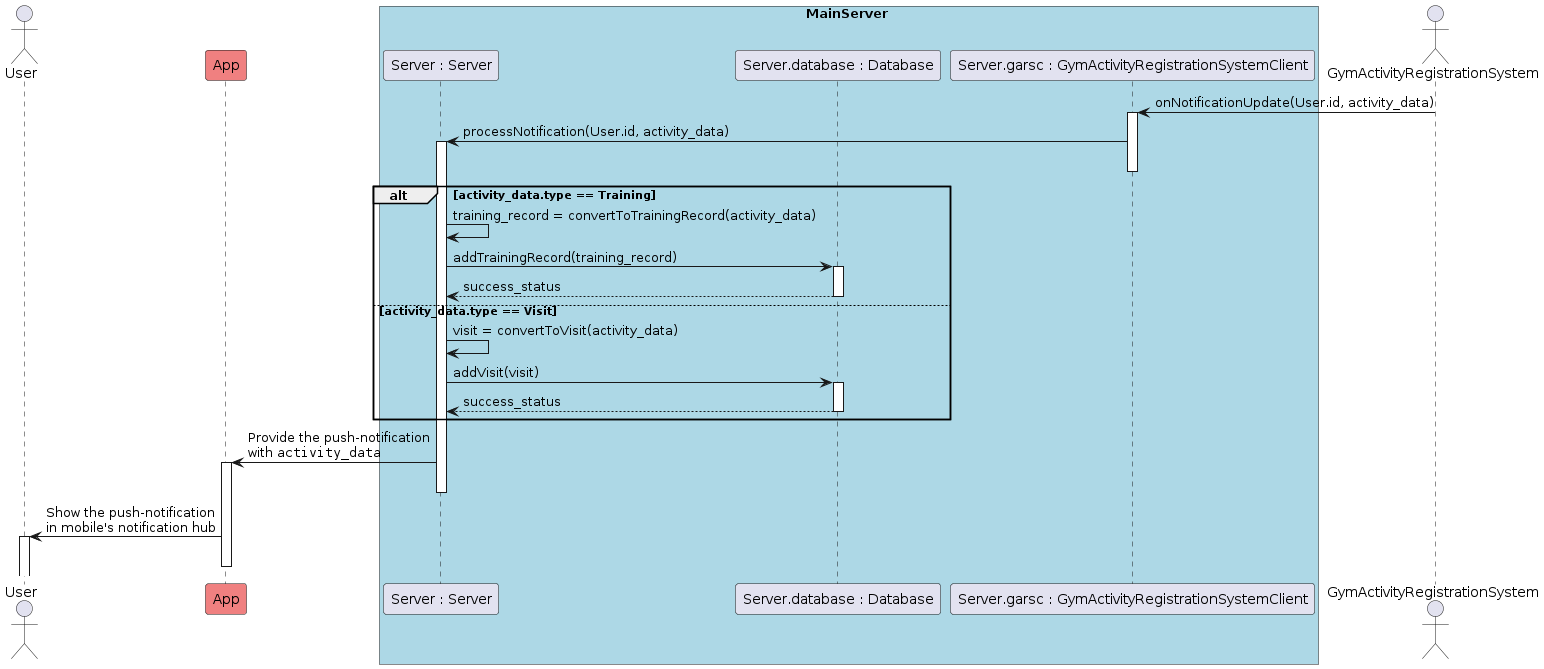
Мобільний додаток відображає QR-код на екрані, дозволяючи користувачу використовувати його.

Для відображення реалізації методу *processActivityRequest(userId, date)*, була розроблена наступна діаграма послідовностей:



Користувач відкриває сторінку активності в мобільному додатку і обирає конкретну дату, для якої бажає переглянути його минулі відвідування залу чи тренування. Мобільний додаток надсилає запит *processActivityRequest(userId, date)* на сервер. Сервер взаємодіє з базою даних для отримання списку відвідувань (visits) і тренувань (trainings) для даного користувача на зазначену дату через методи. Сервер перетворює отримані списки відвідувань та тренувань у формат даних активностей (ActivityData) за допомогою методу *convertToActivityData*. Далі Сервер зливає конвертовані дані в один загальний список activity\_data. Отримані і оброблені дані активностей надсилаються назад у мобільний додаток, який відображає їх користувачу.

Діаграма послідовностей виконання методу *onNotificationUpdate(userId, activity\_data)*:



GymActivityRegistrationSystem ініціює процес, відправляючи оновлення про активність через метод *onNotificationUpdate(userId, activity\_data)* до клієнта системи (GymActivityRegistrationSystemClient (gars\_client)). gars\_client передає отримані дані активності до основного сервера.

Сервер виконує перевірку типу активності:

* якщо це тренування (Training), дані конвертуються в запис тренування (training\_record). Потім цей запис додається до бази даних.
* якщо це візит (Visit), дані конвертуються в запис візиту (visit). Цей візит додається до бази даних.

Server після обробки даних ініціює відправлення пуш-нотифікації до мобільного додатку. Додаток відображає повідомлення у сповіщеннях мобільного пристрою.

Діаграма компонентів

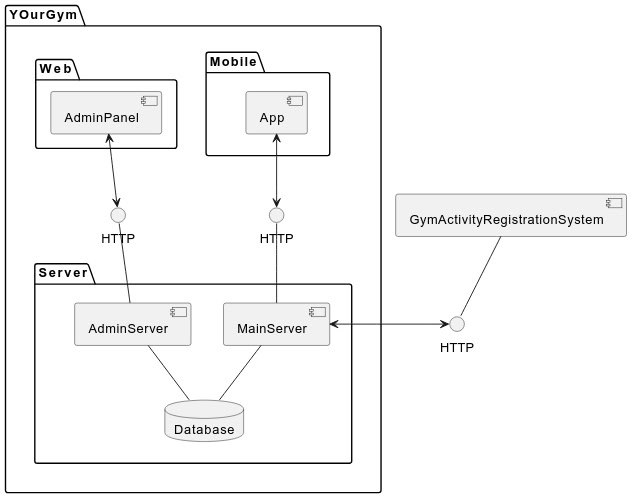
Компоненти:

* Web (AdminPanel) – це веб-інтерфейс для адміністраторів спортзалу, який дозволяє управляти даними користувачів, тренерів та спортзалу.
* Mobile (App) – це мобільний додаток, який надає клієнтам спортзалу доступ до послуг.
* GymActivityRegistrationSystem – це компонент відповідальний за реєстрацію відвідувань та тренувань в тренажерному залі. Він забезпечує інтеграцію активностей з основною системою.

Серверна інфраструктура:

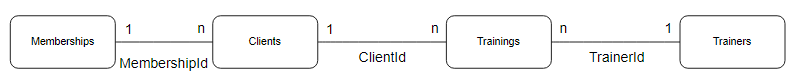
* AdminServer обслуговує запити від панелі адміністратора.
* MainServer обробляє запити від мобільного додатку користувачів.
* Database - центральна база даних, яка зберігає всю інформацію, необхідну для функціонування додатків.

Всі компоненти спілкуються через HTTP, стандартний протокол передачі гіпертексту.

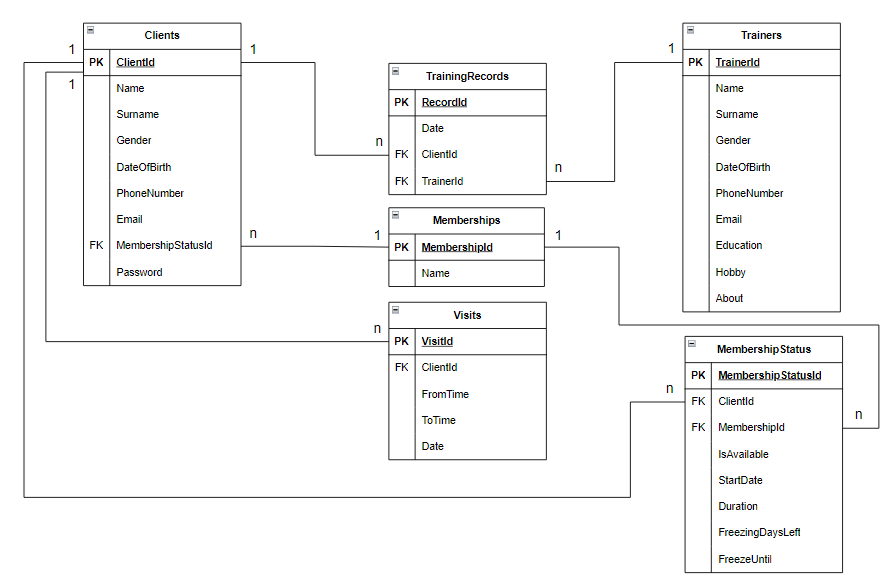


1. ***Архітектура бази даних***

Концептуальна модель



Логічна модель БД



Фізична модель БД

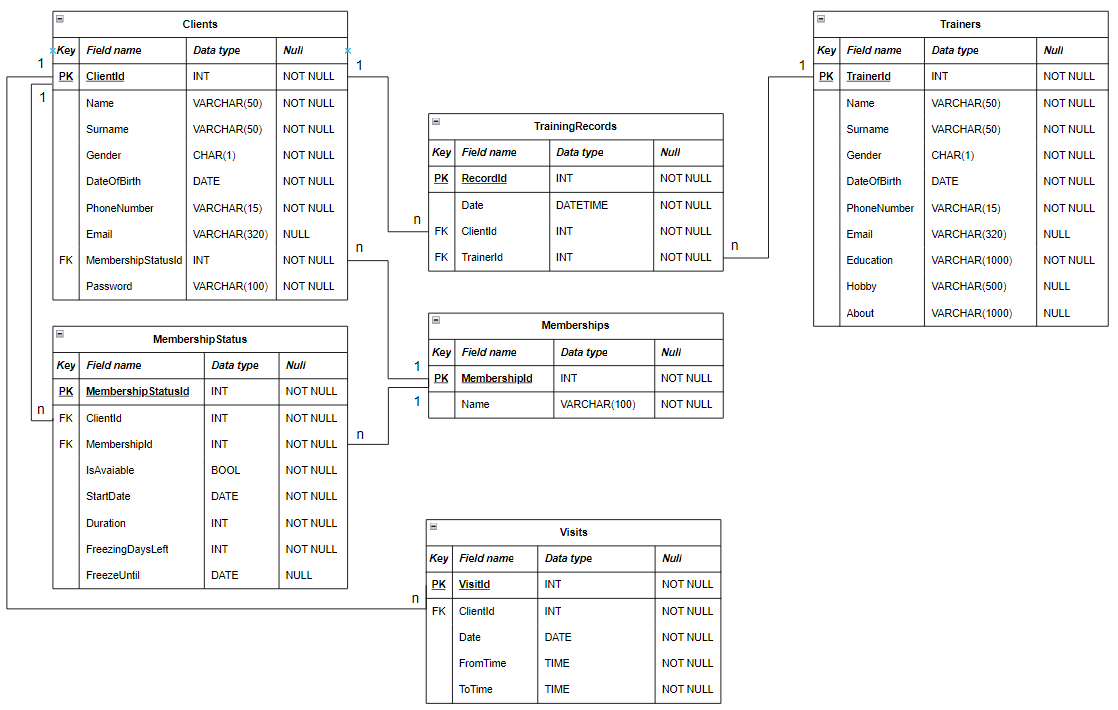
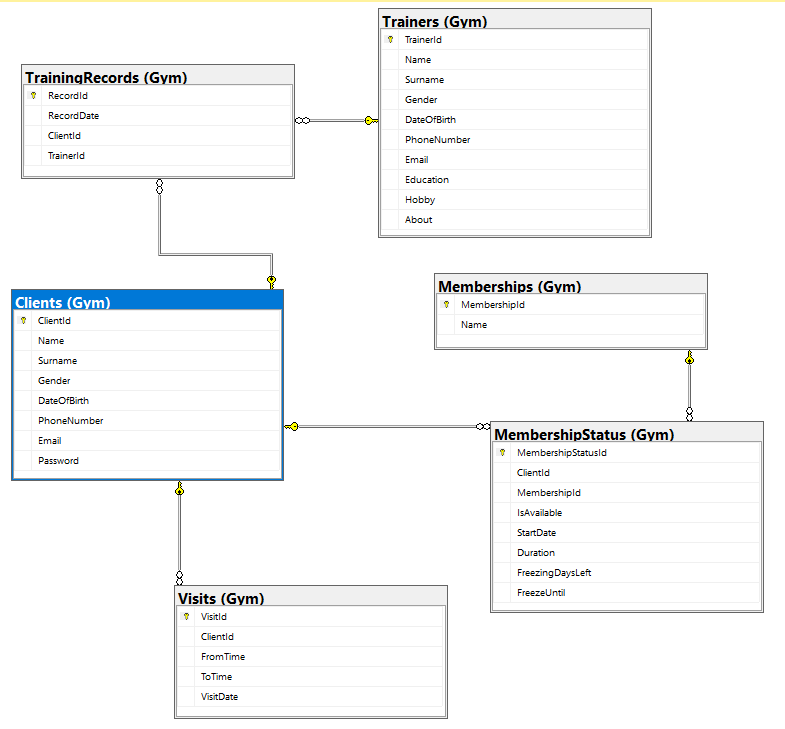


Схема БД створена в MSSQL (скрипт заповнення таблицями БД в Додатку)



**Висновки**: під час виконання індивідуального завдання в рамках практичної підготовки було закріплено практичні навички у розробці архітектури ПЗ, а саме створення алгоритмів та UML-діаграм. Також було розроблено дизайн-проєкт та прототип ПЗ, базу даних системи.

**Додатки**

Скрипт створення таблиць БД

USE GymDB;

CREATE TABLE Gym.Memberships

(MembershipId INT PRIMARY KEY IDENTITY (1,1),

Name VARCHAR(100))

ON InfrequentlyAccessedGroup;

CREATE TABLE Gym.Clients

(ClientId INT PRIMARY KEY,

Name VARCHAR(50),

Surname VARCHAR(50),

Gender CHAR(1),

DateOfBirth DATE,

PhoneNumber VARCHAR(15),

Email VARCHAR (320) NULL,

Password VARCHAR(100))

ON FrequentlyAccessedGroup;

CREATE TABLE Gym.Trainers

(TrainerId INT PRIMARY KEY,

Name VARCHAR(50),

Surname VARCHAR(50),

Gender CHAR(1),

DateOfBirth DATE,

PhoneNumber VARCHAR(15),

Email VARCHAR (320) NULL,

Education VARCHAR(1000),

Hobby VARCHAR(500) NULL,

About VARCHAR(1000) NULL)

ON FrequentlyAccessedGroup;

CREATE TABLE Gym.TrainingRecords

(RecordId INT PRIMARY KEY IDENTITY(1,1),

RecordDate DATETIME,

ClientId INT FOREIGN KEY REFERENCES Gym.Clients,

TrainerId INT FOREIGN KEY REFERENCES Gym.Trainers)

ON InfrequentlyAccessedGroup;

CREATE TABLE Gym.Visits

(VisitId INT PRIMARY KEY IDENTITY(1,1),

ClientId INT FOREIGN KEY REFERENCES Gym.Clients,

FromTime TIME,

ToTime TIME,

VisitDate DATE)

ON FrequentlyAccessedGroup;

CREATE TABLE Gym.MembershipStatus

(MembershipStatusId INT PRIMARY KEY IDENTITY(1,1),

ClientId INT FOREIGN KEY REFERENCES Gym.Clients,

MembershipId INT FOREIGN KEY REFERENCES Gym.Memberships,

IsAvailable BIT,

StartDate DATE,

Duration INT,

FreezingDaysLeft INT,

FreezeUntil DATE NULL)

ON FrequentlyAccessedGroup;