ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр
на тему: «**РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ОСОБИСТИХ СПОРТИВНИХ ТРЕНУВАНЬ МОВОЮ C#»**

Виконав: студент 4 курсу, групи ПД–41
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

Луценко І.В.
(прізвище та ініціали)

Керівник Гаманюк І.М
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Бакалавр»
Напрям підготовки -121 - Інженерії програмного забезпечення"

ЗАТВЕРДЖУЮ

Завідувач кафедри
Інженерії програмного забезпечення
О.В Негоденко
—___‖_____2023 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

ЛУЦЕНКА ІВАНА ВІКТОРОВИЧА
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка програмного забезпечення для моніторингу особистих спортивних тренувань мовою C#»

Керівник роботи Гаманюк І.М., старший викладач кафедри
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від — «2 4 » лютого 2023 року №26.

2. Строк подання студентом роботи 01 червня 2023 року
3. Вхідні дані до роботи:

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).

4.1 Аналіз наявних засобів та технологій для моніторингу спортивних тренувань
4.2 Вимоги та оцінка якості системи
4.3 Вибір засобів розробки
4. 4 Розробка програмного продукту

5. Перелік графічного матеріалу

1 Мета, об'єкт та предмет дослідження
2. Задачі роботи
3. Аналіз аналогів
4. Вимоги до програмного забезпечення
5. Програмні засоби розробки
6. Діаграма прецедентів
7. Діаграма прецедентів
8. Діаграма класів
9. Діаграма пакетів

9. Приклади використання
10. Апробація результатів дослідження
11. Висновки

6. Дата видачі завдання 25 лютого 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	14.03.2023	
2	Аналіз існуючих систем	19.03.2023	
3	Підбір засобів розробки	24.03.2023	
4	Визначення вимог	28.03.2023	
5	Визначення архітектури застосунку	30.03.2023	
6	Розробка застосунку	01.05.2023	
7	Вступ, висновки, реферат	05.05.2023	
8	Розробка демонстраційних матеріалів	06.05.2023	
9	Попередній захист роботи	22.05.2023	
10	Подання роботи в деканат	01.06.2023	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____
підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи: 50с., містить 50 рис., 1 табл., 1 дод., 10 джерел.

МОБІЛЬНИЙ ЗАСТОСУНОК, ТРЕНУВАННЯ, .NET MAUI, SQLITE, ANDROID

Об’єкт дослідження – процес контролю персональної спортивної діяльності.

Предмет роботи – застосунок для моніторингу особистих спортивних тренувань.

Мета роботи – спрощення процесу моніторингу особистих спортивних тренувань за рахунок застосування мобільного додатку, створеного мовою C#.

Було проведено порівняння та аналіз існуючих аналогів таких як Windows Notepad, Excel та My Workout Plan.

На основі порівняння аналогів були сформульовані вимоги до майбутнього програмного продукту, серед яких необхідна гнучкість запису інформації, простота використання та спеціалізований інтерфейс.

Програмне забезпечення створено для операційної системи Android мовою C#, використовуючи платформу .NET 7 та фреймворк MAUI. Обрано та обґрунтовано використання архітектури розробки Clean Architecture, MVVM та Dependency Injection, бази даних SQLite та ORM технології Entity Framework Core. Також були використані система керування Git, IDE Visual Studio 2022 та СУБД DB Browser for SQLite. Було описано та обґрунтовано використання даного технічного стеку.

Галузь використання – спортивні тренування.

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IDE - Інтегроване середовище розробки. Це програма, яка надає широкий набір інструментів та функцій для допомоги розробникам у написанні, налагодженні та керуванні програмним кодом. IDE призначені для підвищення продуктивності та оптимізації процесу розробки, надаючи інтегроване середовище, де розробники можуть писати, компілювати, тестувати та розгортати свої програми.

База даних - це структурована колекція даних, яка зберігається таким чином, що дозволяє ефективно зберігати, відновлювати та маніпулювати інформацією.

Міграція - це підхід у розробці програмного забезпечення, в якому схема бази даних генерується і керується на основі коду або моделей, визначених у додатку. Цей підхід часто використовується в об'єктно-орієнтованих фреймворках програмування та ORM (object relational mapping) для полегшення процесу створення та модифікації схеми бази даних.

Вступ

Актуальність теми: У сучасному світі більшість найрозповсюдженіших професій на даний час вимагають сидячого образу життя. Пандемія COVID-19 зробила цю проблему ще гострішою через необхідності праці вдома. Через це спостерігається зростаючий інтерес до фітнесу та здорового способу життя. Відвідування фітнес клубів та заняття спортом стали дедалі більш популярними серед різних вікових та соціальних груп населення. Спорт стає важливою частиною здорового способу життя, який допомагає підтримувати фізичне та психічне здоров'я. Регулярні фізичні вправи зміцнюють м'язи та кістки, покращують кровообіг, знижують ризик розвитку серцево-судинних захворювань та діабету, підвищують імунітет та загальний тонус організму. Крім того, спорт допомагає контролювати вагу та покращує якість сну, що важливо для здоров'я та довголіття [1].

Спорт також має важливе значення для психічного здоров'я людей. Він допомагає знижувати рівень стресу та тривожності, поліпшує настрій та самопочуття, підвищує самооцінку та впевненість у собі. Тренування дають можливість відволіктися від повсякденних проблем, зняти напругу та насолодитися відчуттям своїх можливостей та досягнень [2].

Моніторинг кожного тренування є надзвичайно корисною практикою для будь-якої людини, яка ставить будь-які фітнес цілі, яка може принести значну користь в процесі фізичного розвитку. Це надає можливість бути усвідомленим в особистій “фітнес подорожі”.

Це допоможе:

- спостерігати негативні явища (наприклад плато або перетренування, останнє загрожує індивіду погіршенням якості життя за межами спортивної активності і викликати депресію, хронічну втому і тд. [3]) і змінювати план тренувань, дієту, час і якість відпочинку;

- бути реалістичним у своїх цілях;

- бути вмотивованим, бачучи свої минулі досягнення;

Також є потреба мати змогу адаптивно змінювати записи тренувань, на випадок якщо, наприклад:

- немає можливості використати обладнання, але є можливість виконати аналогічну вправу;

- виконання інших вправ, які не використовують травмовані м’язи;

- можливість додавати або віднімати об’єм вправ в залежності від самопочуття, інших обставин;

Також потрібно передбачити планування тренувань, яке допоможе користувачеві бути більше організованим та вмотивованим.

Об’єкт дослідження – процес моніторингу особистої спортивної діяльності.

Предмет роботи – застосунок для моніторингу особистих спортивних тренувань.

Мета роботи – спрощення процесу моніторингу особистих спортивних тренувань за рахунок застосування мобільного додатку, створеного мовою C#.

Завданням роботи є проектування та розробка мобільного додатку для моніторингу особистих спортивних тренувань.

Методика дослідження: насамперед був проведений аналіз та порівняння існуючих програмних продукти. На основі цього аналізу були визначені основні вимоги, які мають бути реалізовані. Далі, визначити архітектуру та технологічний стек, що використовуватимуться у майбутньому додатку.

Наукова новизна роботи: полягає у створенні зручного мобільного додатку, який надаватиме достатній функціонал користувачеві.

Практична значущість результатів: даний продукт орієнтований на користувачів будь-якої вікової категорії, які займаються фізичними тренуваннями та бажають слідкувати та керувати своїм фізичним розвитком.

1 АНАЛІЗ НАЯВНИХ ЗАСОБІВ ТА ТЕХНОЛОГІЙ ДЛЯ МОНІТОРИНГУ СПОРТИВНИХ ТРЕНУВАНЬ

1.1 Windows Notepad

Windows Notepad - це найвідоміший текстовий редактор, що входить до складу операційних систем Microsoft Windows. Він дозволяє користувачам створювати та редагувати звичайні текстові файли, такі як конфігураційні файли, скріпти, або прості документи, без жодного форматування.

Блокнот має мінімальний інтерфейс користувача з основними функціями, включаючи можливість пошуку та заміни тексту, вставлення часу та дати та перенос слів. Він також підтримує різні кодування тексту, включаючи UTF-8, ANSI та Unicode. Це важливий інструмент для програмістів, системних адміністраторів і всіх, кому потрібно редагувати текстові файли на комп’ютерах Windows.

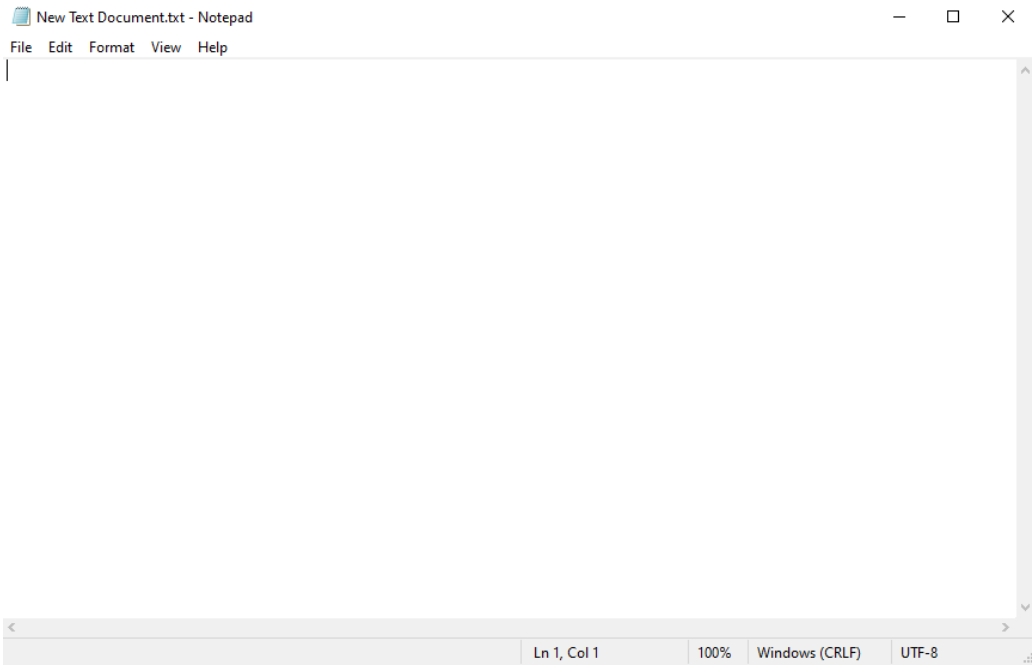


Рисунок 1.1 – Windows Notepad

Переваги Windows Notepad:

- можливість пошуку тренувань по датах;
- можливість додавати свої вправи;
- можливість при записі вправи визначити чи вона на вагу чи на час;
- можливо зовсім видалити підхід вправи;
- можливість призначати тренування на дати;
- незалежність користувача від попередніх навичків користування додатком;

Недоліки Windows Notepad:

- неповноцінний спеціалізований інтерфейс;

1.2 Microsoft Excel

Microsoft Excel — це програма для роботи з електронними таблицями, розроблена компанією Microsoft у складі набору програм Microsoft Office. Excel дозволяє користувачам створювати, редагувати та аналізувати електронні таблиці. Користувачі можуть упорядковувати там фільтрувати дані в рядках і стовпцях і виконувати над ними математичні операції за допомогою вбудованих функцій і формул Excel.

Програма доступна для роботи на різних платформах, таких як Windows, macOS, iOS та Android. Також існує онлайн-версія, яка дозволяє спільно працювати над файлами та взаємодіяти з іншими користувачами у режимі реального часу.

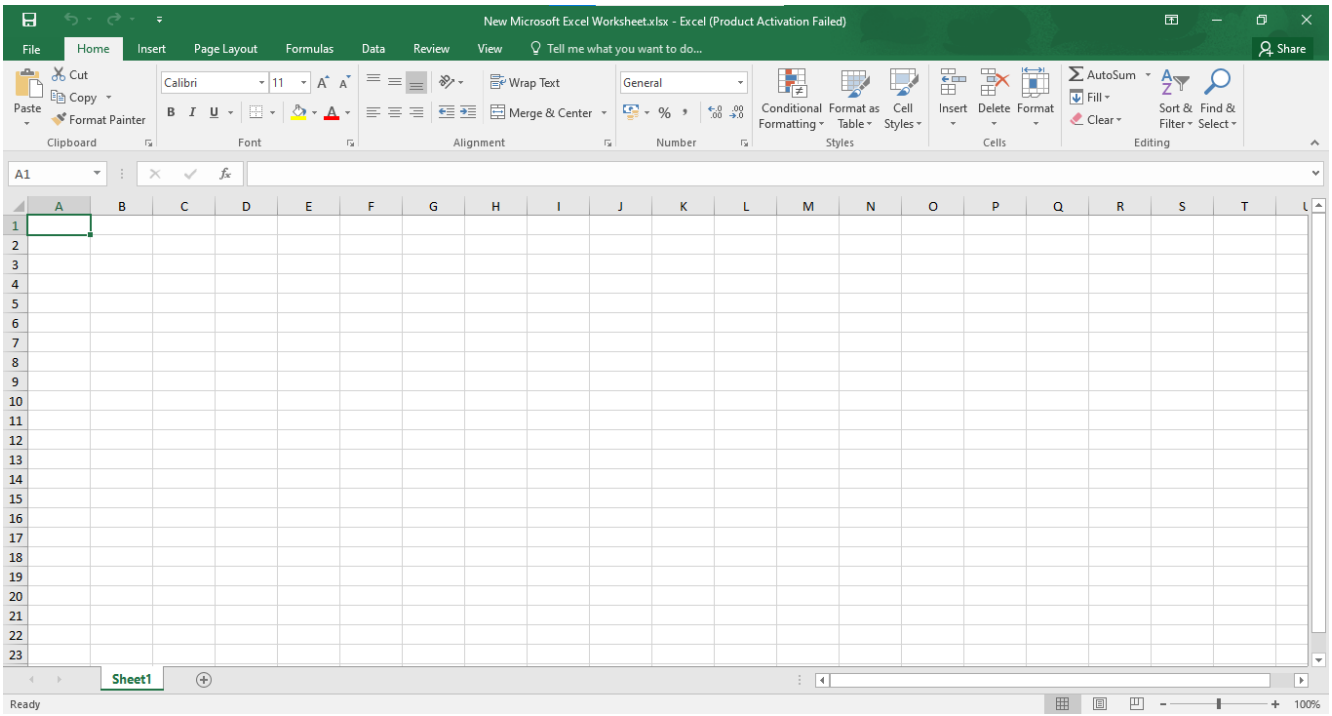


Рисунок 1.2 – Відритий файл Excel

Переваги Microsoft Excel:

- можливість пошуку тренувань по датах;
- можливість додавати свої вправи;
- можливість при записі вправи визначити чи вона на вагу чи на час;
- можливо зовсім видалити підхід вправи;
- можливість призначати тренування на дати;

Недоліки Microsoft Excel:

- неповноцінний спеціалізований інтерфейс;
- залежність від Excel навичків користувача;

1.3 My Workout Plan

My Workout Plan – це простий додаток для Android, розроблений, щоб допомогти користувачам створювати та відстежувати свої фітнес рутини. Додаток містить різноманітні вправи для різних груп м’язів із покроковими інструкціями, які допомагають користувачам виконувати кожну вправу правильно.

Додаток має понад мільйона завантажень, середню користувацьку оцінку 4.4 із 5 та 9 тисяч відгуків у Play Market.

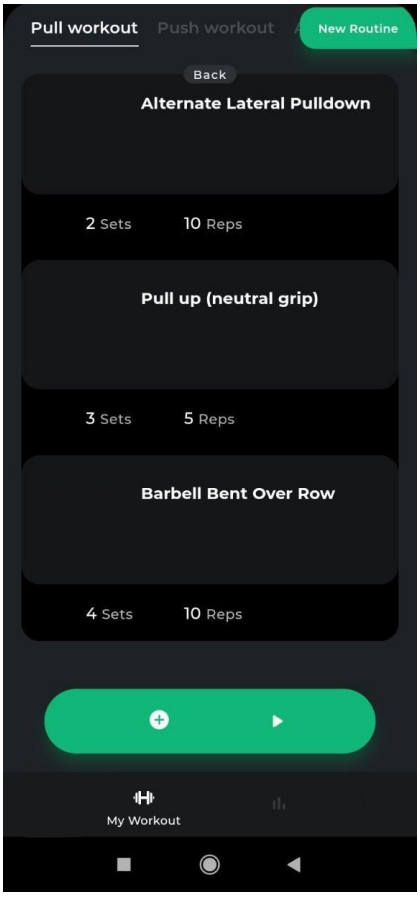


Рисунок 1.3 – Головна сторінка My Workout Plan

Зверху на головній сторінці (рис 1) знаходиться панель із переліком усіх самостійно створених спортивного заняття та кнопка New Routine для створення нового.

При створенні потрібно додати принаймні одну вправу та вказати кількість підходів та повторів. Також можливо створити свою вправу.

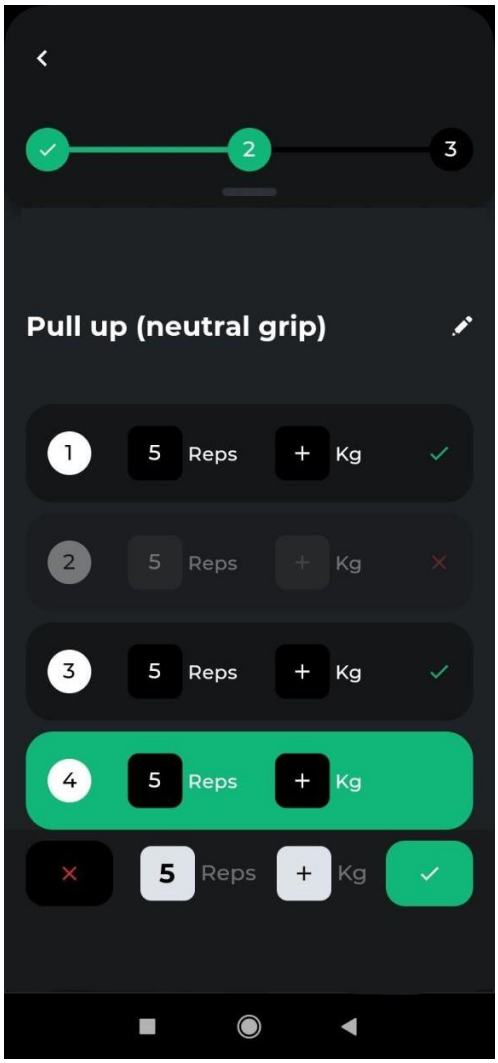


Рисунок 1.4 – Сторінка виконання заняття My Workout Plan

Під час виконання заняття (рис 2) можливо змінювати вагу підходу але не можна його зовсім видалити, лиш пропускати його.

Після завершення заняття є змога переглянути (рис 3) всі завершенні заняття обравши дату в календарі за допомогою кнопки All Workouts.

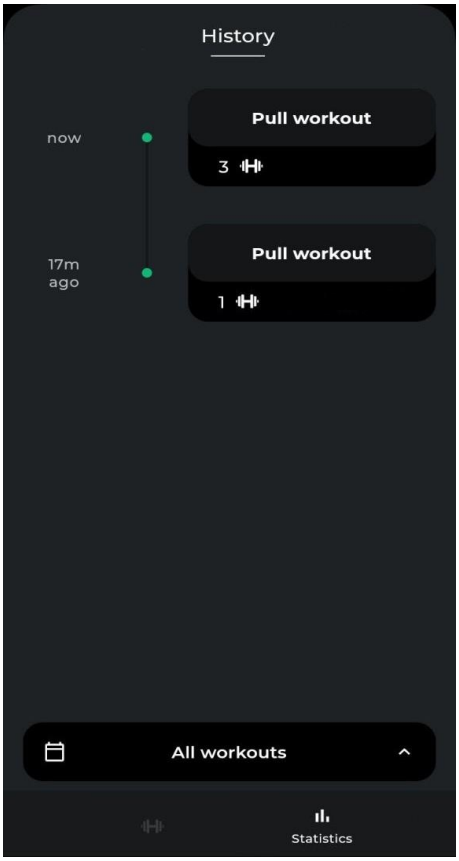


Рисунок 1.5 – Сторінка історії виконаних фітнес занять My Workout Plan

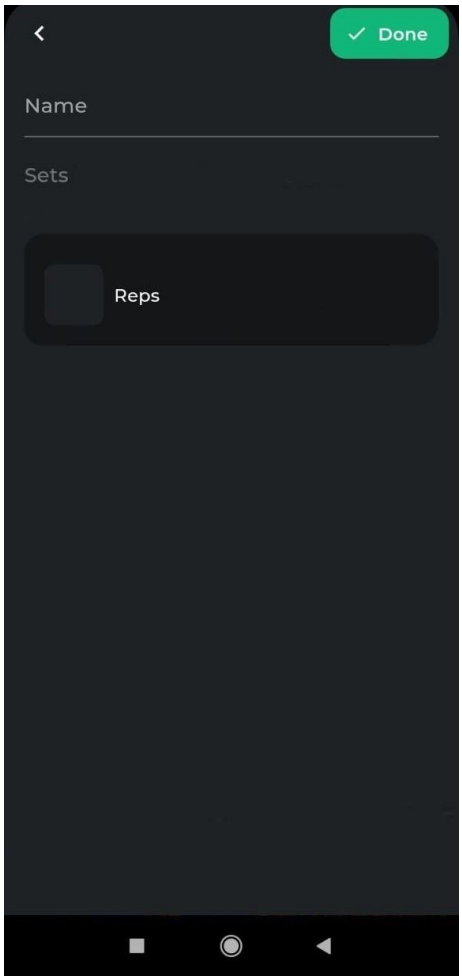


Рисунок 1.6 – Сторінка створення своєї вправи My Workout Plan

Не має можливості обрати у чому вимірювати додану вправу: у вазі чи часі.

Переваги My Workout Plan:

- повноцінний спеціалізований інтерфейс;
- незалежність користувача від попередньої навичків користування додатком;

Недоліки My Workout Plan:

- додавати свої вправи можна лише для конкретного заняття, а не взагалі для всього додатку;
- неможливо при створюванні вправи визначити чи вона на вагу чи на час;
- неможливо зовсім видалити підхід вправи;

- неможливо призначати тренування на дати;
- неможливо пошуку тренувань по датах;

1.4 Висновок

Аналіз призводить до висновку, що у аналогів багато переваг, але є й загальні недоліки, і якщо скласти ці переваги та виключити недоліки, то можна отримати зручний продукт для занять спортом. Складемо таблицю порівняння аналогів, де "+" - наявність, а "-" - відсутність.

Таблиця 1.1 – Порівняння існуючих програм-аналогів

	Windows Notepad	Microsoft Excel	My Workout Plan	TrainingApp
Спеціалізований інтерфейс	-	-	+	+
Пошук по даті	+	+	-	+
Додавання своїх вправ	+	+	+/- (для тренування)	+
Поділ вправ на вагу або час	+	+	-	+
Можливість видалити підхід із тренування	+	+	-	+

Можливість призначити тренування на дату	+	+	-	+
Потребує попередніх знань ПЗ	-	+	-	-

Під час аналізу предметної галузі та аналогів, виявлено, що розроблюваний продукт немає абсолютних аналогів, тому в даному розділі було розглянуто продукти, які мають схожий функціонал та галузь використання. За результатами порівняння було складено таблицю, яка описує ключовий функціонал, що повинен бути реалізований.

2 ВИМОГИ ТА ОЦІНКА ЯКОСТІ СИСТЕМИ

2.1 Функціональні вимоги

2.1.1 Вправи

Користувач повинен мати змогу переглядати вбудовані вправи та переглядати, додавати, редагувати та видаляти свої вправи. Вправи матимуть наступні параметри:

- Назву
- Міра числення (повторення або час)

2.1.2 Тренування

Користувач повинен мати змогу переглядати всі заняття за датою, видаляти та редагувати заняття. В самому занятті додавати та видаляти вправи; змінювати кількість та порядок підходів. У самому підході змінювати кількість повторень або час.

2.1.3 Рутини

Рутина повинна відчуватися як шаблон до тренувань. Користувач повинен мати змогу переглядати, додавати, редагувати та видаляти рутини. В самій рутині додавати та видаляти вправи; змінювати кількість та порядок підходів. Користувач повинен мати змогу обрати дати за якими виконувати; почати рутину означає почати тренування по шаблону рутини.

2.1.3 Діаграма використання

Функціональні вимоги додатково представлені у виді діаграми використання (див. рис. 1).

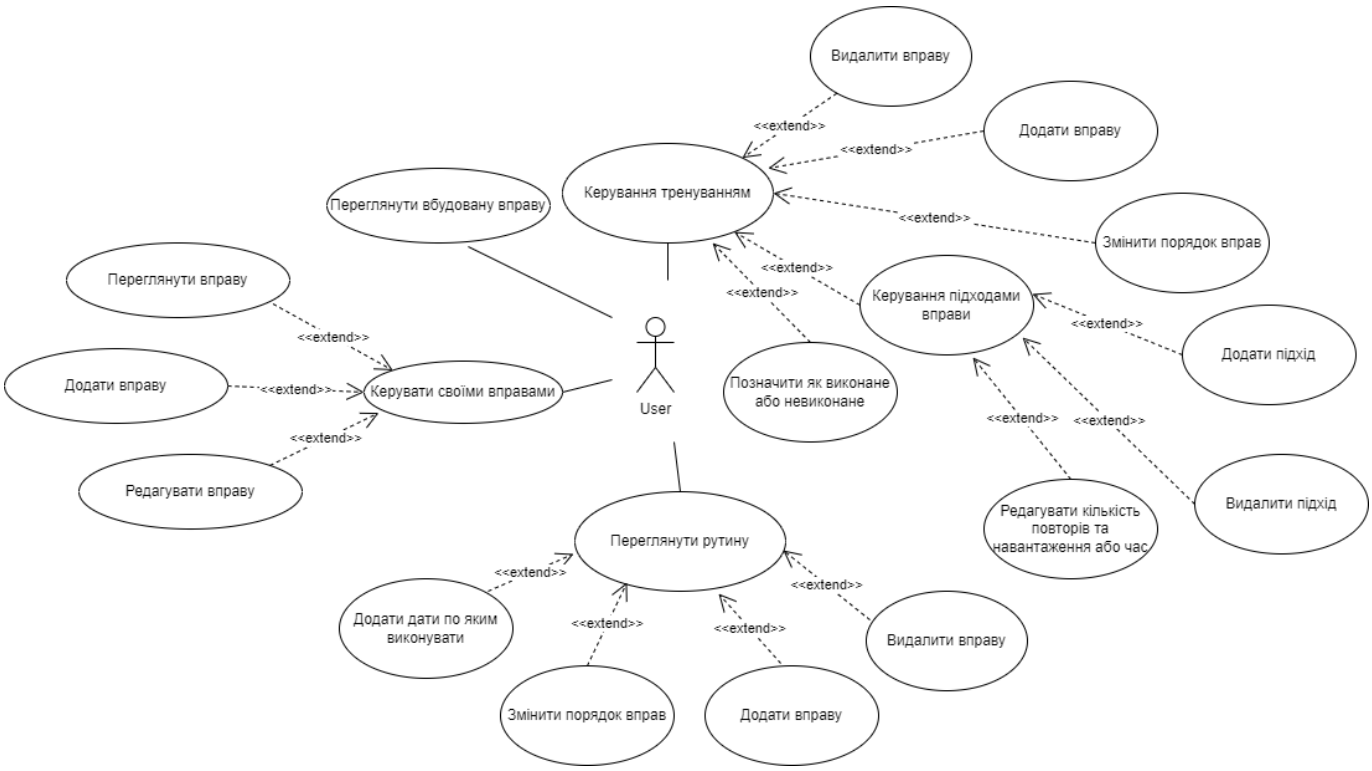


Рисунок 2.1 – Діаграма використання

2.2 Нефункціональні вимоги

До нефункціональних вимог застосунку входить:

- Операційна система – Android;

3 ВИБІР ЗАСОБІВ РОЗРОБКИ

3.1 ІНСТРУМЕНТИ РОЗРОБКИ

3.1.1 Visual Studio 2021

Visual Studio 2021 - це інтегроване середовище розробки (IDE) від Microsoft, яке надає розробникам інструменти для створення різноманітних програмних продуктів, веб-сайтів, мобільних додатків, ігор, сервісів, та багатьох інших рішень. Visual Studio 2021 підтримує багато мов програмування, таких як C++, C#, Visual Basic, F#, Python, JavaScript, і багатьох інших.

Середовище включає різноманітні інструменти для дизайну інтерфейсу користувача, візуального програмування, створення коду, налагодження, тестування, інтеграції з різними засобами розробки, такими як Git, Azure DevOps, та іншими. Visual Studio 2021 має широку підтримку платформ, таких як Windows, Linux, Android, iOS, macOS, та інших.

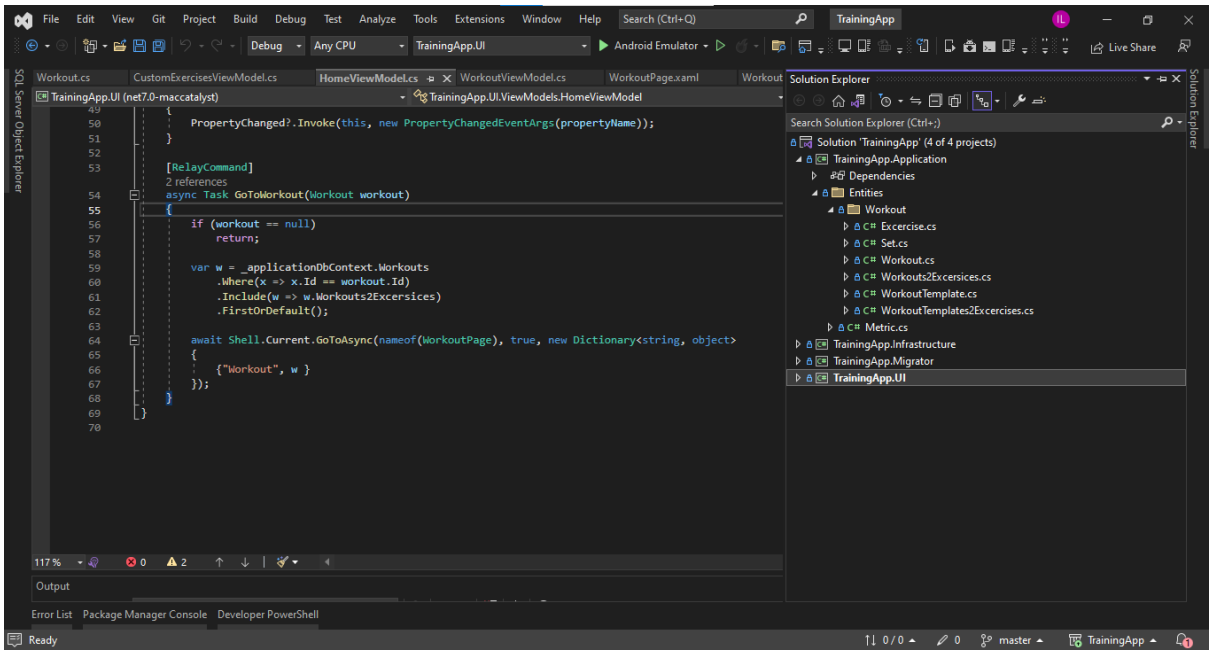


Рисунок 3.1 – Вікно IDE Visual Studio 2021

3.2.2 Git

Git – популярна розподілена система керування версіями файлів. Вона надає можливість легко відслідковувати зміни в програмному продукті, вносити нові, створювати гілки, повертатися до старих версій та розроблювати продукт в команді. Для роботи з системою контролю версій більшість IDE має зручний візуальний інтерфейс, але для розробки також можна використовувати Git Bash та консоль Windows Terminal.

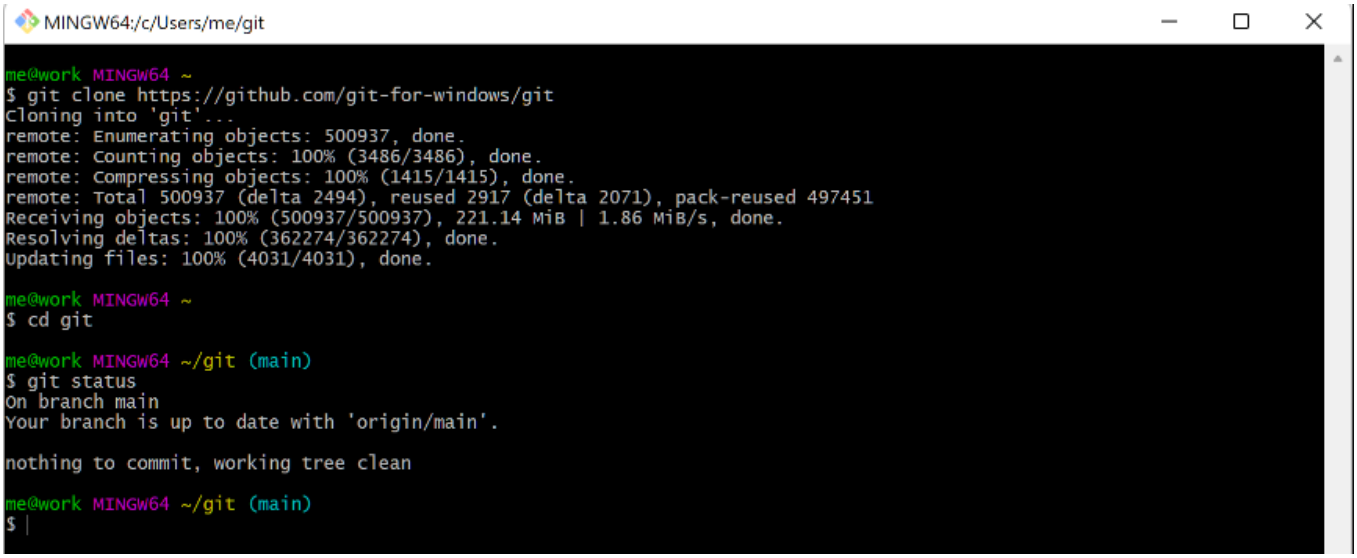


Рисунок 3.2 – Консоль Git

3.2.3 DB Browser for SQLite

DB Browser for SQLite - це безкоштовний крос-платформний інструмент для роботи з базами даних SQLite. Він надає зручний інтерфейс для перегляду, редагування та керування базами даних SQLite. DB Browser for SQLite підтримує створення та виконання запитів SQL, імпортування та експортування даних, а також забезпечує можливість перегляду структури таблиць та їх вмісту.

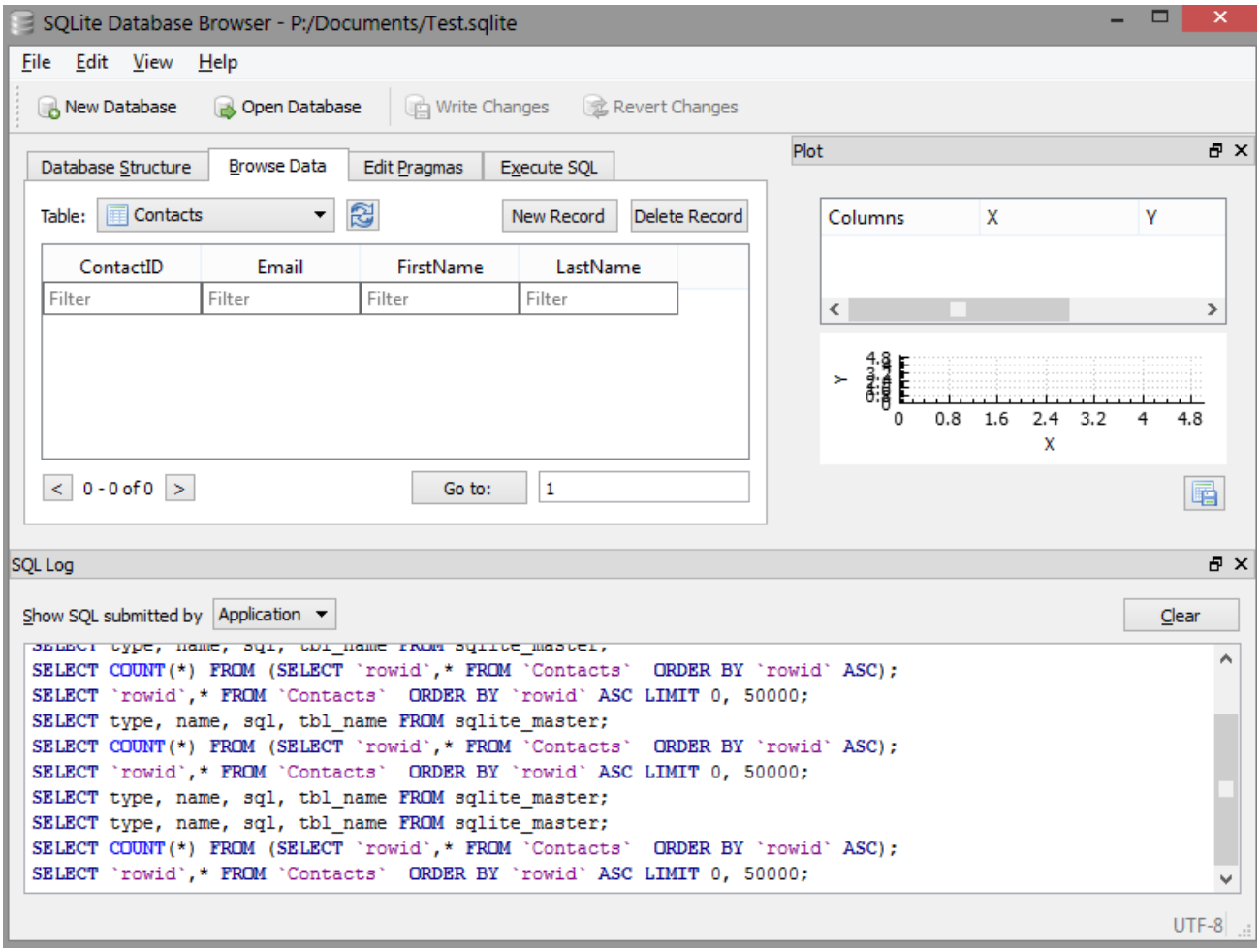


Рисунок 3.3 – Інтерфейс DB Browser for SQLite

3.2 Технології розробки

Для реалізації програмного продукту було вирішено використати архітектуру Clean Architecture. Це патерн архітектури програмного забезпечення, який акцентує на розділенні відповідальностей та незалежності бізнес-логіки додатку від технічних деталей реалізації. Він базується на принципах SOLID та використовує принцип інверсії залежностей для досягнення модульності та гнучкості [4].

Архітектура складається з чотирьох основних рівнів:

1. Domain: Надає основу застосунку та містить сутності та бізнес-логіку системи.

2. Infrastructure: Забезпечує доступ до зовнішніх ресурсів, таких як бази даних, веб-сервіси та файлові системи.

3. Application: Надає точку входу для користувацького інтерфейсу та організовує взаємодії між рівнями Domain та Infrastructure.

4. Presentation: Містить компоненти користувацького інтерфейсу, такі як веб-сторінки, форми та види.

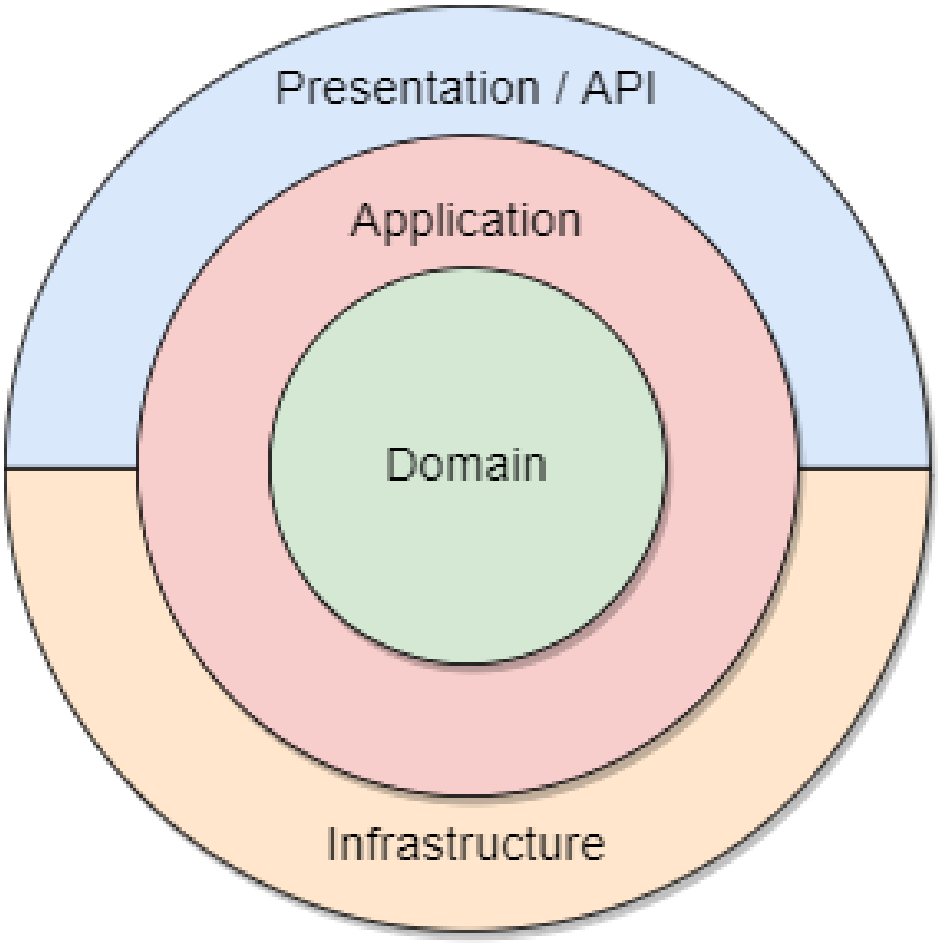


Рисунок 3.4 – Приклад Clean Architecture

Переваги використання Clean Architecture:

- масштабування різних рівнів незалежно один від одного, що дозволяє кращу продуктивність і роботу зі змінами;

- розділення програмної системи на окремі рівні сприяє модульності, що робить систему легшою у підтримці та оновленні;

- розділення рівнів дозволяє легше перевикористовувати код між різними проектами та системами;

- забезпечення кращої безпеки даних, надаючи чітке розмежування між рівнем презентації та рівнями додатку та даних;

- гнучкість архітектури дозволяє різним рівням бути розробленими за допомогою різних технологій, що дозволяє використовувати найбільш підходящу технологію для кожного рівня;

Для реалізації програмного продукту було вирішено використати спрощену архітектуру в якій рівні Application та Domain об'єднанні в один. Це спростить розробку системи, але не змінить головний принцип шаблону.

3.2.1 Середовище розробки .NET

Усі рівні було розроблено на базі фреймворку .NET 7 мовою програмування C# версії 11.

.NET – потужна open-source платформа розробки програмного забезпечення, що активно розробляється компанією Microsoft. Ця платформа призначена для створення крос-платформенних застосунків на різних операційних системах, таких як Windows, Linux та macOS. .Вона складається з мов програмування, бібліотек класів та інструментів, що дозволяє розробникам створювати програмне забезпечення для різних платформ, таких як Windows, Linux, macOS та мобільних пристроїв. Платформа .NET підтримує різні мови програмування, такі як C#, F#, Visual Basic та інші, і дозволяє розробникам використовувати їх для створення програмного забезпечення [5].

Переваги використання платформи .NET:

- підтримка кросплатформенності;

- велика спільнота розробників;

- багата екосистема, яка включає в себе широкий спектр інструментів та технологій, таких як IDE Visual Studio, Azure та NuGet, що дозволяє легко розробляти, розгортати та управляти додатками;

- оптимізація та швидкість;

- вбудована Dependency Injection;

3.2.2 Dependency Injection

Dependency injection (Ін’єкція залежності) - це шаблон проектування, вбудований у платформу .NET, який дозволяє об’єкту оголошувати свої залежності від зовнішнього об’єкта, який надає ці залежності. Залежності, оголошені, зазвичай є інтерфейсами класів, а залежності, які надає, є конкретними реалізаціями для цих інтерфейсів. Це допускає слабке з’єднання коду, оскільки об’єкт більше не потребує ініціалізації власних залежностей [6].

3.2.3 MAUI

MAUI - це фреймворк, створений компанією Microsoft, який використовується для створення кросплатформних застосунків мовами програмування C# та XAML. Був вперше випущений у травні 2022 року на основі платформи .NET 6 і є еволюцією Xamarin.Forms, що дає можливість не тільки легко адаптуватися розробникам із досвідом, але і мігрувати старий код на нову платформу. Наразі підтримується операційні системи Android, Windows, iOS та macOS [7].

MAUI націлений на реалізацію якомога більшої кількості спільної логіки та елементів користувацького інтерфейсу із єдиної кодової бази, але водночас надання можливості створювати код та ресурси для специфічних платформ.

```
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MauiApp8.MainPage">

    <ScrollView>
        <VerticalStackLayout Spacing="25" Padding="30">

            <Label
                Text="Hello, World!"
                FontSize="32"
                HorizontalOptions="Center" />

            <Label
                Text="Welcome to .NET Multi-platform App UI"
                FontSize="18"
                HorizontalOptions="Center" />

            <Label
                Text="Current count: 0"
                FontSize="18"
                FontAttributes="Bold"
                x:Name="CounterLabel"
                HorizontalOptions="Center" />

            <Button
                Text="Click me"
                FontAttributes="Bold"
                Clicked="OnCounterClicked"
                HorizontalOptions="Center" />

            <Image
                Source="dotnet_bot.png"
                WidthRequest="250"
                HeightRequest="310"
                HorizontalOptions="Center" />

        </VerticalStackLayout>
    </ScrollView>
</ContentPage>
```

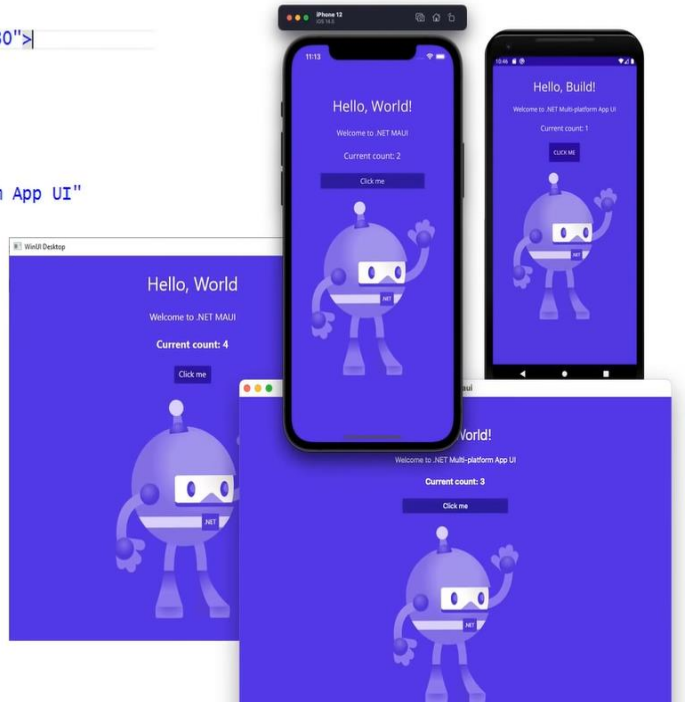


Рисунок 3.5 - Приклад XAML коду та його репрезентацій на різних платформах

Фреймворк базується на новій відносно Xamarin.Forms архітектурі (Рис. 3.5), яка абстрагує деталі конкретної платформи і дозволяє розробникам використовувати спільний API для доступу до інформації про девайс, датчиків, файлів, буфер обміну і тд.

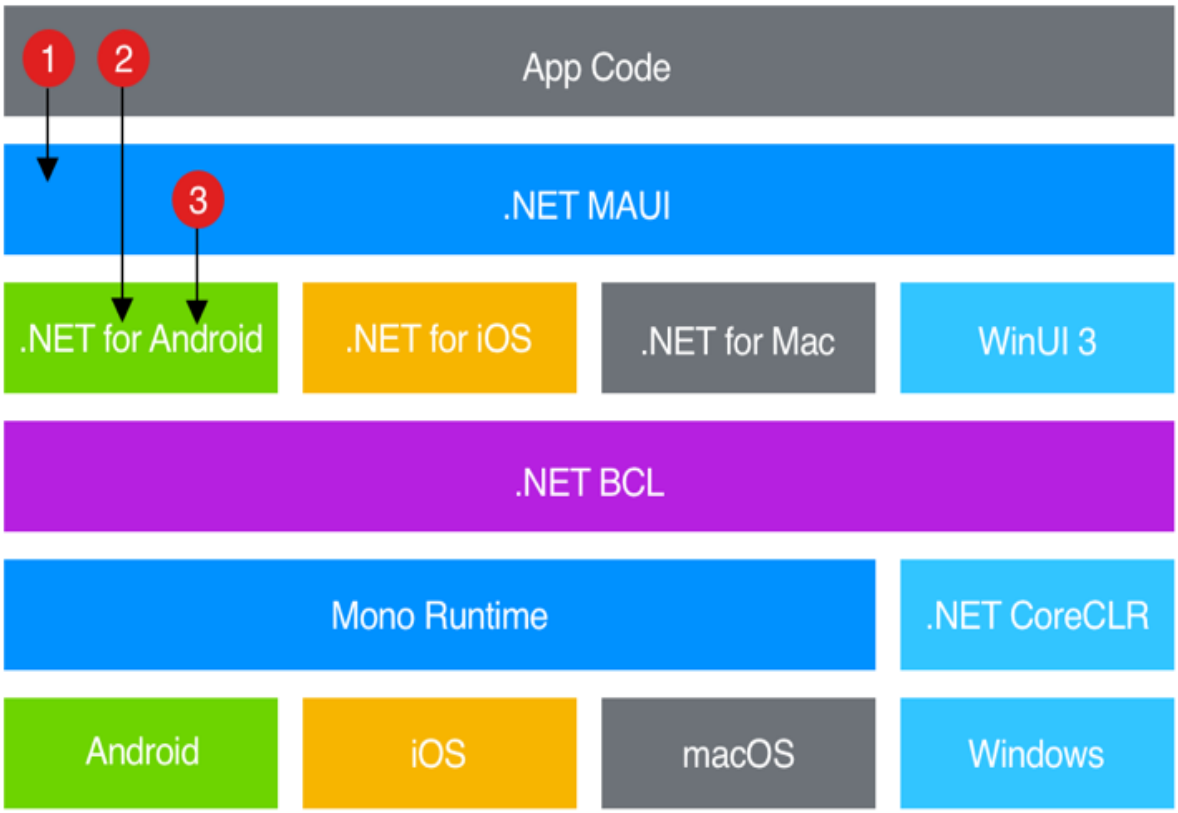


Рисунок 3.6 - Архітектура MAUI

Фреймворк використовує ряд бібліотек, для взаємодії з кожною конкретною платформою: .NET for Android, .NET for iOS, .NET for macOS та WinUI 3. Кожна з них має доступ до бібліотеки .NET Base Class Library (BCL), яка надає спільний абстрактний API [7].

На рисунку 3.5 наведено приклад більшості випадків, у яких код програми звертається до платформи .NET MAUI (1), а потім до бібліотеки для конкретної ОС (3). Також, якщо потрібно, код програми може звернутися відразу до бібліотеки платформи (2).

3.2.4 MVVM

MVVM (Model-View-ViewModel) – архітектурний патерн програмного забезпечення, який відокремлює логіку користувацького інтерфейсу (UI) від бізнес-логіки додатку. UI розділяється на дві частини: View, що відповідає за відображення даних, та View Model, що відповідає за керування даними та надання їх для View. Model представляє дані та бізнес-логіку додатку.

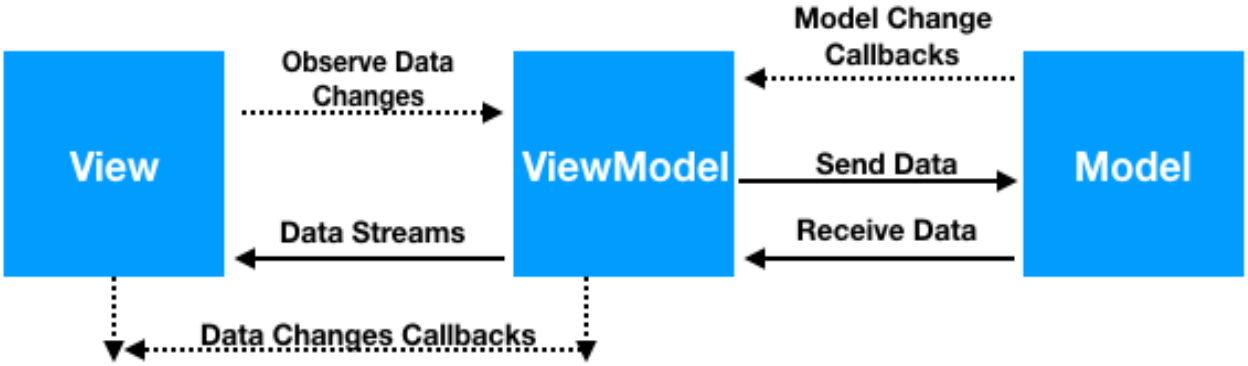


Рисунок 3.7 – Приклад патерну MVVM

Для реалізації патерна MVVM була використана безкоштовна open-source бібліотека CommunityToolkit.Mvvm, яка активно розроблюється спільнотою .NET на GitHub. Вона надає набір інструментів, які спрощують звичайні завдання в додатках, які використовують MVVM, такі як зв'язування даних між View та View Model, команди, валідація та навігація. Вона також включає набір базових класів для реалізації патерну MVVM.

3.2.5 SQLite

Для зберігання даних було вирішено використовувати базу даних SQLite, через такі переваги:

- SQLite працює на різних платформах, включаючи iOS, Android та Windows;
- займає мінімальну кількість пам’яті та місця на пристрої;
- висока швидкість;
- не потребує установки та адміністрування [8];

3.2.6 ORM

ORM (Object-relational mapping) - це технологія, яка дозволяє взаємодіяти з базою даних, використовуючи модулі та об'єкти мов програмування. Всі таблиці та записи в джерелі даних відображаються у вигляді класів та об'єктів, а CRUD (Create Read Update Delete) операції виконуються за допомогою відповідних методів. Для транслювання запитів до конкретної бази даних ORM використовує провайдери або драйвери баз даних. Однак, особливості технології, такі як транслювання запитів та створення схеми бази, залежать від бібліотеки та мови програмування [9].

Таблиці SQLite будуть створені за допомогою Code-First міграцій. Схема бази даних автоматично генерується на основі класів C# та конфігурації Fluent API, які вказують відповідності між класами та схемою бази даних.

Для підключення до SQLite використовується ORM Entity Framework Core 7 та провайдер Microsoft.EntityFrameworkCore.Sqlite. Це потужна бібліотека, яка вміє транслювати код LINQ (Language-Integrated Query) в запит до бази даних. Також є система відслідковування змін, яка дозволяє просто оновлювати властивості об’єкту та викликати метод для збереження [9].

3.2.7 Repository Pattern

Repository Pattern - це шаблон проектування програмного забезпечення, який абстрагує та інкапсулює логіку доступу до даних та надає спосіб розділити логіку застосунку від конкретного способу збереження даних, такого як база даних. Основна ідея шаблону полягає в створенні інтерфейсу, який визначає операції, які можна виконувати з певним об'єктом або набором об'єктів, а потім реалізовувати цей інтерфейс у конкретному класі, який взаємодіє з сховищем даних [10].

3.2.8 Додаткові бібліотеки

Додатково до вбудованих можливостей .NET MAUI також використовуються такі додаткові бібліотеки для розробки користувацького інтерфейсу:

1. CommunityToolkit.Maui – open-source бібліотека, яка співпрацює із розробниками .NET MAUI;

2. Syncfusion.Maui;

Вони додають елементи інтерфейсу та розширюють поведінку існуючих, що робить розробку інтерфейсу простішою.

4 РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

4.1 Реалізація архітектури

В межах реалізації спрощеної Clean Architecture було створено три проекта та один додатковий для міграцій (див. рис 4.1).

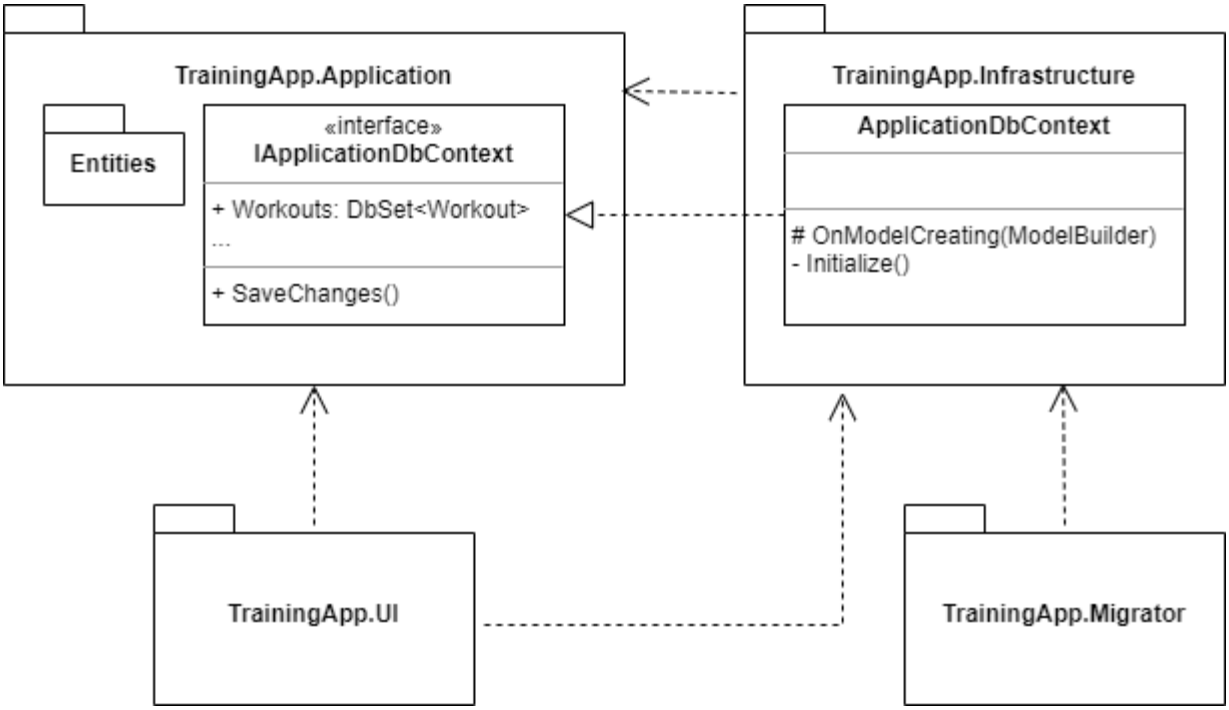


Рисунок 4.1 – Діаграма пакетів застосунку

1. TrainingApp.Application: містить сутності, інтерфейси взаємодії між рівнями інфраструктури та презентації.

2. TrainingApp.Infrastructure: містить логіку підключення до БД.

3. TrainingApp.UI: містить користувацький інтерфейс MAUI.

4. TrainingApp.Migrator: використовується для створення міграцій.

Для реалізації шаблону Repository Pattern на рівні TrainingApp.Application було створено інтерфейс IApplicationDbContext, який має у собі поля вбудованого до Entity Framework Core типу DbSet із всіма сутностями, які знаходяться у Entities. Його імплементує ApplicationDbContext, який знаходиться у TrainingApp.Infrastructure. Використовуючи вбудовану у .NET біблотеку Microsoft.Extensions.DependencyInjection, саме цей інтерфейс використовується для доступу до SQLite.

У пакеті TrainingApp.Application.Entities знаходяться сутності, які використовуються у бізнес логіці застосунку.

4.2 Моделювання таблиць SQLite

Оскільки взаємодія з БД буде здійснюватися за допомогою об’єктів, а SQLite не потребує схеми, то моделювання таблиць здійснено за допомогою діаграми класів (див. рис. 4.2).

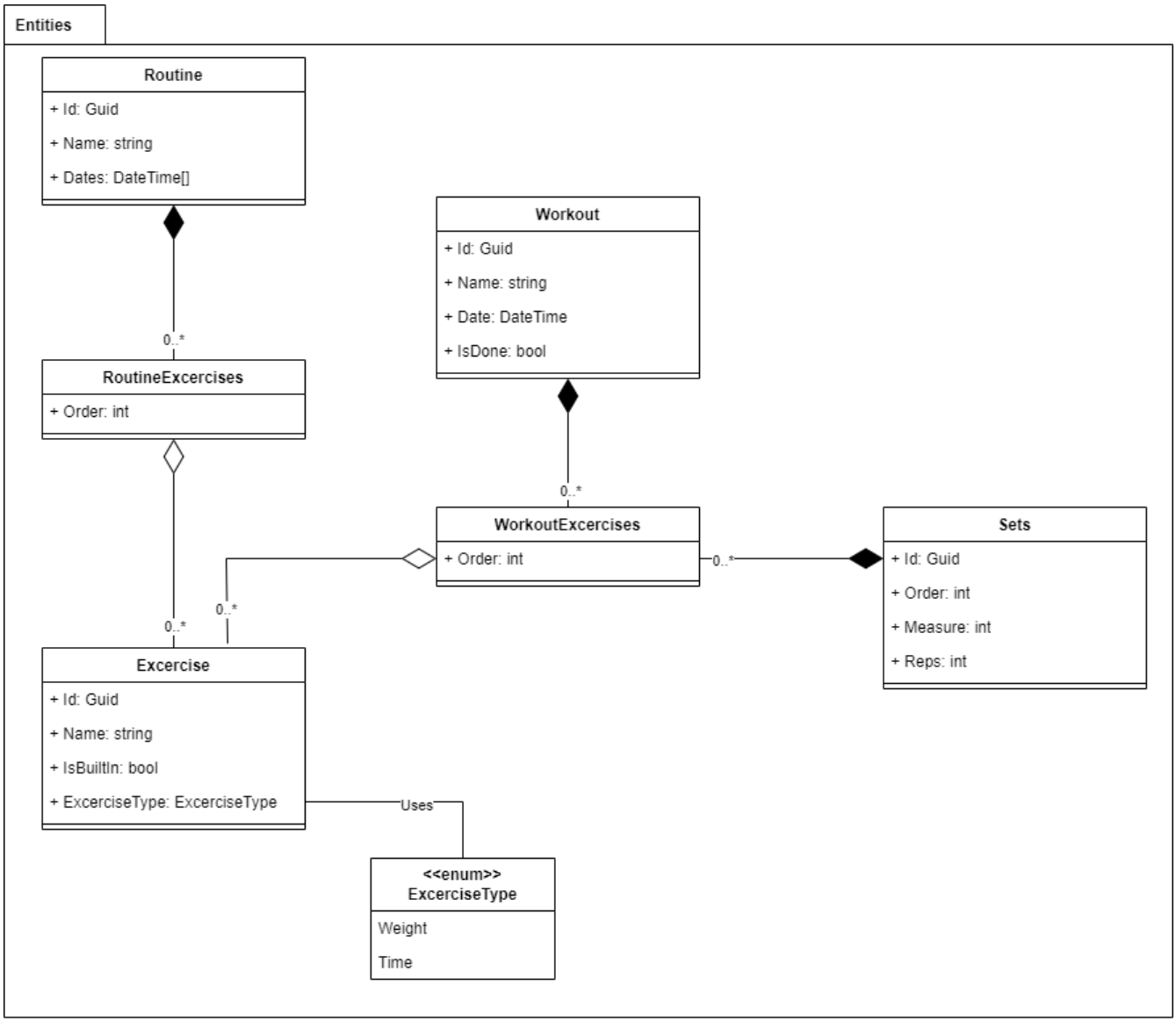


Рисунок 4.2 – Діаграма класів сутностей застосунку

Для конфігурації таблиць для кожної сутності було використано інтерфейс IEntityTypeConfiguration, який є вбудований до Entity Framework Core. Він дозволяє правильно визначити зв'язки і надає методи для конфігурування різних аспектів сутності, таких як назва таблиці, назви колонок, типи даних, ключі, зв'язки між сутностями та багато іншого.

Для створення міграцій використовується звичайний консольний застосунок TrainingApp.Migrator. Він створює класи міграцій у TrainingApp.Infrastructure, які використовує самий MAUI проект із користувацьким інтерфейсом.

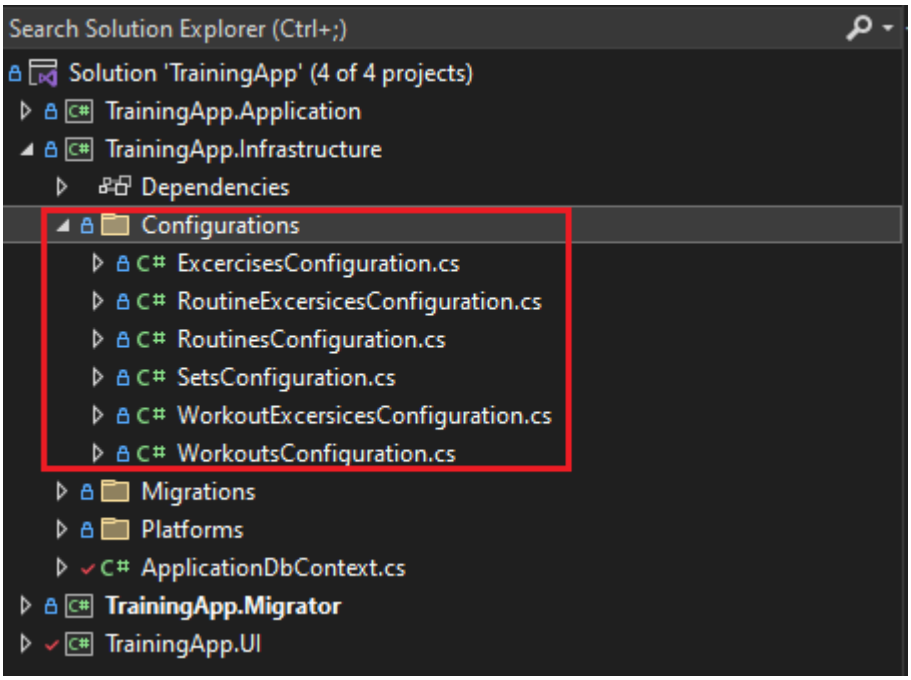


Рисунок 4.4 – Всі класи конфігурації сутностей

4.3 Розробка користувацького інтерфейсу

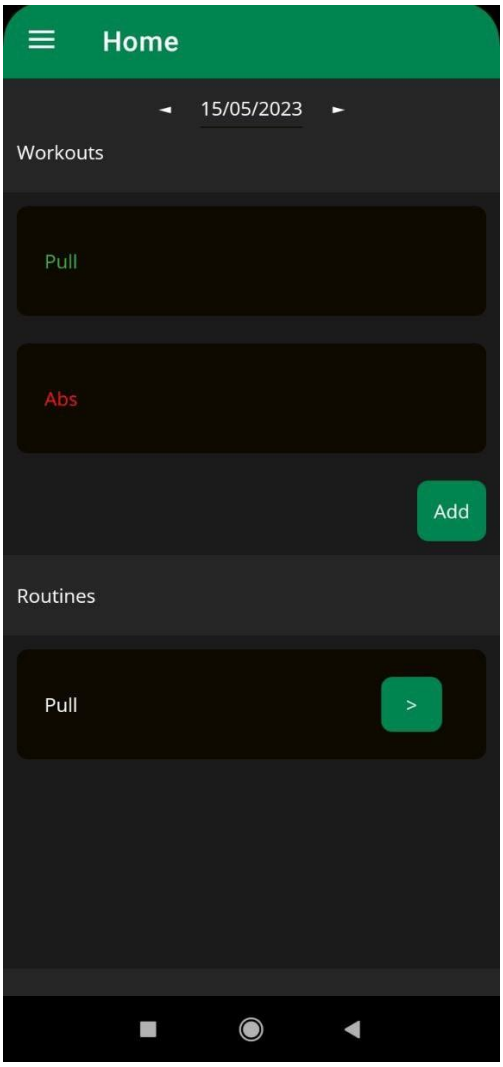


Рисунок 4.5 – Головна сторінка

На головні сторінці (див рис 4.5) знаходиться навігація по датам, тренування та рутини за поточну дату. Тренування, які користувач відмітив зробленими, зображуються зеленим кольором, а незроблені червоним. Також можна додати нове тренування або почати тренування із рутини.

Зверху знаходиться кнопка, яка відкриває бокове меню, розроблене за допомогою Shell – вбудований у MAUI функціонал, який надає інфраструктуру управління потоком навігації між сторінками застосунку.

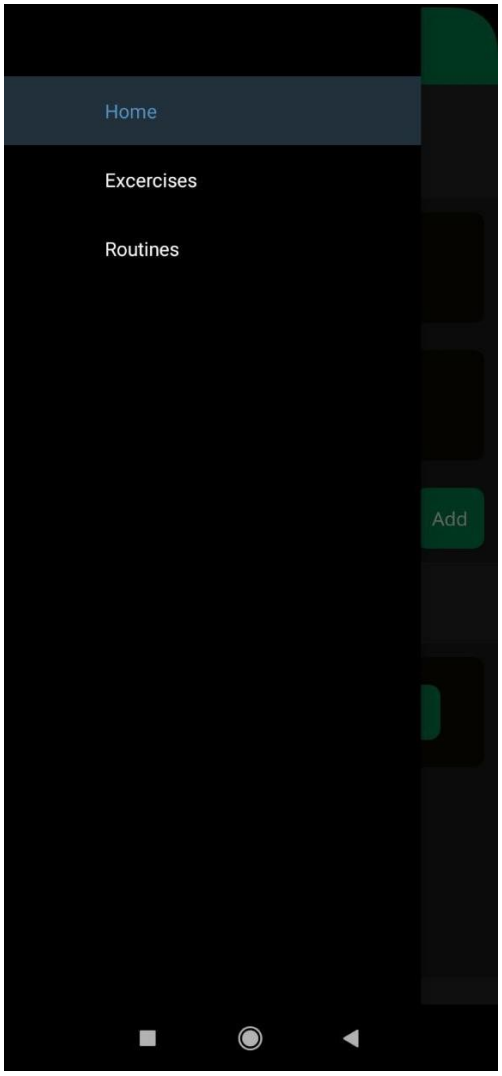


Рисунок 4.6 – Бокове меню застосунку

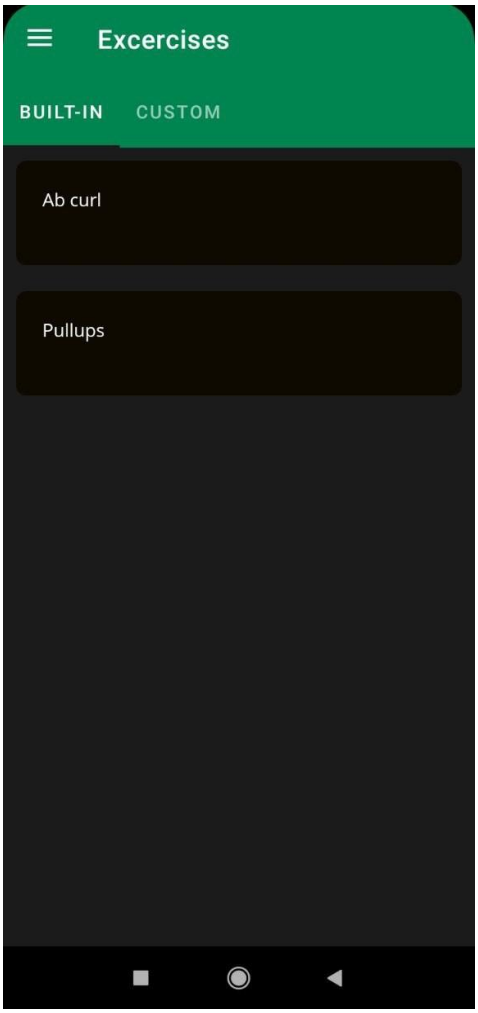


Рисунок 4.7 – Сторінка вправ

Рисунок 4.8 – Сторінка списку рутин

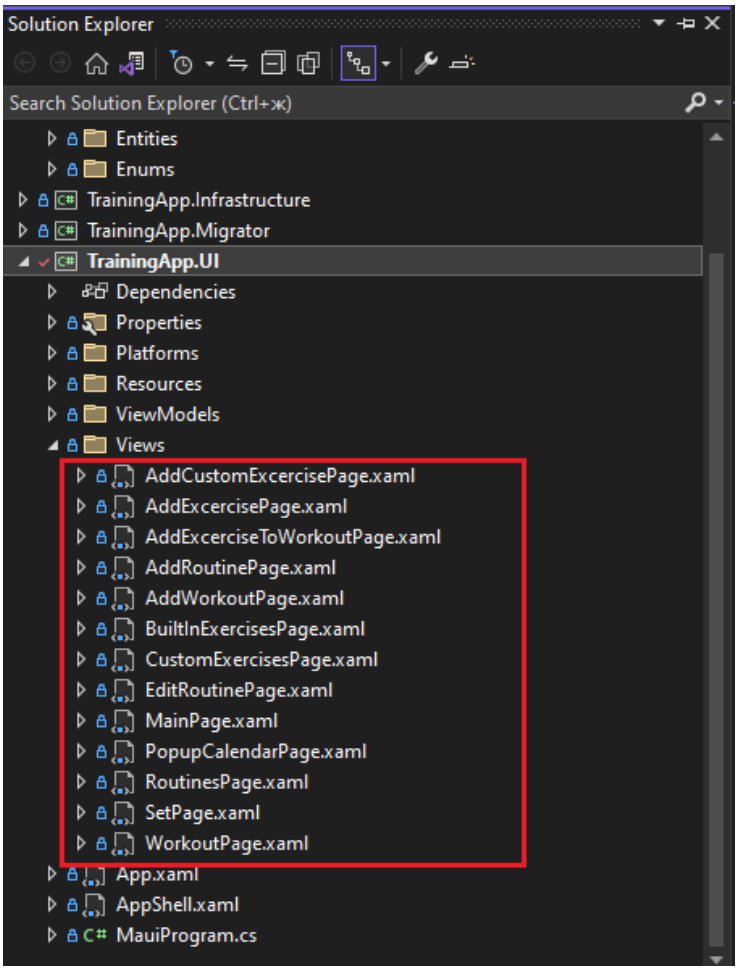


Рисунок 4.9 – Всі створені сторінки

4.4 Мануальне тестування системи

Мануальне тестування є процесом, в якому тестувальники виконують тестові сценарії вручну, без використання автоматизованих інструментів. Цей підхід дозволяє перевірити, чи відповідає програмне забезпечення вимогам, встановленим у відповідних документах. Мануальне тестування вважається важливим, оскільки воно дозволяє виявити як видимі, так і приховані помилки програми. Коли фактичний результат відрізняється від очікуваного, це вважається помилкою, яку треба документувати та виправляти.

Мануальне тестування є обов'язковим кроком для нещодавно розробленого програмного забезпечення. Хоча воно вимагає значних зусиль і часу, ручне тестування забезпечує гарантію на відсутність помилок у програмному забезпеченні.

4.4.1 Тест керуванням рутинами

Перейшовши на сторінку Routines через бокове меню, можна бачити список всіх рутин, які додав користувач.

Рисунок 4.10 – Сторінка рутин

Створити нову та потім редагувати рутину можливо натиснувши на кнопку Add.

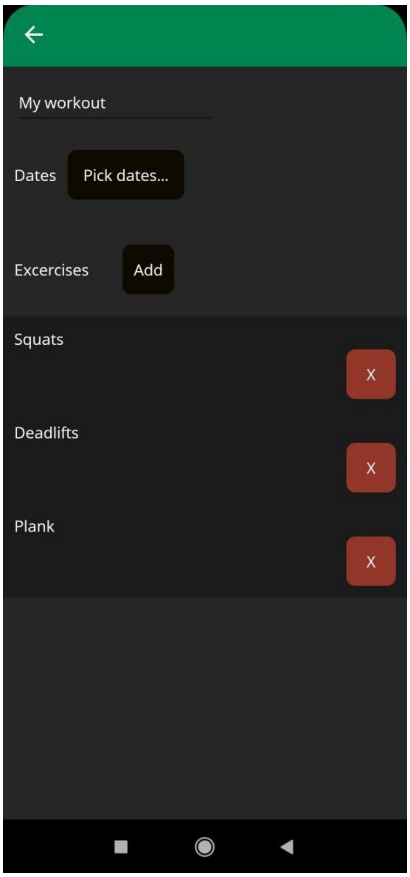


Рисунок 4.11 – Сторінка редагування рутини

На сторінці редагування рутини можливо додавати дати, по яким планується тренування, та вправи (див. рис. 4.12 та 4.13).

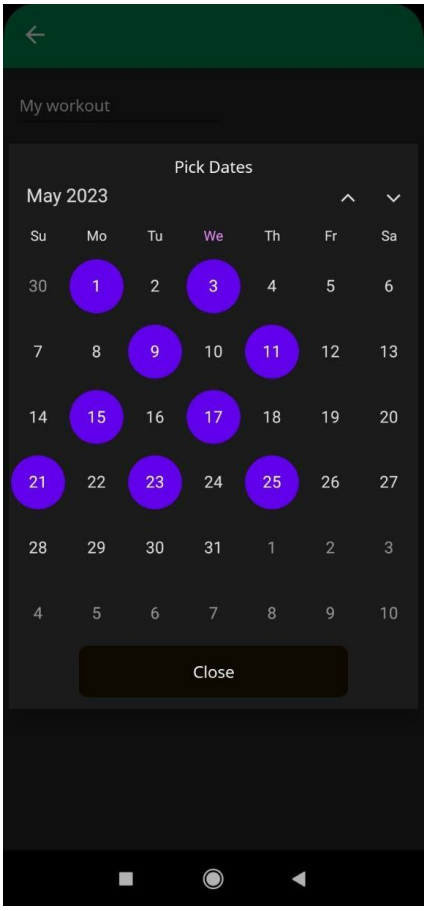


Рисунок 4.12 – Сторінка обрання дат

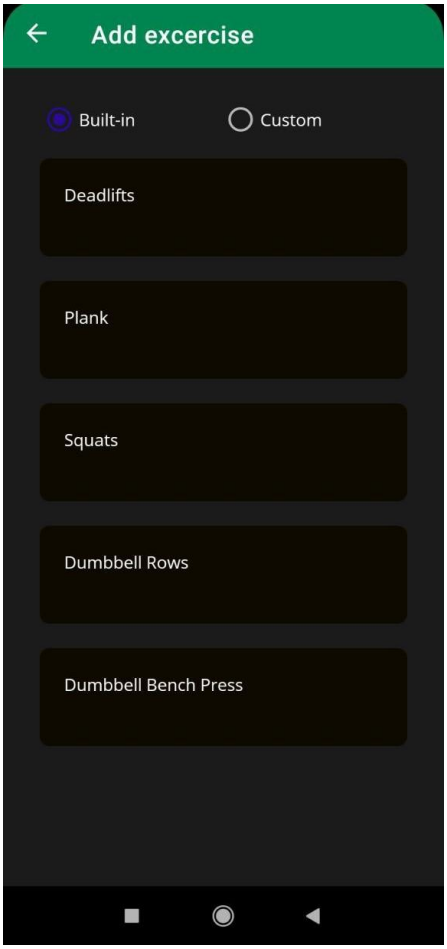


Рисунок 4.13 – Сторінка додавання вправ

Після того, як була обрана дата, на головній сторінці з’являється ця рутина і можна почати тренування по ній (див. рис. 4.14 та 4.15).

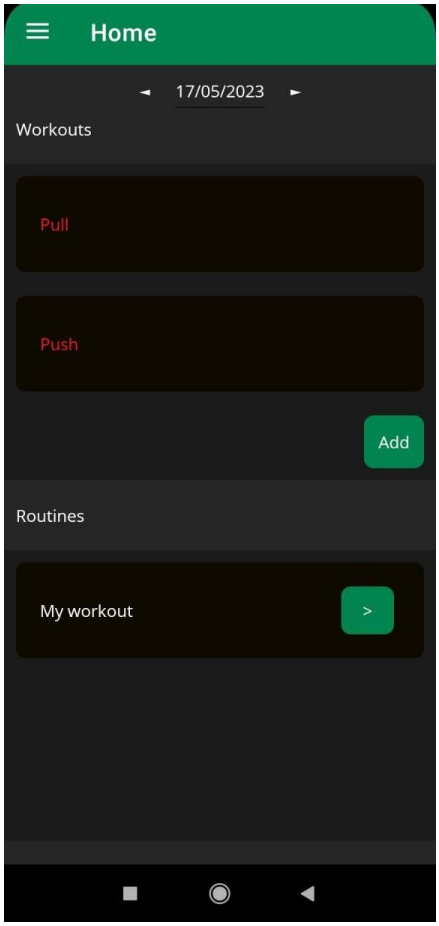


Рисунок 4.14 – Головна сторінка із новою рутиною

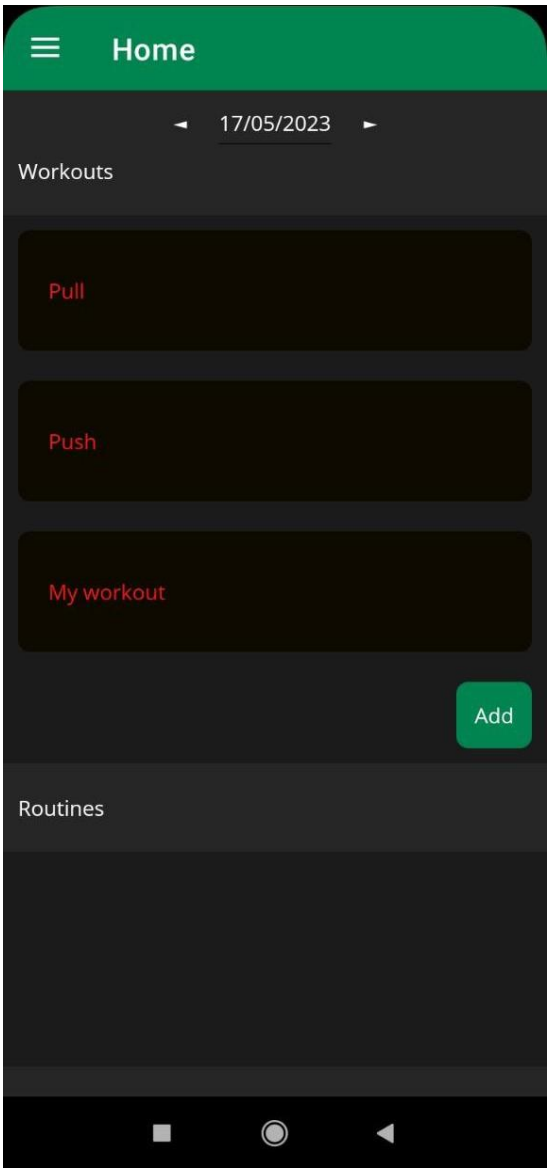


Рисунок 4.15 – Головна сторінка із новим тренування

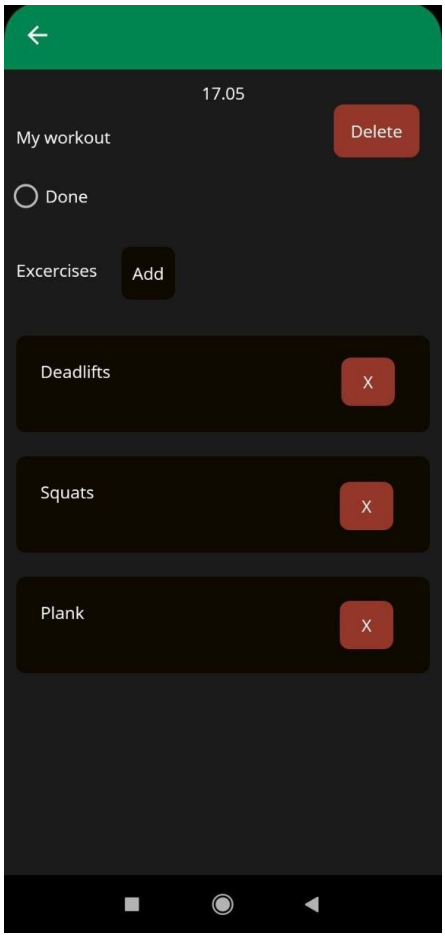


Рисунок 4.16 – Сторінка тренування

Було створене нове тренування по шаблону рутини і було відмічено невиконаним.

4.4.2 Тест керуванням тренуваннями

Перейшовши на головну сторінку можна бачити список всіх тренувань за поточну дату (див. рис. 17).

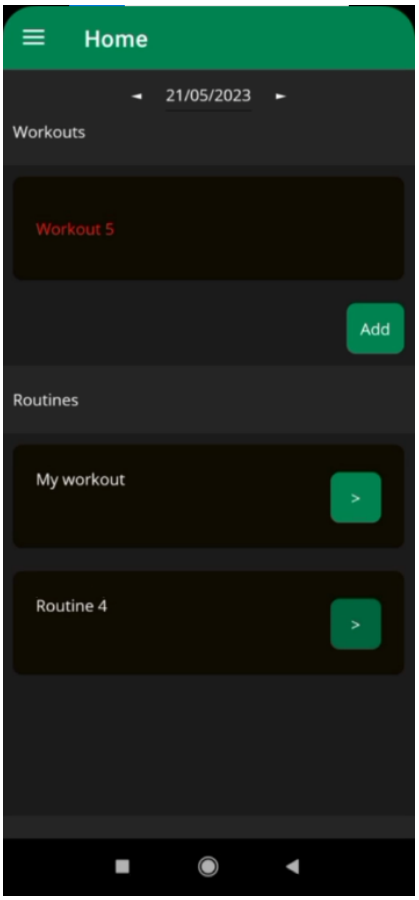


Рисунок 4.17 – Головна сторінка

Перейшовши до сторінки тренування можна її видалити, позначити як виконану; додавати нові вправи, змінювати їх порядок та видаляти (див. рис. 18).

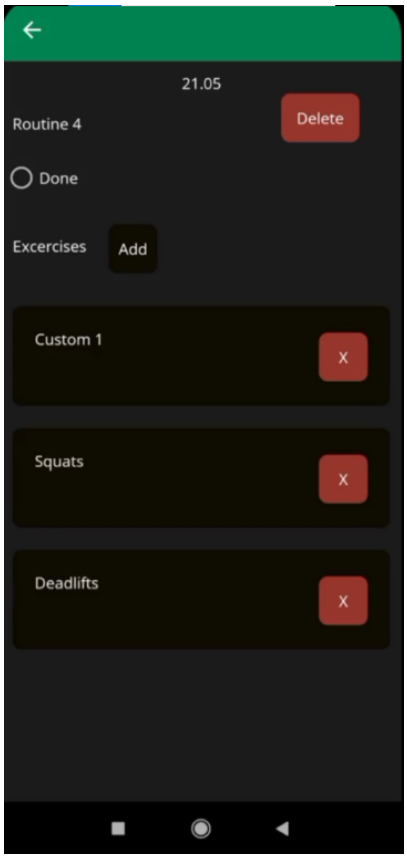


Рисунок 4.18 – Сторінка редагування тренування

Натиснувши кнопку додавання нових вправ є можливість додати вправу із списку (див. рис. 19) до тренування.

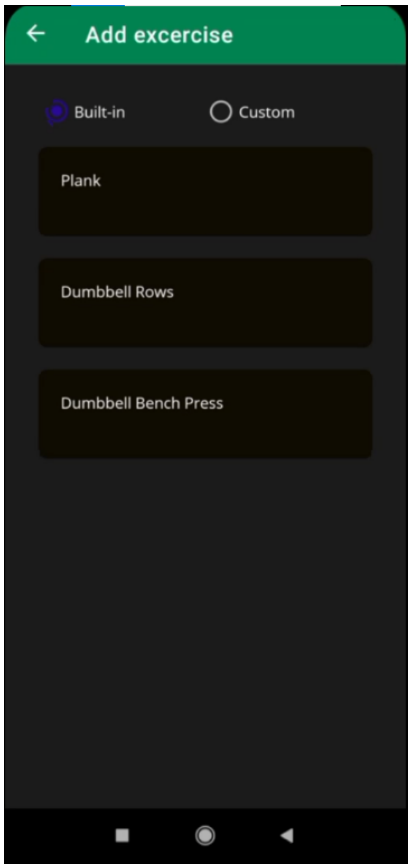


Рисунок 4.19 – Сторінка додавання вправ до тренування

Додана вправа відображається у списку (див. рис. 20).

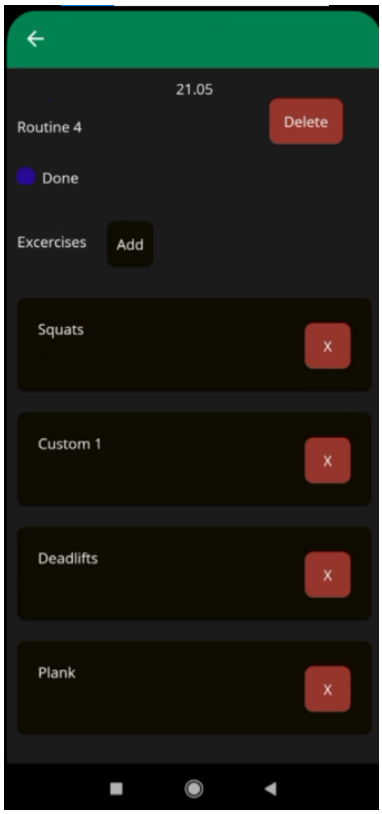


Рисунок 4.20 – Сторінка тренування із доданою вправою

Натистувши на вправу користувач переходить до сторінки редагування підходами на якій можна додавати, видаляти підходи за змінювати їх повтори та вагу або час (див. рис. 21).

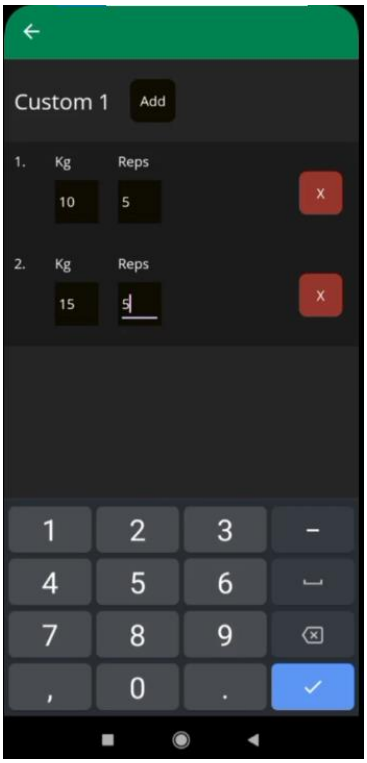


Рисунок 4.21 – Сторінка редагування підходами вправи

Висновки

Таким чином, дана робота була спрямована на проектування та розробку програмного забезпечення для моніторингу особистої спортивної діяльності.

1. Досліджено актуальність програмного продукту та її наукову новизну шляхом проведення аналізу та порівняння актуальності існуючих систем-аналогів, виявлено їх недоліки.

2. За результатами порівняння аналогів було сформулювано функціональні та нефункціональні вимоги із використання UML діаграм.

3. Обрано створити мобільний застосунок для операційної системи Android мовою C#, використовуючи платформу .NET 7 та фреймворк MAUI, архітектуру та патерни розробки Clean Architecture, MVVM, Dependency Injection та Repository Pattern, базу даних SQLite та ORM технологію Entity Framework Core, систему керування Git, IDE Visual Studio 2022 та СУБД DB Browser for SQLite.

4. Спроектовано розгортання застосунку за допомогою UML діаграм прецедентів, класів та пакетів.

5. Розроблено користувацький інтерфейс мобільного застосунку згідно із вимогами.

6. Успішно виконано мануальне тестування застосунку.

Перелік посилань

1. Exercise: Background and Benefits on Health. [Електронний ресурс]. -Режим доступу: https://www.researchgate.net/publication/356972634_Exercise_Background_and_Benefits_on_Health

2. Mental Health Conditions and Exercise. [Електронний ресурс]. -Режим доступу: https://www.researchgate.net/publication/370475339_Mental_Health_Conditions_and_Exercise

3. Overtraining Syndrome. [Електронний ресурс]. -Режим доступу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3435910/

4. The Clean Architecture—Beginner's Guide. [Електронний ресурс]. -Режим доступу: https://betterprogramming.pub/the-clean-architecture-beginners-guide-e4b7058c1165

5. Advantages of .Net Core. [Електронний ресурс]. -Режим доступу: https://techjatinder.medium.com/advantages-of-net-core-b605aa76fbb8

6. .NET dependency injection. [Електронний ресурс]. -Режим доступу: https://learn.microsoft.com/en-us/dotnet/core/extensions/dependency-injection

7. What is .NET MAUI? [Електронний ресурс]. -Режим доступу: https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui

8. SQLite As An Application File Format [Електронний ресурс]. -Режим доступу: https://www.sqlite.org/aff_short.html

9. Entity Framework Core [Електронний ресурс]. -Режим доступу: https://learn.microsoft.com/en-us/ef/core/

10. Design the infrastructure persistence layer [Електронний ресурс]. -Режим доступу: https://learn.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design

Додаток А

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІТОРИНГУ ОСОБИСТИХ СПОРТИВНИХ ТРЕНУВАНЬ МОВОЮ C#

Виконав студент 4 курсу

групи ПД-41
Луценко Іван Вікторович
Керівник роботи

старший викладач кафедри Гаманюк Ігор Михайлович

Київ – 2023

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи**

  Спрощення процесу моніторингу особистих спортивних тренувань за рахунок застосування мобільного додатку, створеного мовою C#.

- **Об’єкт дослідження**

  Процес моніторингу особистої спортивної діяльності.

- **Предмет дослідження**

  Застосунок для моніторингу особистих спортивних тренувань.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести аналіз та порівняння існуючих програмних продуктів-аналогів.
2. Визначити функціональні та нефункціональні вимоги до майбутнього застосунку.
3. Обрати архітектуру застосунку та програмні засоби розробки.
4. Спроектувати розгортання застосунку.
5. Розробити користувацький інтерфейс мобільного застосунку.
6. Виконати мануальне тестування застосунку.

АНАЛІЗ АНАЛОГІВ

	Windows Notepad	Microsoft Excel	My Workout Plan	TrainingApp
Спеціалізований інтерфейс	-	-	+	+
Пошук по даті	+	+	-	+
Додавання своїх вправ	+	+	+/- (для тренування)	+
Поділ вправ на вагу або час	+	+	-	+
Можливість видалити підхід із тренування	+	+	-	+
Можливість призначити тренування на дату	+	+	-	+
Потребує попередніх знань ПЗ	-	+	-	-

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

- пошук тренувань по даті;
- додавати свої вправи;
- використовувати та додавати вправи на вагу або час;
- видаляти підхід із тренування;
- назначати тренування на дати;

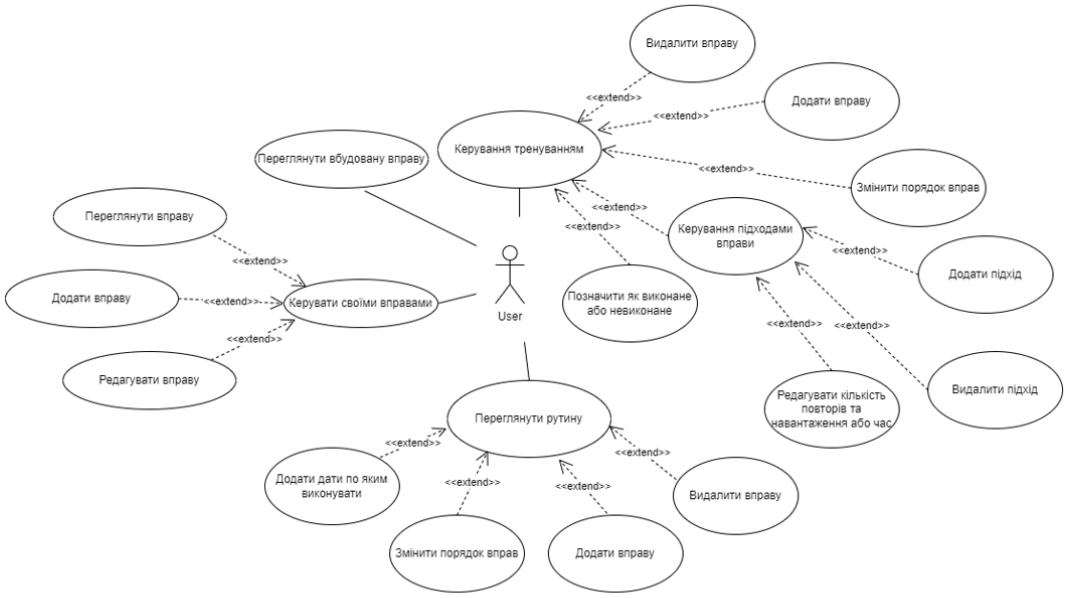
Нефункціональні вимоги:

- Операційна система – Android;

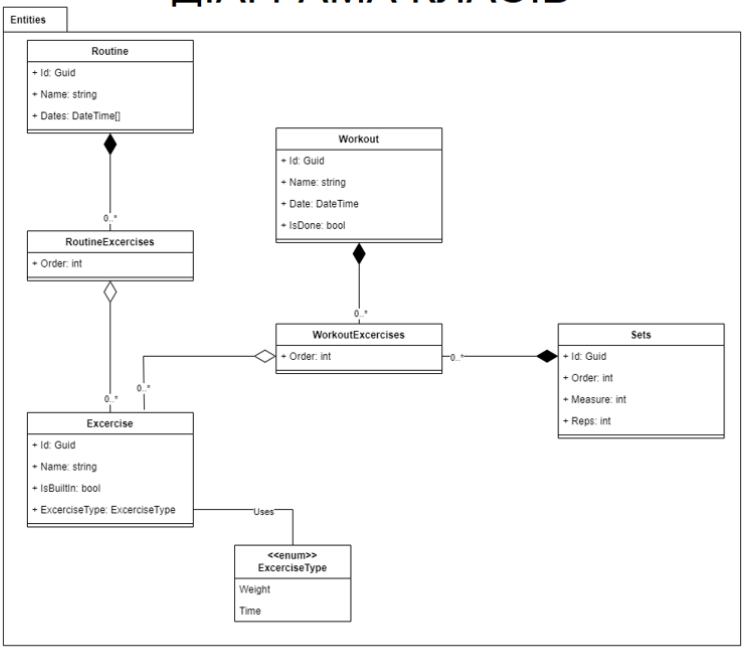
ПРОГРАМНІ ЗАСОБИ РОЗРОБКИ

Бібліотека CommunityToolkit

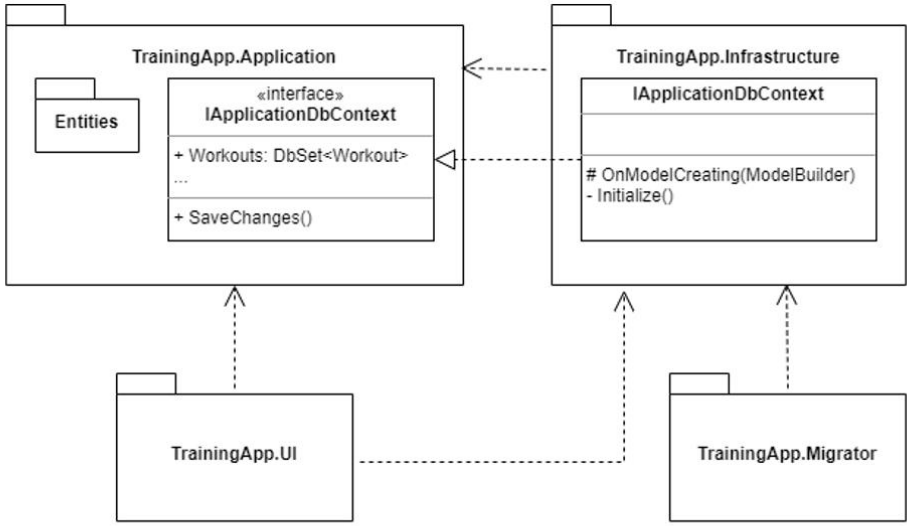
ДІАГРАМА ПРЕЦЕДЕНТІВ



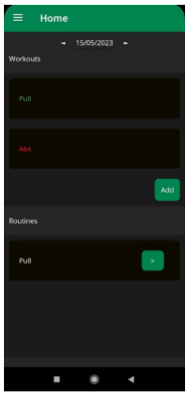
ДІАГРАМА КЛАСІВ



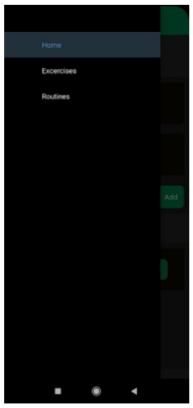
ДІАГРАМА ПАКЕТІВ



ЕКРАННІ ФОРМИ



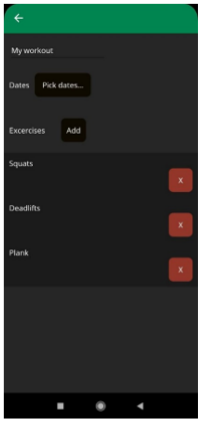
Головна сторінка застосунку



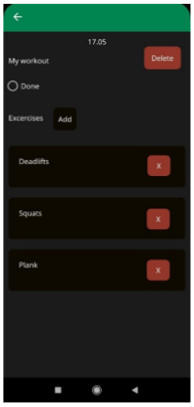
Бокове меню навігації

Сторінка вправ

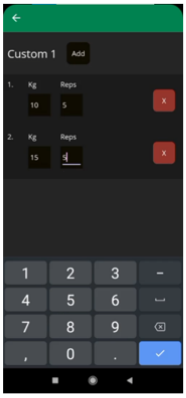
ЕКРАННІ ФОРМИ



Сторінка рутини



Сторінка тренування



Сторінка підходів

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Луценко І.В. Розробка програмного забезпечення для моніторингу особистих спортивних тренуванням з використанням UML діаграми прецедентів/І.В Луценко, І.М Гаманюк // Застосування програмного забезпечення в інфокомунікаційних технологіях: Матеріали всеукраїнської науково-технічної конференції кафедри інженерії програмного забезпечення Державного університету телекомунікацій. Збірник тез. 20.04.2023, ДУТ, м. Київ — К.: ДУТ, 2023. — С. 52.
2. Луценко І.В. Використання .NET MAUI/ І.В. Луценко, І.М Гаманюк // Застосування програмного забезпечення в інфокомунікаційних технологіях: Матеріали всеукраїнської науково-технічної конференції кафедри інженерії програмного забезпечення Державного університету телекомунікацій. Збірник тез. 20.04.2023, ДУТ, м. Київ — К.: ДУТ, 2023. — С. 104.

ВИСНОВКИ

1. Досліджено актуальність програмного продукту та її наукову новизну шляхом проведення аналізу та порівняння актуальності існуючих систем-аналогів, виявлено їх недоліки.
2. За результатами порівняння аналогів було сформулювано функціональні та нефункціональні вимоги із використання UML діаграм.
3. Обрано створити мобільний застосунок для операційної системи Android мовою C#, використовуючи платформу .NET 7 та фреймворк MAUI, архітектуру та патерни розробки Clean Architecture, MVVM, Dependency Injection та Repository Pattern, базу даних SQLite та ORM технологію Entity Framework Core, систему керування Git, IDE Visual Studio 2022 та СУБД DB Browser for SQLite.
4. Спроектовано розгортання застосунку за допомогою UML діаграм прецедентів, класів та пакетів.
5. Розроблено користувацький інтерфейс мобільного застосунку згідно із вимогами.
6. Успішно виконано мануальне тестування застосунку.

ДЯКУЮ ЗА УВАГУ!