Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут комп'ютерних наук
та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Магістерська кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему: «Розроблення мультиплатформної
системи для особистісного розвитку»

Виконав студент 6 курсу групи КН(м)-61
спеціальності
122 "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)

Бац Б. М.
(прізвище та ініціали)

Керівник Крошний І. М.
(прізвище та ініціали)

Нечепуренко А. В.
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Львів – 2024 рік

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ комп'ютерних наук та інформаційних технологій
Кафедра комп'ютерних наук
Рівень вищої освіти другий (магістерський)
Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Борецька І.Б.
"____" _____2024 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Бац Богдан Михайлович
(прізвище, ім'я, по батькові)

1. Тема роботи "Розроблення мультиплатформної системи для особистісного розвитку"

керівник роботи: *Крошний І.М., к.т.н., доцент, Нечепуренко А. В., ст. викл.*,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "13" лютого 2023 року № С-49

2. Термін подання студентом роботи "05" січня 2024 року
3. Вихідні дані до роботи документація .NET MAUI, ASP.NET Core Web API, REST, теоретичний матеріал по штучному інтелекту, Azure, MS SQL Server, Entity Framework Core, опис наукових досліджень стосовно звичок

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Розділ 1. Стан проблемної області.
Розділ 2. Інформаційне забезпечення.
Розділ 3. Математичне забезпечення.
Розділ 4. Програмне забезпечення.
Розділ 5. Розроблення стартап-проєкту.
Висновки
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):
діаграми варіантів використання, логотип розробленого додатку, схеми сутностей, архітектура Web API, реалізованих серверного та клієнтського застосунків, знімки екранів із відображенням їх роботи

6. Дата видачі завдання "15" лютого 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1.	Вивчення літератури згідно з досліджуваною темою. Збір необхідних матеріалів. Проходження виклику RD71.	20.02.23-05.03.23	виконано
2.	Розробка архітектури сервісного застосунку, визначення основних технологій та платформ.	06.03.23-18.03.23	виконано
3.	Опис і автоматизація конвенцій написання коду.	24.03.23-25.03.23	виконано
4.	Розробка першої версії схеми БД.	27.03.23-01.03.23	виконано
5.	Опис принципів функціонування програми.	03.03.23-08.03.23	виконано
6.	Реалізація логінування та реєстрації.	10.03.23-13.03.23	виконано
7.	Визначення бізнес логіки.	17.03.23-20.03.23	виконано
8.	Створення БД.	24.03.23-28.03.23	виконано
9.	Спілкування з психологом стосовно ідеї проєкту.	01.04.23	виконано
10.	Розробка діаграм варіантів використання.	03.04.23-05.04.23	виконано
11.	Створення архітектури клієнтського застосунку.	14.04.23-22.04.23	виконано
12.	Виправлення помилок та покращення у розробленій функціональності клієнтського та серверного застосунків. Оформлення першої версії пояснювальної записки.	28.04.23-16.06.23	виконано
13.	Складання плану по доробці додатків.	17.06.23-19.06.23	виконано
14.	Розміщення серверу на хмарному сервісі Azure.	20.06.23-01.07.23	виконано
15.	Вивчення комерційних шаблонів в платформі .NET MAUI. Розробка вкладки «Профіль».	03.07.23-26.08.23	виконано
16.	Реалізація вкладки «Прогрес».	29.08.23-11.11.23	виконано
17.	Розробка вкладки «Помічник».	11.11.23-25.11.23	виконано
18.	Покращення клієнтського та серверного застосунків. Виправлення помилок. Реалізація сторінки «Налаштування».	25.11.23-16.12.23	виконано
19.	Оформлення останньої версії пояснювальної записки.	16.12.23-05.01.24	виконано

Студент _____ Бац Б. М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Крошний І. М.,
(підпис) (прізвище та ініціали)

_____ Нечепуренко А. В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дипломна робота містить 109 сторінок пояснювальної записки, 88 рисунків, 1 додаток, 57 джерел.

Дипломна робота присвячена розробці кросплатформного мобільного застосунку, основною функціональністю якого є допомога користувачам дотримуватись та відстежувати позбуття чи набуття звичок. Додаток також концентрується на тому, щоб пропонувати клієнту доцільні рекомендовані звички, згенеровані штучним інтелектом. Для того, щоб користувач міг отримати різноманітну інформацію стосовно саморозвитку й інших сфер, було реалізовано асистента на основі штучного інтелекту, який може відповідати на визначені користувачем питання.

Ключові слова: C#, XAML, Entity Framework Core, Azure, MS SQL Server, DevExpress, .NET MAUI, ASP.NET Core Web API, штучний інтелект, особистісний розвиток, психологія, успіх, звичка.

ABSTRACT

The thesis contains 109 pages of explanatory note, 88 pictures, 1 appendix, 57 sources.

The thesis focuses on the development of a cross-platform mobile application whose main functionality is to help users stick to and track breaking or acquiring habits. The application also focuses on offering the client appropriate recommended habits generated by artificial intelligence. To provide the user with a variety of information regarding self-development and other areas, an AI-based assistant was implemented that can answer questions defined by the user.

Keywords: C#, XAML, Entity Framework Core, Azure, MS SQL Server, DevExpress, .NET MAUI, ASP.NET Core Web API, artificial intelligence, personal development, psychology, success, habit.

ТЕХНІЧНЕ ЗАВДАННЯ

Необхідно дослідити й розробити кросплатформний додаток для особистісного розвитку:

- визначити найосновнішу функціональність корисну й важливу для користувачів;
- дослідити алгоритми розвитку, їх переваги та недоліки, вибрати з них найефективніший та реалізувати в програмі;
- створити базу даних на сервері;
- розмістити сервер на хмарному сервісі Azure;
- надати користувачам можливість створювати власні звички;
- реалізовувати рекомендаційний центр на основі штучного інтелекту для генерації рекомендованих звичок у відповідності до даних, особливостей та вимог користувача;
- розробити відслідковувач набуття звичок клієнта;
- реалізовувати віртуального помічника на основі штучного інтелекту, який може надавати різноманітну інформацію відповідно до запитів;
- створити графічний мультиплатформний інтерфейс для заданої системи;
- провести тестування розробленого застосунку.

ПЕРЕЛІК СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ШІ – Штучний Інтелект;

ПЗ – Програмне Забезпечення;

ВВ – Варіант Використання;

API - Application Programming Interface (Інтерфейс Прикладного Програмування);

ASP - Active Server Pages (Активні Сторінки Сервера);

EF - Entity Framework (Фреймворк Сутностей);

VS - Visual Studio (Візуальна Студія);

IIS - Internet Information Services (Інформаційні Послуги в Інтернеті);

GUID - Global Unique Identifier (Глобальний Унікальний Ідентифікатор);

JSON - JavaScript Object Notation (Нотація Об'єктів JavaScript).

ЗМІСТ

ВСТУП

- Актуальність проблеми

Особистісний розвиток набуває все більшої й більшої популярності, оскільки саме характер визначає якість життя та чи досягає людина своїх мрій. Але через те, що ми живемо в ері перенасиченої інформації, утворився хаос в принаймні двох речах: покращення якості життя та розвиток особистості. Це спричиняє постійні перегорання, невдоволення (інколи навіть собою) та стрес. Розроблений застосунок допоможе уникати цього й досягати тих вершин, які користувачі давно прагнули.

- Мета роботи

Розробити кросплатформний додаток, який допомагатиме користувачу в досягненні масштабних цілей. Застосунок націлений на те, щоб підтримувати, підказувати та направляти користувача на ефективні шляхи покращення його характеру та особистості, а, отже, реалізації його прагнень та потенціалу.

- Об’єкт дослідження

Систематизація покращення звичок користувача, в результаті дотримання яких клієнт досягає свої далекоглядні цілі.

- Предмет

Особливості реалізації системи для допомоги поліпшення звичок користувача у відповідності до його потреб, запитів та прагнень. Визначення деталей кросплатформної та комерційної розробки застосунку.

- Завдання

До виконання в межах роботи були поставлені наступні основні завдання.

1) провести комунікацію з потенційними клієнтами та психологом на предмет ідеї додатку;
2) виконати огляд літератури по тематиці психології, створення мультиплатформного застосунку, штучного інтелекту, конкурентного та веб-програмування;
3) реалізувати схему БД;

4) визначити та дослідити найважливіші та найефективніші психологічні принципи та алгоритми послідовного розвитку, набуття звичок, досягнення далекоглядних цілей;
5) розробити можливість спілкування з помічником, який створений на основі штучного інтелекту;
6) імплементувати кросплатформний сервер та розмістити його у хмарі;
7) реалізувати сервіс на основі штучного інтелекту, який генеруватиме рекомендовані звички відповідно до запитів клієнта;
8) створити графічний мультиплатформний інтерфейс для заданої системи.

- Наукова новизна роботи

Наукова інноваційність даного застосунку полягає в тому, що він єдиний, що систематизує саме особистісний розвиток користувача, надаючи при цьому змогу спілкуватись з віртуальним помічником для отримання різноманітної інформації. Окрім цього, він рекомендує звички користувачу на основі його даних, особливостей та запиту. Більшість додатків даної тематики концентруються лише на мотивації, візуалізації та відслідковуванні звичок, проте зрідка можна побачити використання штучного інтелекту та системності. Вони можуть містити багато різноманітної інформації, проте вона не систематизована таким чином, щоб мати змогу послідовно, поступово, постійно й з бажанням розвиватись. Часто стається лише навпаки й користувач зовсім перестає цілеспрямовано працювати над своїм життям.

- Практична значимість

Оскільки попит на даного роду додатків стає все більшою й більшою, згідно з рейтингом Google [1, 2], а важливість чіткості та рекомендацій (що забезпечує поточний застосунок) ніколи не впаде на низький рівень, то цей додаток зможе значно допомогти користувачу поступово, постійно й з бажанням розвиватись. Саме прагнення особистісного покращення є найважливішим для досягнення цілей. В результаті клієнти зможуть дійсно побудувати життя своєї мрії.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Аналіз предметної області

Коли вийшла перша книга по особистісному розвитку «Думай і багатій» Наполеона Гілла 1937 року людство почало зовсім по-іншому дивитись на світ і в результаті останній дуже сильно змінився. Багато інновацій, зародок і розвиток ІТ, безліч статей та книг на теми психології й особистісного зростання. Усе це неминуче привело до справжнього хаосу у цих сферах. Одна ідея повністю суперечить іншій, ніхто не навчає цілісного бачення світу і як його здобути. Деякі говорять про певні універсальні методи, що працюють у будь-якій сфері. Проте це, як і в розробці ПЗ, в більшості випадків є недоречним, оскільки кожна з них є унікальною.

Утворений хаос досить серйозно шкодить людям:

- більше вчить привчатись до думки інших, аніж Творця чи свого серця;
- створює перегорання через зміну усього й одразу;
- через відсутність чіткої системи розвитку, часто стаються розчарування, інколи навіть собою;
- більшість лекторів чи авторів намагаються радити суто те, що працює для них, забуваючи, що кожна людина унікальна. В кожного може бути свій фундамент, який надає змогу стійко стояти на ногах і проходити крізь випробування. Проблема в тому, що це рідко дають зрозуміти й не вчать більше прислуховуватись до внутрішнього голосу, аніж до зовнішнього. Хоча це й логічно, оскільки надасть змогу довше зберегти клієнтів.

Саме через те, що ці наслідки хаосу було пережито на власному досвіді, вирішено створити такий додаток, який не буде повністю навіювати свою ідеологію, а навпаки – допоможе користувачу розробити свою.

Оглянемо роль психології в досягненні цілей. Беззаперечно, основною мотивацією саморозвитку є досягнення того, чого люди насправді хочуть. Очевидно, щоб дойти до певної мети, потрібно стати тим, хто може це зробити. Саме тому, для отримання бажаного слід наполегливо працювати над своєю

психологією, тобто особистими якостями, принципами, звичками - тим, що буде непохитним при будь-яких випробуваннях.

Проте користувачу також варто розуміти, коли він готовий до наступного кроку, тобто коли він знає, що певні звички є вже автоматизованими й слід починати працювати над наступними звичками. Щоб це визначити, необхідно зважити різні фактори, серед яких зазвичай включать наступні.

- Сумарна кількість неавтоматизованих звичок, над якими зараз працює користувач;
- загальна їх складність;
- сумарне значення частоти їх виконання.

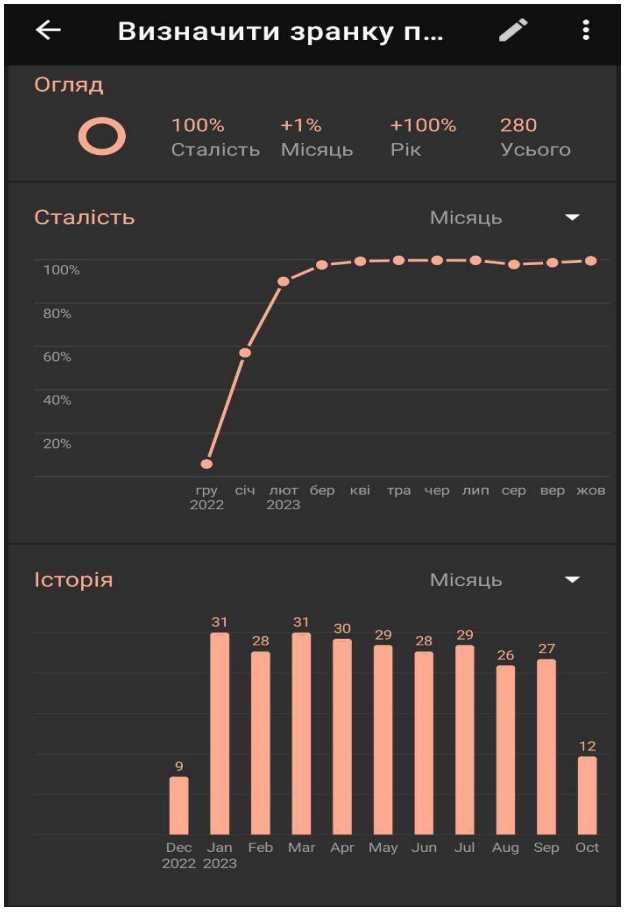

Рис. 1.1.1. Приклад прогресу автоматизації власної звички

Отже, задля цього застосунок має вміти визначати відповідний прогрес кожної зі звичок відповідно до їхнього дотримання чи занедбання, знати складність частоту кожної з них. Щоб обчислювати відсоток автоматичності додатку слід розглядати певний шлях роботи над нею та включати кожен день, коли звичка мала бути дотримана.

1.2. Штучний інтелект і особистісний розвиток

Штучний інтелект має потенціал революціонізувати підхід до особистого розвитку [3] й багатьох інших сфер. Чи то через персоналізовані навчальні програми, чи то через інструменти самовдосконалення, ШІ може допомогти людям визначити свої нові звички та працювати над їх автоматизацією більш ефективно та результативно. Це неминуче приведе до великих результатів в майбутньому.

Рис. 1.2.1. Концепція розвитку ШІ

Однією з головних переваг використання ШІ для особистого розвитку є його здатність надавати персоналізовані рекомендації. Маючи доступ до величезних обсягів даних і можливість аналізувати індивідуальні вподобання та потреби, ШІ може створювати індивідуальні навчальні програми, які відповідають цілям і завданням людини. Наприклад, навчальна платформа зі

штучним інтелектом може рекомендувати конкретні курси або ресурси на основі інтересів і кар'єрних прагнень людини. Це може допомогти людям залишатися вмотивованими та залученими, оскільки вони можуть досягати своїх цілей у спосіб, що відповідає їхнім унікальним потребам та вподобанням.

Інший спосіб використання ШІ для особистого розвитку - це розробка інструментів і додатків для самовдосконалення. Наприклад, існують програми для усвідомленості та медитації на основі штучного інтелекту, які можуть допомогти людям зменшити стрес і покращити загальний стан. Ці програми часто використовують алгоритми машинного навчання для аналізу голосу людини і надають персоналізовані відгуки та рекомендації щодо покращення психічного здоров'я. Крім того, інструменти для підвищення продуктивності на основі штучного інтелекту можуть допомогти людям керувати своїм часом і більш ефективно розставляти пріоритети у виконанні завдань. Аналізуючи робочі звички та патерни людини, ці інструменти можуть надавати персоналізовані пропозиції щодо підвищення продуктивності та досягнення цілей.

Однак важливо зазначити, що ШІ не замінює людську взаємодію та підтримку, тому досягати успіху слід разом принаймні з однією людиною. Хоча ШІ може бути цінним ресурсом для особистого розвитку, в кінцевому підсумку саме особа має діяти й вносити необхідні зміни у власне оточення та для автоматизації своїх корисних звичок, а отже, й для досягнення своїх далекоглядних цілей. Важливо також пам'ятати про потенційні етичні наслідки використання ШІ для особистого розвитку, такі як занепокоєння щодо конфіденційності. Саме тому дані користувача ніде не мають використовуватись поза застосунком і йому слід мати змогу видалити непотрібні звички й історію спілкування з ШІ.

ШІ в сучасному світі активно використовується для персоналізації користувацького досвіду та генерації рекомендацій. За допомогою алгоритмів машинного навчання, ШІ аналізує величезні обсяги даних, зібраних від користувачів, для розуміння їхніх цілях, звичок та потреб. Базуючись на цих відомостях, системи ШІ можуть індивідуалізувати рекомендації, пропонуючи

користувачам контент, який найбільше відповідає їхнім особистим потребам та очікуванням. Аналізуючи дані користувача, система визначає його унікальні потреби, звички та цілі. На основі цього аналізу ШІ розробляє та рекомендує індивідуально підходящі звички, сприяючи покращенню ефективності та задоволенню користувача. Цей підхід дозволяє створювати персоналізовані стратегії для досягнення поставлених цілей, сприяючи збалансованому та здоровому способу життя.

1.3. Висновок до розділу

З врахуванням попередньо описаної інформації, ШІ має потенціал, щоб значно покращити підхід до особистого розвитку. Завдяки персоналізованому навчанню та інструментам самовдосконалення, ШІ може допомогти людям визначити свої звички та працювати над їх автоматизацією більш ефективно та результативно. Проте в цей же час слід розробити оптимальний алгоритм визначення прогресу звички, щоб користувач міг знати, чи слід переходити до наступної.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Глосарій

Глосарій – це словник термінів, абревіатур, власних назв, слоганів і скорочень, характерних для певної галузі або проєкту [8]. При розробці діаграм варіантів використання (див. підрозділ «Вимоги до програмного забезпечення») туди прийнято включати всі поняття, що в них використовуються.

Таблиця 1

Глосарій

Назва	**Значення**
Користувач	Клієнт, що хоче покращити свій характер та досягти далекоглядні і непрості цілі.
Дані застосунку	Модель штучного інтелекту, опис дій користувача та база даних (БД).
Атомний	Джерело великої енергії чи сили; властивий найменшій частинці речовини; відповідає неподільній одиниці більшої системи.
Звичка	Автоматична дія чи реакція на певну ситуацію. Виконується регулярно. Може бути наступних типів. • Незмінна (слідуємо завжди без жодних винятків); • Ціломудрена. Тобто та, яку ми дотримуємось за будь-яких умов, але винятки можливі - лише в тих випадках, коли виникає конфлікт між звичками. Тоді варто вибирати, який для нас є більш пріоритетною. Існують також дні відпочинку, тоді ми, звісно, не використовуємо звички, які напряму стосуються нашої праці.

Принцип	Принципи – це основа того, ким клієнт хоче стати, як особистість; глибоко закріплене переконання або система цінностей, які визначають ваші дії, вибори та погляди на життя. Їх використання планується додати в майбутніх версіях додатку. Їхні види ідентичні типам звичок.
Реєстрація клієнта	Введення користувачем своїх особистих даних (наприклад, паролю, електронної пошти й т.д.).
Логінування клієнта	Авторизація та вхід користувача в систему. Тобто введення email та паролю.
Вікно з коротким описом, що таке принципи та важливість їхнього набуття	Відкривається після першого входу користувача при відкритті сторінки вибору сфер життя. Потім подібний текст можна переглянути у вкладці «Помічник». Планується додати в майбутні версії застосунку.
Сторінка вибору сфер життя	Тут користувач вибирає ті, що є для нього важливими й в яких хоче будувати свої принципи. Планується додати в майбутні версії застосунку.
Вікно опису сторінки вибору цілей.	Наголошується їхня нестабільність та важливість. Потім подібний текст можна переглянути у вкладці «Цілі». Планується додати в майбутні версії застосунку.
Сторінка вибору цілей	Відкривається одразу після вибору сфер життя, якщо це є перший вхід користувача. Вибирається користувачем для кожної вибраної попередньо сфери. Це довгострокові цілі. Їхній дедлайн – мінімум через 1 місяць. Планується додати в майбутні версії застосунку.
Вікно опису сторінки вибору принципів.	Описується найважливіша інформація про них. Планується додати в майбутні версії застосунку.

Сторінка вибору поточних принципів користувача	Відкривається одразу після вибору цілей, якщо це є перший вхід користувача. Містить перелік різних принципів, які за замовчуванням є в додатку та згруповані по папкам, що відповідають важливим сферам клієнта. Також існує можливість створити користувацький принцип, який буде видимий іншим клієнтам, лише якщо розробник його додасть у відповідну таблицю БД. Планується додати в майбутні версії застосунку.
Вкладка «Принципи»	Представляються сфери життя у вигляді папок та принципи, що відповідають їм. Вони(принципи) можуть мати підлеглі звички, тобто ті, що допомагають дотримуватись їх. Тут є можливість відкрити сторінку створення нового принципу. Планується додати в майбутні версії застосунку.
Вкладка «Прогрес»	Відповідає за представлення атомних звичок користувача та їхнього прогресу.
Вкладка «Помічник»	Чат з асистентом, створеного на основі ШІ. Можна задавати йому різноманітні питання. Також в майбутньому тут планується представляти для вибору часті запитання на тему звичок, принципів, успіху, функціональності додатку та саморозвитку (наприклад, правило 3-х П (поступовість, послідовність, постійність)).
Вкладка «Профіль»	Надає можливість встановити свій псевдонім, основне гасло та місію. В майбутньому тут планується додати вибір певного персонажа, поряд з яким відображається те, який в нього рівень і

	скільки ще залишилось, щоб перейти на наступний. Це так звана гейміфікація застосунку.
Вкладка «Цілі»	Перелік сфер та цілей, що належать їм.
Адміністратор	Розробник застосунку.

2.2. Назва та логотип додатку

Оскільки застосунок концентрується на допомозі створення, зберігання та відстежування звичок користувача, то було вирішено назвати його «**Атомні Звички**», оскільки існує дуже популярна книга з ідентичною назвою [50], причому на останню немає авторських прав. Оскільки однією з вимог додатку є підтримка англійською та української мов, то відповідно його назва теж має бути локалізована. На першій мові вона буде звучати «Atomic Habits». З точки зору візуального представлення це також дуже зручно, через те що на обох мовах обидва слова є однакової довжини.

Іконка логотипу виглядає наступним чином.

Рис. 2.2.1. Логотип застосунку «Атомні Звички»

Було вибрано саме цей, оскільки він дещо нагадує атом і має привабливе візуальне представлення.

2.3. Схема сутностей

Для побудови БД було намальовано схему сутностей (entity), оскільки саме з них Entity Framework Core генеруватиме таблиці БД. Для цього використовувався редактор StarUML [34] та діаграма класів (тип статичної структурної діаграми, яка описує структуру системи, показуючи її класи, їхні атрибути, операції (або методи) та зв'язки між об'єктами [35]), що надало змогу згенерувати C#-код засобами програми.

Для позначення полів, що можуть мати NULL значення або мати 0 елементів (якщо це стосується колекцій), використовується стереотип Nullable. Для тих полів, що є перечисленнями, застосовується – enum. У StarUML сутності, які використовуються клієнтським додатком виглядають наступним чином.

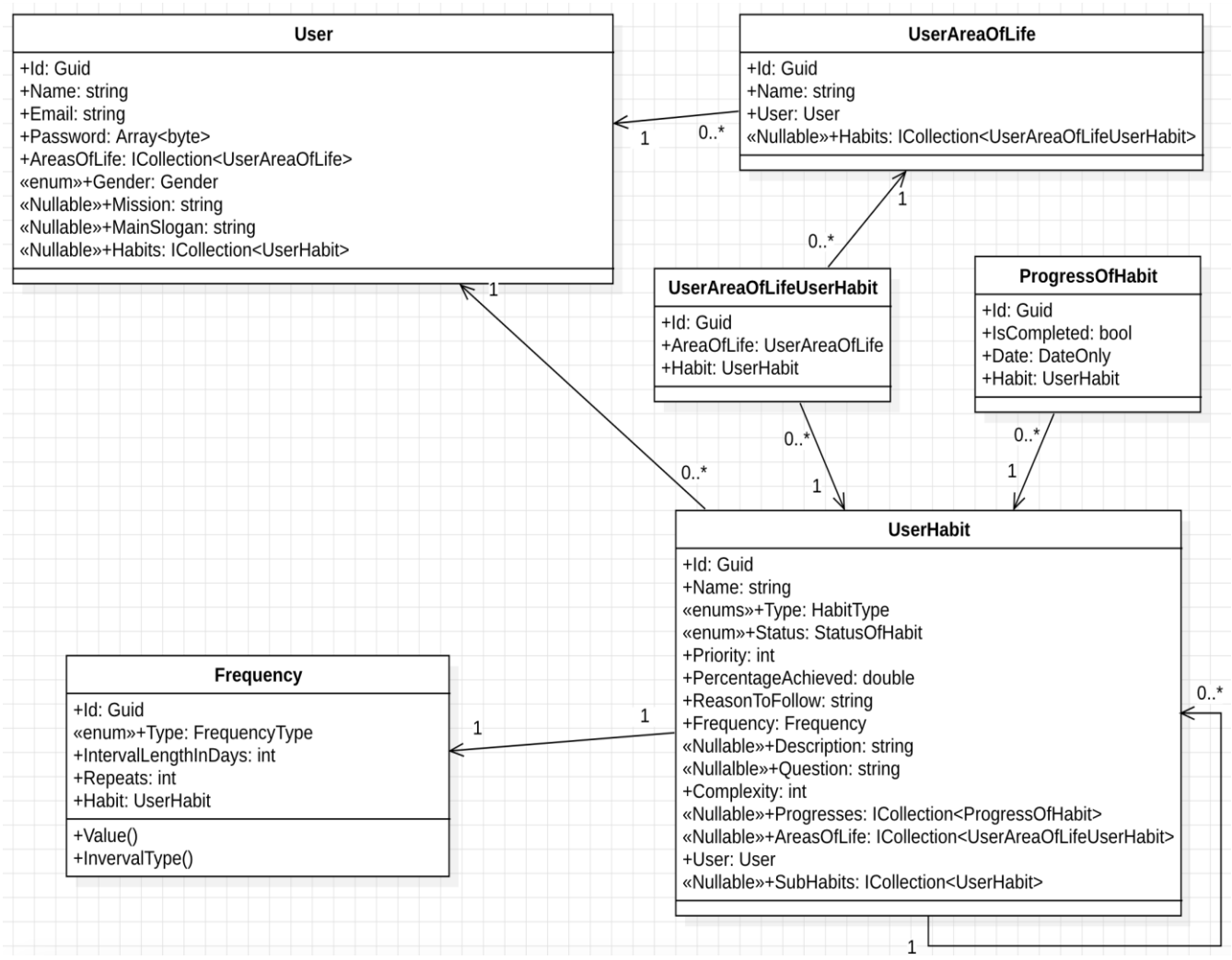

Рис. 2.3.1. Схема активних сутностей

Як можна помітити, для ідентифікаторів записів застосовується тип Guid (128-бітний унікальний текстовий рядок [40]). Цьому передували кілька причин.

1) Дозволяє легко розподіляти БД між декількома серверами;
2) можна генерувати ідентифікатори будь-де, замість того, щоб звертатися до БД, якщо тільки не потрібна часткова послідовність (наприклад, за допомогою newsequentialid() в SQL Server);
3) більшість сценаріїв реплікації вимагають стовпців GUID;
4) унікальний для кожної таблиці, кожної бази даних і кожного сервера.

Опишемо найважливіші з вище представлених сутностей.

1) User – зберіає дані користувача або посилання на них. Варто відзначити, що кожен клієнт має наступні важливі дані.
    - Множину різних сфер життя (AreasOfLife), які на даному етапі формуються за замовчуванням і огортають всі можливі;
    - Звички (Habits), на відстежуванні та допомозі слідувати яких додаток саме й концентрується;
    - Стать (Gender); Місію (Mission), тобто ціль на все життя; основний слоган по життю (MainSlogan), які мотивують користувача та надають змогу системі штучного інтелекту генерувати більш доцільні рекомендовані звички.
2) UserAreaOfLife – містить дані про сферу життя користувача. На даний час це її назва та звички (Habits). Генеруються вони за замовчуванням.
3) UserHabit – ієрархічна сутність з даними про звички користувача.
4) ProgressOfHabit – зберігає дані про те, чи звичка була дотримана в конкретний день.

Також було визначено перші версії сутностей, які плануються в майбутньому використовувати в клієнтському застосунку для задоволення вимог. Розглянемо спершу програми розвитку.

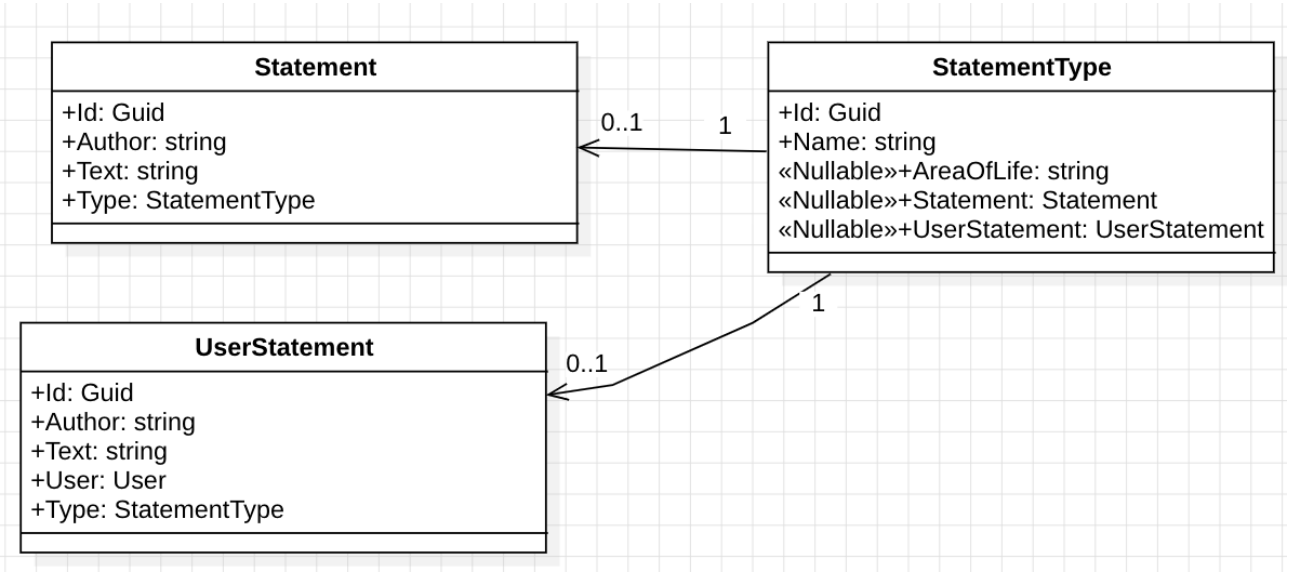

Рис. 2.3.2. Таблиці, що напряму стосуються виразів

Це так звані вислови, тобто мотивації, цитати і т.д. За допомогою поля Type визначатимуться, з якої вибірки визначати, який вислів представляти користувачу.

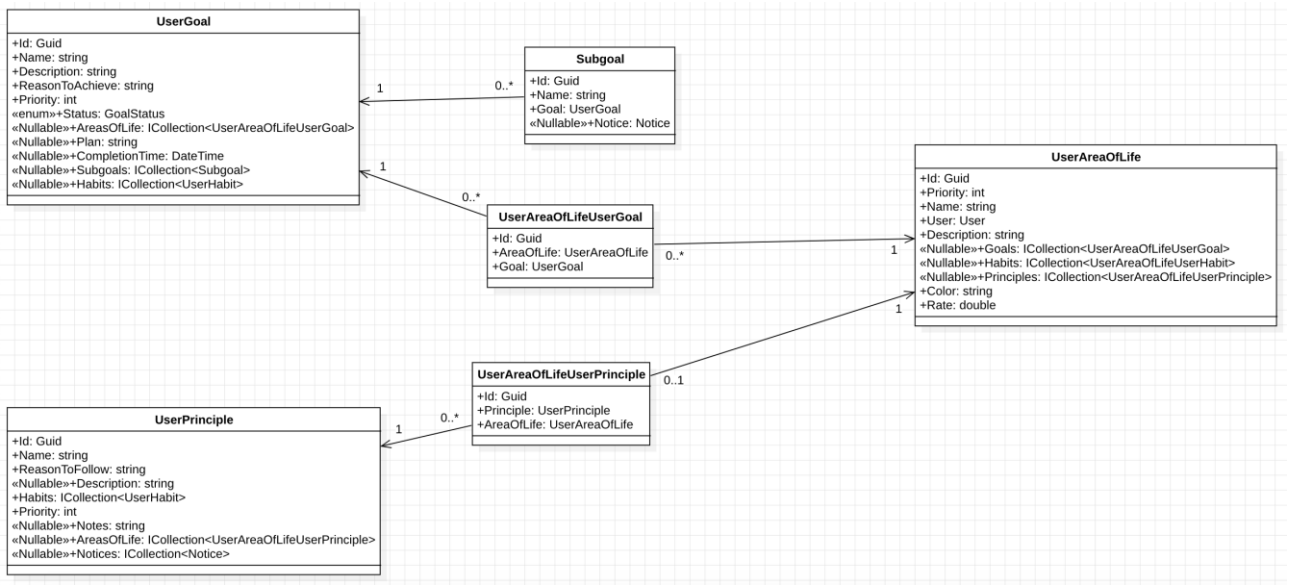

Рис. 2.3.3. Сутності цілей та принципів користувача

Як бачимо, в кожній сфері життя припустимий набір принципів та цілей. Кожна з останніх може мати набір звичок, які допомагатимуть їх досягнути. Цілі також можуть бути поділені на підцілі з коротким описом.

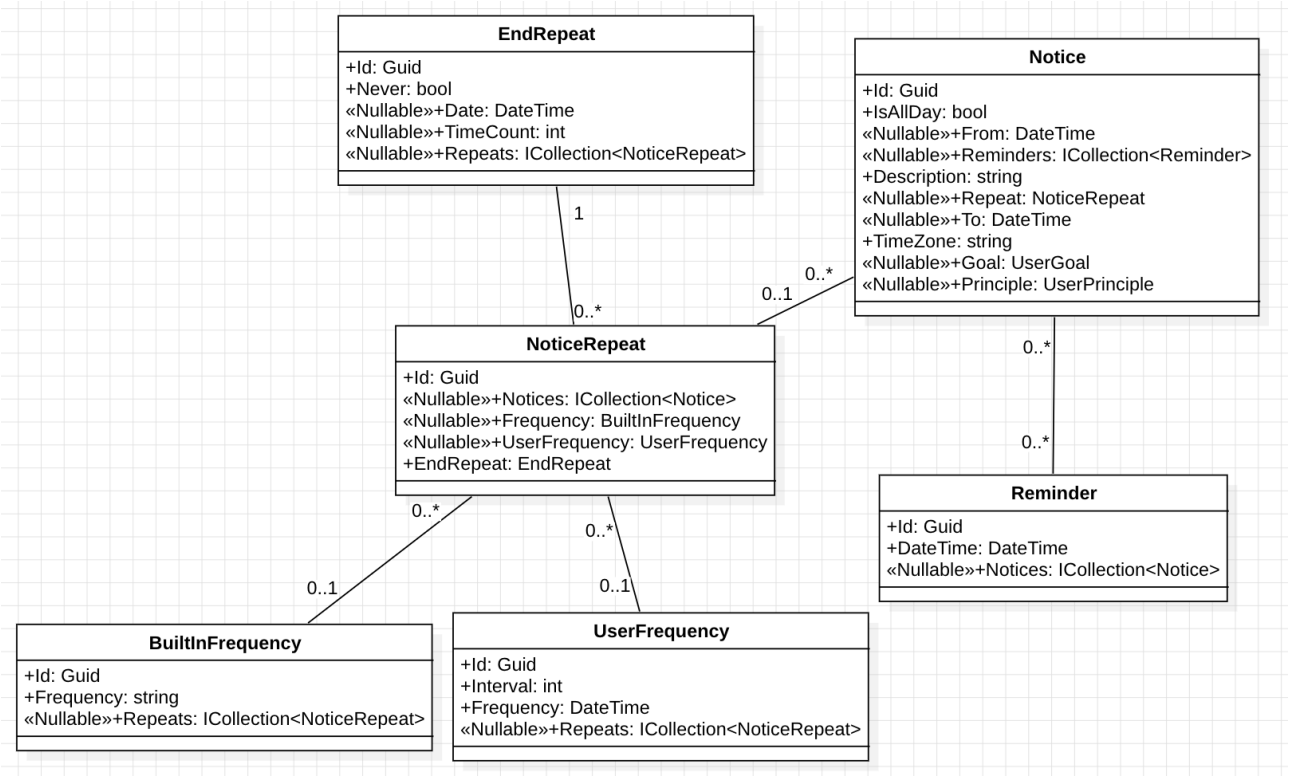

Рис. 2.3.4. Сутності, що пов'язані з нагадуваннями

Тут представляються таблиці для створення нагадувань користувачем. Вони використовуються в цілях, для створення плану їх досягнення й принципах для нагадування їх виконання. Таку схему було створено на основі того, як реалізований з Google-календар.

2.4. Принципи функціонування проєкту

Принцип функціонування проєкту в програмуванні можна описати узагальненою послідовністю дій, які виконуються для створення, розробки та реалізації програмного продукту. Їх описує архітектура.

Архітектура ПЗ – спосіб структурування програмної або обчислювальної системи, абстракція елементів системи на певній фазі її роботи [8].

Основні принципи функціонування, які можуть бути використані при проектуванні архітектури проекту, включають такі:

Модульність: Принцип модульності вказує на те, що систему слід розбивати на окремі модулі або компоненти, які виконують певні функції. Це дозволяє забезпечити легкість розробки, тестування та підтримки системи, а також спрощує її масштабування.

Розширюваність: Принцип розширюваності означає, що система повинна бути гнучкою та легко розширюватися, додавати нові функції або змінювати існуючі. Це досягається шляхом використання модульної структури та стандартизованих інтерфейсів.

Відновлюваність: Принцип відновлюваності передбачає, що система повинна бути стійкою до відмов та здатною до відновлення після збоїв. Для цього можуть використовуватися механізми резервного копіювання, реплікації даних, забезпечення високої доступності тощо.

Ефективність: Принцип ефективності вказує на необхідність оптимізації роботи системи та використання ресурсів (таких як час виконання, пам'ять, пропускна здатність) для досягнення максимальної продуктивності та швидкодії.

Безпека: Принцип безпеки ставить у центрі уваги захист системи від несанкціонового доступу(одержання (добування) інформації, що охороняється, незаконним протиправним шляхом [9]).

За основу розробки архітектури було вибрано клієнт-серверну, оскільки саме вона слугує оптимізацією пам’яті на клієнті, що є важливим чинником в мобільних застосунках через їхній невеликий об’єм.

Архітектура клієнт – сервер (client-server architecture) – це концепція інформаційної мережі, в якій основна частина її ресурсів зосереджена в серверах, обслуговуючих своїх клієнтів [10].

Клієнт – це пристрій на стороні користувача, який запитує сервер для отримання певних даних або викоання деяких дій.

Сервер – це більш потужний комп’ютер або пристрій, що забезпечує клієнта певними послугами.

2.5. Вимоги до програмного забезпечення

Вимоги до програмного забезпечення – набір вимог щодо властивостей, якості та функцій програмного забезпечення, що буде, є чи планується розробити. Вимоги визначаються в процесі аналізу вимог та фіксуються в специфікації вимог, діаграмах прецедентів(варіантів використання; див. опис нижче) та інших артефактах процесу аналізу та розробки вимог. Насамперед

саме їх варто визначити, для того щоб в подальшому значно зменшити кількість змін до існуючого коду.

Було вирішено розробити саме діаграми варіантів використання. Проте виконанню цього завдання передував опис глосарію, оскільки саме він допоможе якісно це зробити (див. відповідний підрозділ в «Інформаційному забезпеченні»).

Діаграма прецендентів(ВВ) – поведінкова діаграма UML(Unified Modeling Language - уніфікована мова моделювання), яка представляє загальну картину того, як програма буде застосовуватись [37]. Вона є найкориснішою при висвітленні функціональних особливостей програм для людей, що не мають глибоких знань в IT галузі.

Діаграми - це графічні представлення структури або функціоналу програмної системи, на яких використовуються стандартні графічні зображення [37].

Оскільки для моделювання системи було прийнято використовувати діаграми варіантів використання, то методом моделювання є формалізація.

Формалізація використовується як метод теоретичного дослідження об’єктів шляхом відображення їх структури за допомогою символів при використанні формальних мов (мова математики, хімії, фізики і т.д.). У порівнянні з іншими методами теоретичного дослідження формалізація має ряд беззаперечних переваг:

- лаконічність і чіткість представлення об’єктів і їх властивостей;
- забезпечення єдності підходів до вирішення різних проблем;
- однозначність інтерпретації символіки;
- можливість створювати символьні моделі і вивчати реальні об’єкти і їх властивості на отриманих моделях.

Отож, опишемо розроблену модель ВВ, тобто діаграми прецендентів.

Її було побудовано в додатку IBM Rational Software Architect (RSA). В «Оглядачі проєктів» модель виглядає так:

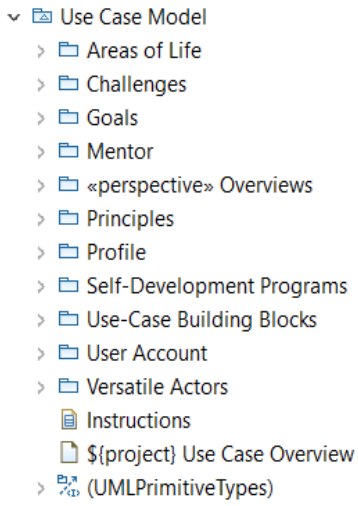

Рис. 2.5.1. Вигляд моделі варіантів використання в «Оглядачі проєктів»

Усі пакети стосуються власне моєї магістерської роботи, окрім ««perspective» Overviews»(документація по BB моделях в RSA), «Use-Case Building Blocks»(містить елементи, на основі яких варто створювати власні), «Versatile Actors» (універсальні актори, тобто ті, що використовуються в кількох пакетах; в даному випадку там містяться Admin та User).

Пакети описуються в тому порядку, в якому вони будуть доступні користувачу.

Пакет – це набір взаємопов'язаних елементів, як-от BB, акторів, зв'язків, діаграм(зазвичай лише одна).

1. «Profile» (профіль)

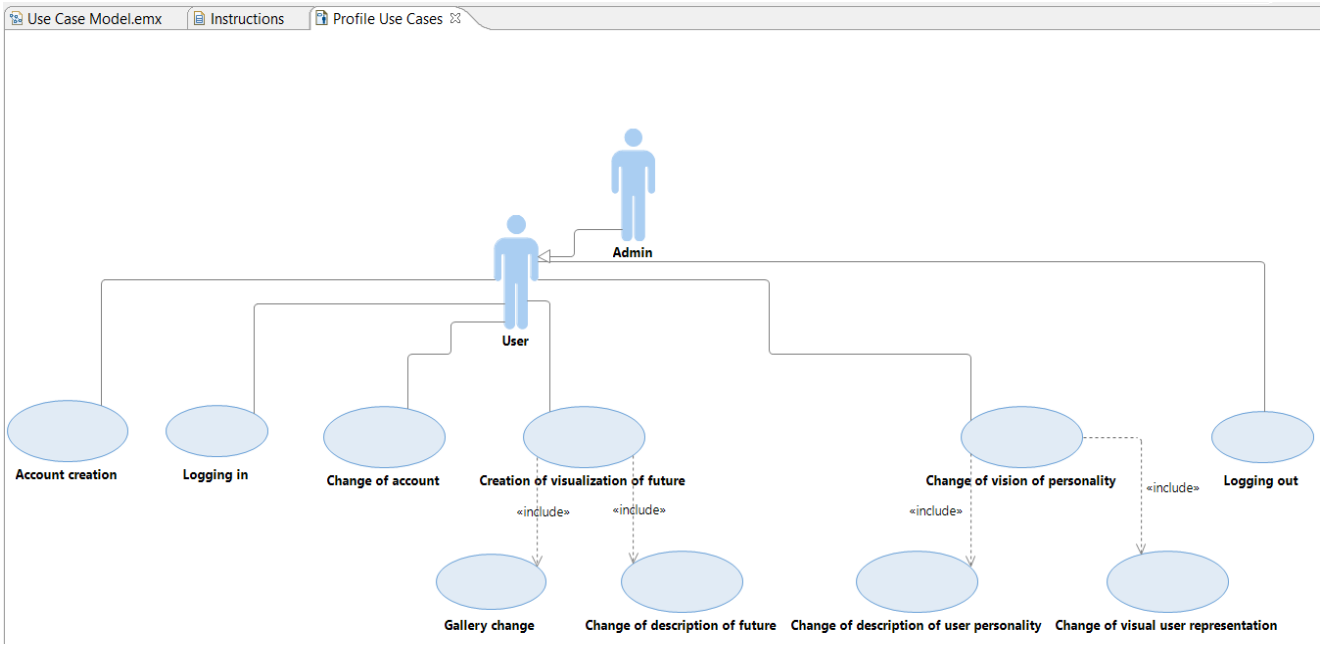

Рис. 2.5.2. Діаграма BB «Profile»

Як бачимо, «Admin» успадковує усі дані актора «User»(користувач), який в свою чергу асоціюється з різноманітними ВВ, які описуються за допомогою віддієслівного іменника. Прецедент «Creation of visualization of future» включає в себе такі ВВ, як «Gallery change» (в галереї буде міститись набір фото для візуалізації майбутнього) та «Change of description of future», який відповідає за відповідний текстовий опис. Прецедент «Change of vision of personality» створений ідентично.

2. «Areas of Life» (сфери життя)

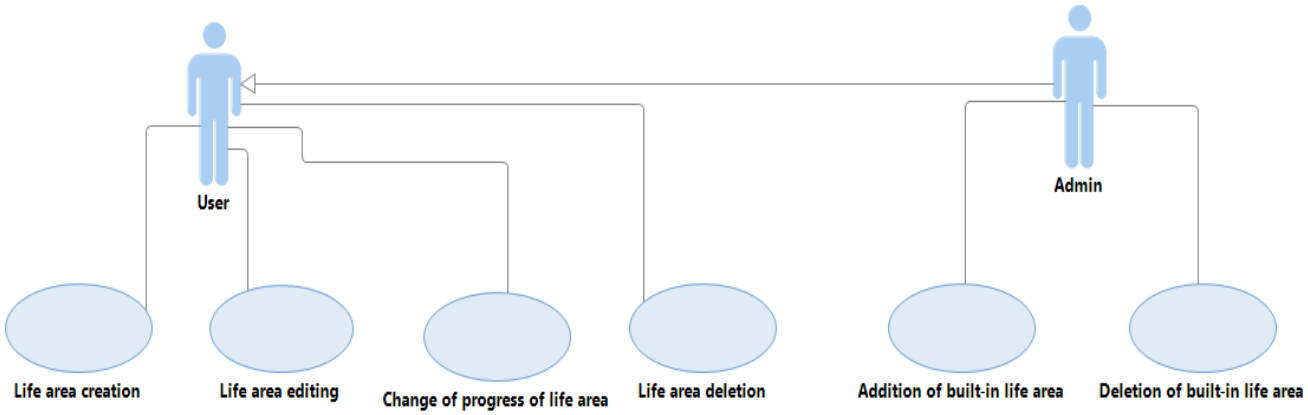

Рис. 2.5.3. Діаграма ВВ «Areas of Life»

Як бачимо, в даному випадку «Admin» також має додаткові функції, тобто це додання та видалення вбудовної сфери життя(built-in life area), тобто тої, що буде доступна користувачам для вибору. Аналогічно є в пакетах (Principles, Challenges та Self-Development Programs).

3. Goals (цілі)

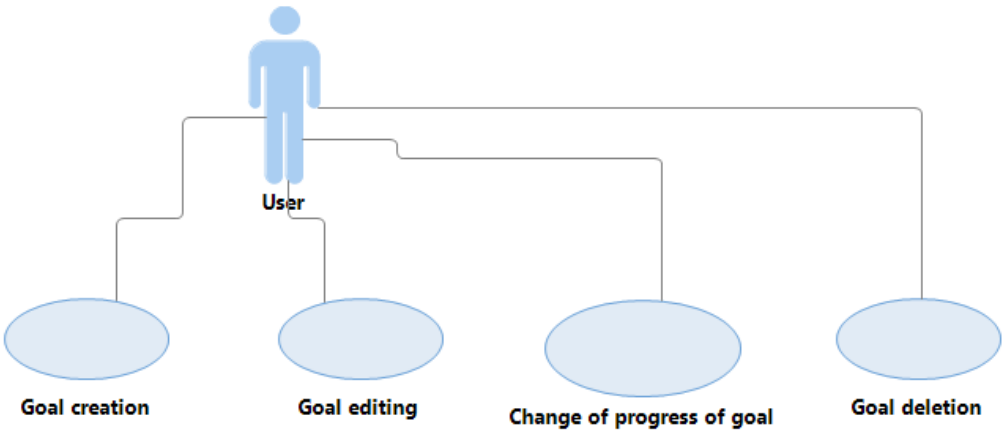

Рис. 2.5.4. Діаграма ВВ «Goals»

Це більш простий пакет, оскільки цілі в кожного клієнта різні.

4. «Principles» (принципи)

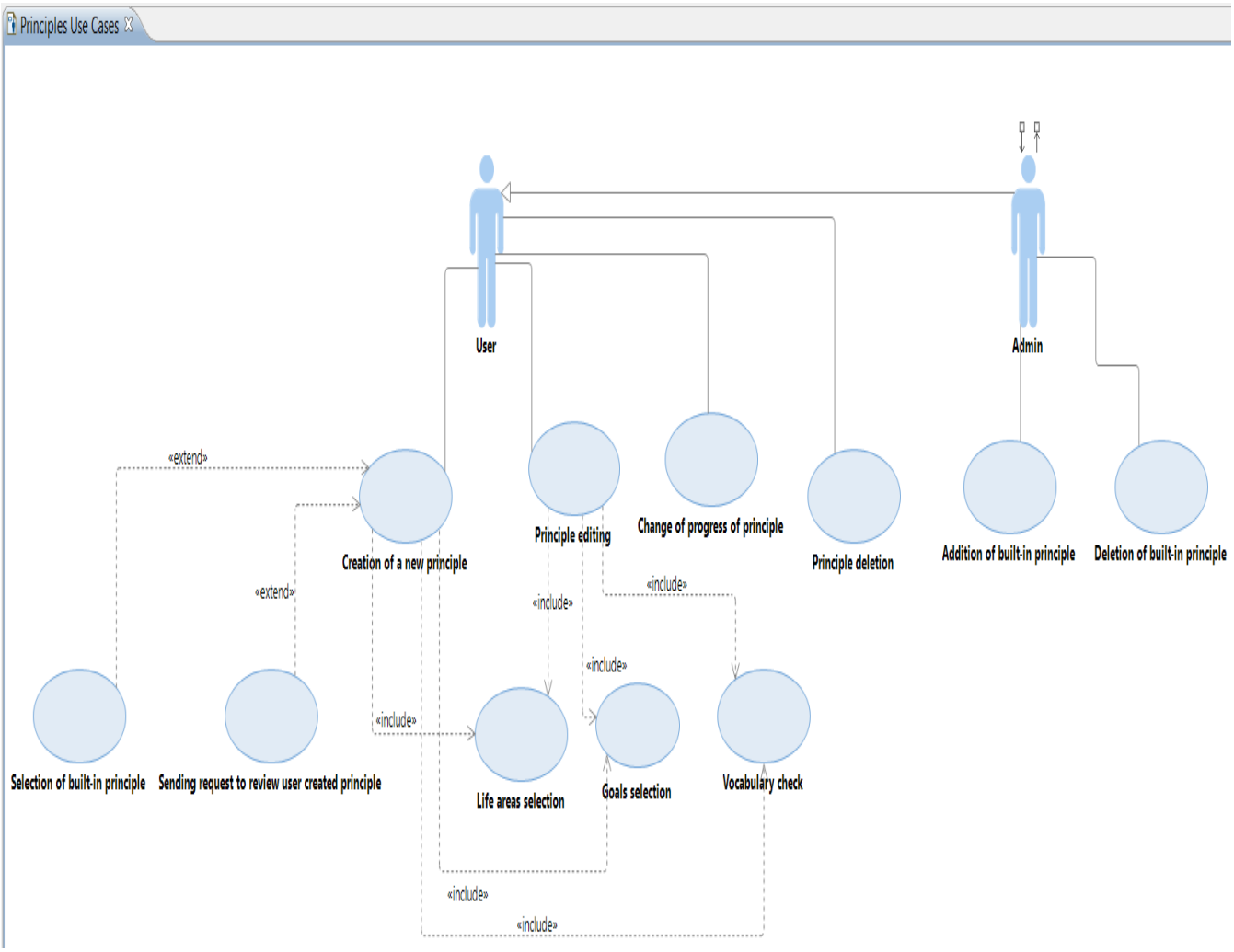

Рис. 2.5.5. Діаграма ВВ «Principles»

Як можна тут побачити, щоб додати принцип користувач або може вибрати зі списку запропонованих (ВВ «Selection of built-in principle»), або ж створити свій власний, при останньому він може також надіслати запит для огляду розробником створеного принципу. Інженер впродовж певного часу надішле відповідь на пошту. При доданні також відбувається перевірка на лексику, в тому числі цензурну (ВВ «Vocabulary check»), вибір сфер життя та цілей, до яких належить принцип.

Варто виділити прецедент «Change of progress of principle». Він відповідає за збільшення чи зменшення того, на скільки користувач набув принцип у відсотковому співвідношенні й базується на прогресі звичок, які відносяться до цього принципу.

5. «Helper» (помічник)

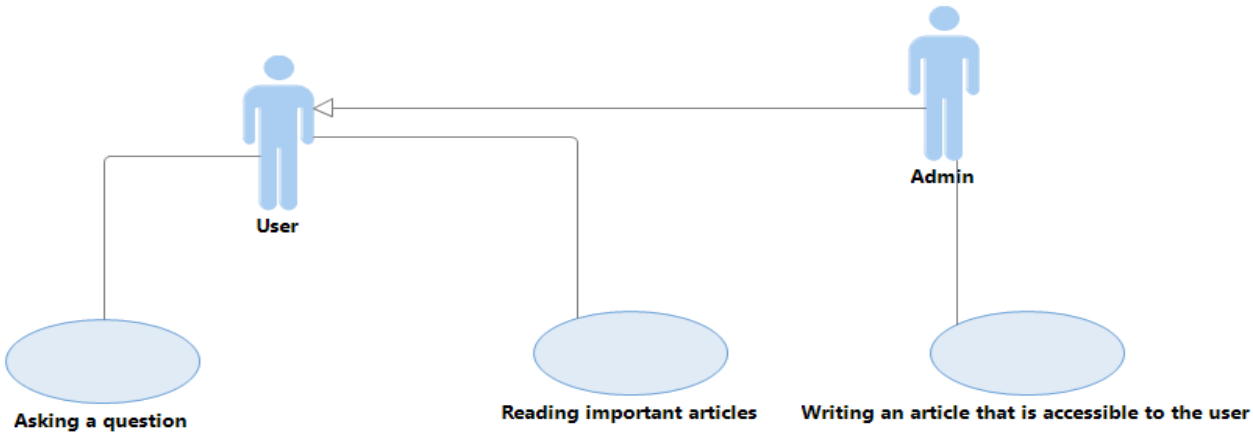

Рис. 2.5.6. Діаграма ВВ «Helper»

Отже, було успішно побудовано 5 пакетів з діаграмами ВВ, що було дуже важливо для подальшого розвитку проєкту. Вони допомогли не виконувати частих змін в програмі й полегшили розуміння вимог.

2.6. Вибір основних технологій та мов

1) Вибір мови програмування, основної платформи та середовища розробки

Оскільки основним напрямком мого професійного розвитку є мова C# [11] й платформа .NET, то було вибрано саме їх (версій 11.0 та 7.0 відповідно, тобто найновіший на поточний день). Вони на даний час активно розвиваються й представляють усі необхідні можливості для створення кросплатформних додатків з клієнт-серверною архітектурою.

.NET - це безкоштовна, кросплатформна, відкрита платформа для розробників для створення різних типів додатків [12]. За допомогою .NET можна використовувати різні мови, редактори та бібліотеки для створення додатків для Інтернету, мобільних, настільних комп'ютерів, ігор, Інтернет-речей тощо. Розробляється компанією Microsoft, а, отже, потенціал в неї надзвичайно великий, оскільки це одна з найуспішніших компаній по розробці ПЗ [17].

Найпопулярнішим й, на мою думку, найзручнішим середовищем розробки для них є Visual Studio Community 2022 [13]. Через вище вказані описи було вирішено використовувати саме їх.

2) Вибір мови розмітки та платформи для розробки клієнтського застосунку

Оскільки клієнтський застосунок повинен фокусуватися на кількох ОС, то загалом вибір міг лежати лише між двома платформами:

- Xamarin - це платформа з відкритим вихідним кодом для створення сучасних додатків для iOS, Android та Windows. В листопаді 2021 року компанія Microsoft припинала її оновлення. Загалом я та інші розробники схиляємся до думки, що ця платформа є зовсім не зручної до розробки, оскільки як здавалося звичайна функціональність працює досить дивним чином або ж зовсім виходить з ладу.
- .NET MAUI - це кросплатформна платформа для створення нативних мобільних і настільних додатків (застосунок, який для розроблений для конкретної ОС з врахуванням її специфіки та доступом до всіх її ресурсів [16]) за допомогою C# і мови розмітки XAML [14] (про XAML див. детальніше тут [15]). Використовуючи .NET MAUI, можна розробляти додатки, які можуть працювати на ОС Android (більшість мобільних телефонів та планшетних ПК), iOS (iPhone, iPod Touch), macOS та Windows з єдиної спільної кодової бази. Нативність забезпечується за рахунок так званих «обробників» [19]. Ця платформа прийшла на заміну Xamarin з огляду на її недоліки й вважається дуже молодою. Про її ряд переваг над Xamarin можна ознайомитись тут [20].

Враховуючи усі вище вказані причини, було вибрано саме платформу .NET MAUI і мову розмітки XAML (тільки її підтримує даний фреймворк) для розробки клієнтського застосунку. Також через те, що актуальність десктопних додатків з часом все падає й падає, вирішено зосередитись тільки на ОС-ах Android та iOS. В майбутньому ще планується розробка веб-додатку. Другою причиною такого вибору слугувало те, що лише їх підтримує безкоштовна бібліотека DevExpress.Maui.

3) Вибір платформи для розробки серверного застосунку

Тепер перейдемо до вибору платформ для розробки серверу. Загалом вибір був між 3-ма фреймворками, які і представляються компанією Microsoft для відповідних цілей. Це .NET Framework, .NET Standard та .NET Core.

- .NET Standard - набір фундаментальних інтерфейсів API (зазвичай їх називають бібліотеками базових класів або BCL), які повинні реалізовувати всі реалізації .NET (.NET Framework, .NET Core, .NET MAUI). Орієнтуючись на .NET Standard, існує можливість створювати бібліотеки, які можна використовувати у всіх .NET-додатках, незалежно від того, під якою реалізацією .NET чи ОС вони працюють. Тобто це зазвичай має сенс лише в тому випадку, коли проєкт розробляється для NuGet-пакет(механізм для поширення свого коду та можливості простого його установлення користувачами[25]).
- .NET Core - це найновіша реалізація .NET [22]. Вона має відкритий вихідний код і доступна для багатьох Windows, Linux та macOS. За допомогою .NET Core можна створювати кросплатформні консольні додатки, API, веб-додатки ASP.NET Core (безкоштовна, відкрита та крос-платформна платформа для створення хмарних застосунків, таких як веб-додатки, програми Інтернет-речей та мобільні бекенди [22]), хмарні сервіси та багато інших.
- .NET Framework - це середовище розробки програмного забезпечення для створення та запуску додатків у Windows. На даний час його використовують значно рідко, ніж раніше, оскільки .NET Core повністю його здатен замінити. А останній на відміну від першого є крос-платформним та продовжує активно оновлюватись. Крім того, .NET Core підтримує усі найновіші версії мови C# та .NET, в той час як .NET Framework зупинився на .NET 4.8 та C# 7.3 [24] (нагадую, що станом на сьогоднішній день існують версії .NET 7.0 та C# 11.0).

Отже, з огляду на усі вище вказані недоліки .NET Framework, переваги .NET Core та те, що .NET Standard має менше можливостей, ніж .NET Core, було прийнято рішення, використовувати саме останню вказану платформу.

4) Вибір утиліти для роботи зі сховищем даних.

В порівняння беруться дві, оскільки в основному мені доводилось лише з ними працювати:

- SQL Server Management Studio (SSMS) - інтегроване середовище для управління будь-якою SQL-інфраструктурою [26]. Вона надає можливість доступу, налаштування, керування, адміністрування та розробки всіх компонентів SQL Server, SQL Server у віртуальній машині Azure, керованого екземпляра Azure SQL, БД Azure SQL та Azure Synapse Analytics. Утиліта включає скриптовий редактор і графічну програму, яка працює з об'єктами й настройками сервера.

Головним інструментом SQL Server Management Studio є Object Explorer, який дає змогу користувачеві переглядати, отримувати об'єкти сервера, а також повністю ними управляти. Тут мовою, що використовується для запитів, є Transact-SQL.

- MySQL Workbench - це візуальний інструмент для архітекторів баз даних, розробників і адміністраторів баз даних [27]. MySQL Workbench надає послуги з моделювання даних, розробці SQL і комплексні засоби адміністрування для налаштування сервера, адміністрування користувачів, резервного копіювання та багато чого іншого. MySQL Workbench доступний під Windows, Linux і Mac OS X [27].

Як MySQL, так і MS SQL Server широко використовуються системою баз даних підприємств. MySQL - це СУБД з відкритим вихідним кодом, в той час як SQL Server - це продукт Microsoft. Microsoft дає змогу підприємствам вибирати з декількох версій SQL Server відповідно до своїх потреб та бюджету.

Основною відмінністю між ними є те, що MySQL не дає змогу користувачам вбивати або скасовувати запит під час його виконання. Користувачі повинні вбити весь процес, щоб зупинити виконання SQL-запиту. Однак програмісти SQL-сервера можуть усікти запит до бази даних під час

виконання, не вбиваючи весь процес. Крім того, він використовує транзакційний рушій для підтримки постійного стану. Ця функція робить SQL-сервер більш успішним в порівнянні з MySQL. На противагу цього MySQL підтримує декілька механізмів зберігання, на відміну від MS SQL Server, що робить першого більш гнучким. Також перевагою SQL Server є те, що він, на відміну від MySQL, не блокує базу даних при резервному копіюванні даних. Ця особливість дає змогу користувачам створювати резервні копії та відновлювати величезну кількість даних без додаткових витрат часу і зусиль.

Під самі проєкти на мові C# найпродуктивнішою системою керуванням баз даних є Microsoft SQL Server Management Studio, оскільки обидва розробляються компанією Microsoft, яка забезпечує оптимізовану взаємодію між ними.

З огляду на переваги MS SQL Server, а також те, що немає необхідності в кількох механізмів зберігання, мій вибір склонився саме до нього.

5) Вибір технології взаємодії між базою даних та застосунком.

Найпопулярнішими структурами для взаємодії з базою даних у C# є EF і EF Core. Оскільки ми використовуємо .NET Core, то простий EF звісно ми вибрати не можемо через несумісність платформ.

EF Core - це легка, розширювана, відкрита та крос-платформна версія популярної технології доступу до даних EF [30]. EF Core зазвичай служить в якості об'єктно-реляційного маппера (ORM (object-relational mapper)), як і в нашому випадку, який:

- Дозволяє .NET розробникам працювати з БД, використовуючи об'єкти .NET.
- Усуває необхідність у написанні більшої частини коду доступу до даних, який зазвичай потрібно писати.

Відмінною рисою Entity Framework є використання запитів LINQ для вибірки даних з БД. За допомогою LINQ ми можемо не тільки отримувати певні рядки, що зберігають об'єкти, з бд, а й отримувати об'єкти, пов'язані різними асоціативними зв'язками.

EF Core підтримує багато механізмів БД, в тому числі й Microsoft.EntityFrameworkCore.SqlServer, який і застосовується в поточному

проєкті. Як раз його і потрібно устанвоити, щоб мати змогу користуватись EF Core. Для цього ПКМ на назву проєкту у вікні «Оглядач рішень»->Керування пакетами NuGet. Вводимо у рядку пошуку EntityFramework і встановлюємо наступне.

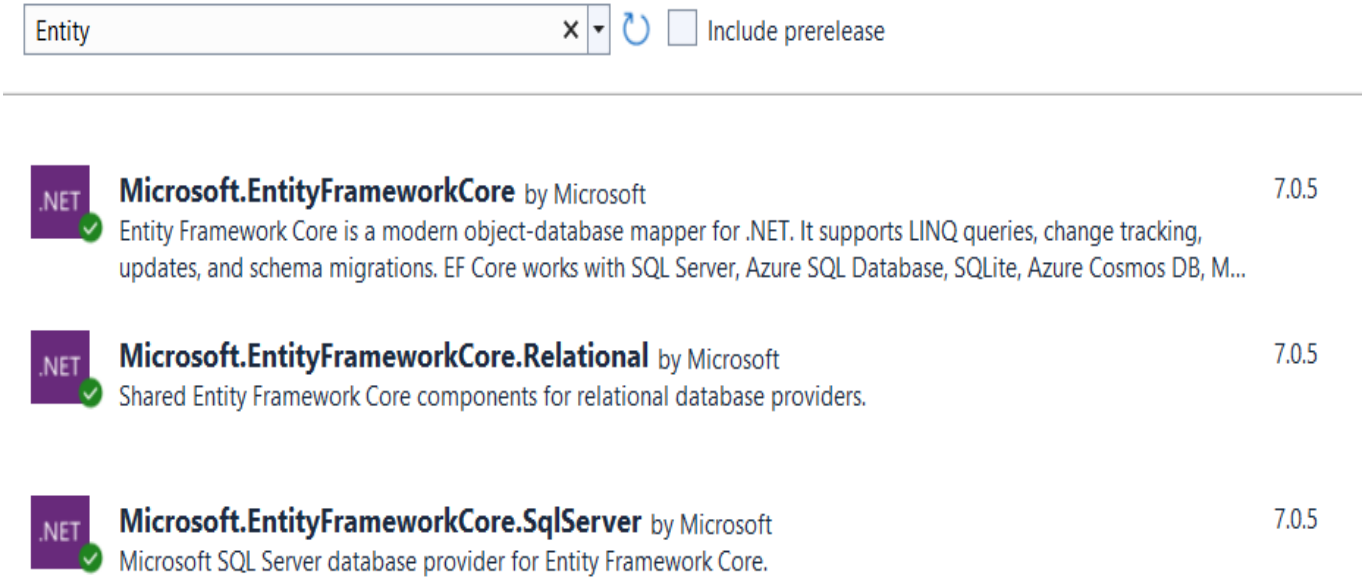

Рис. 2.6.1. Установлення EntityFramework.

- Microsoft.EntityFrameworkCore – базовий EF Core для всіх провайдерів.
- Microsoft.EntityFrameworkCore.Relational – містить EF Core-компоненти для реляційний провайдерів БД, до яких, звісно, відноситься наступний пакет
- Microsoft.EntityFrameworkCore.SqlServer - постачальник баз даних Microsoft SQL Server для EF Core.

6) Платформа для розробки проміжного шару між frontend та backend

Загалом в нас як такого вибору немає тут через використання .NET Core, тому було використано платформу ASP.NET Core Web API. Для початку визначимо, що таке API. Це проміжний програмний агент, який дозволяє двом або більше програмам взаємодіяти один з одним [31].

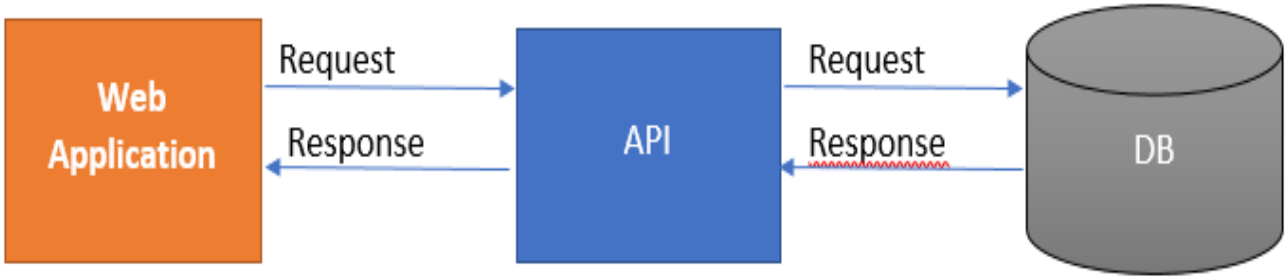

Рис. 2.6.2. Архітектура Web API.

Веб-API - це інтерфейс прикладного програмування для веб-додатків або веб-серверів [31]. Він використовує протокол HTTP для зв'язку між клієнтами та веб-сайтами для доступу до даних чи виконання певних дій.

ASP.NET Core – це платформа з відкритим програмним кодом, що розроблена компанією Microsoft і може запускатись на macOS, Linux, Windows і Docker (надає змогу швидко розгортати проєкти у production-версію [32]). Вона створена на основі .NET Core.

Отож, тепер ми можемо сформулювати, що таке ASP.NET Core Web API. Це мультиплатформний Web API, розроблений на платформі ASP.NET.

Різні пристрої запитують Web API, який, у свою чергу, відповідає у форматі JSON (формат для простого зберігання та транспортування даних [33]). Більшість пристроїв здатні розуміти вихідні дані у цьому форматі.

2.7. Висновки до розділу

Отже, було успішно ознайомлено з інформацією зі сфери психології, саморозвитку та штучного інтелекту, що надало також змогу визначити вимоги до програмного забезпечення. Як бачимо, принципи – не схожі на звички, як вважалось на початках розробки застосунку. Останні можуть лише відображати виконання особистих принципів, але це не завжди так. Наприклад, звичка щоденного фізичного вправлення може бути наслідком особистого принципу здорового способу життя. Однак не всі звички безпосередньо пов'язані з особистими принципами.

Якщо узагальнити, звичка - це конкретна діяльність або реакція, яка виникає через її повторення, тоді як особистий принцип - це глибоко укорінене переконання чи цінність, яка визначає загальну поведінку та вибори у житті.

Стосовно технологій, платформ та мов програмування, то їх було успішно вибрано, оскільки не довелось їх змінювати в процесі розробки програмного забезпечення. Скоріше навпаки, вони значно пришвидшували процес, через те що деяка базова функціональність одразу містилась в них.

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Математична модель обчислення прогресу звички

Світ активно вивчає мозок людини, що призводить до розуміння процесу виконання його функцій. Це допомагає визначити, як швидко в людини формуються автоматичні дії й думки. Зараз це є актуальним питанням, оскільки допомагає автоматизувати процес досягнення цілей людини за рахунок обмеження кількості звичок для набуття чи позбуття в конкретний проміжок часу. Це забезпечує уникнення психологічного й фізичного перегорання.

Метою підрозділу є побудувати математичну модель, яка визначає прогрес звички, тобто те, на скільки вона є автоматичною. Така поведінка демонструє усі або деякі з наступних ознак: неусвідомленість, неконтрольованість, ненавмисність та ефективність [51]. Це ще раз підкреслює важливість поточної математичної моделі, яка, в свою чергу, має відповідати наступним вимогам.

1) Кожне повторення звички чи його відсутність має впливати на прогрес, навіть якщо воно виконувалось в минулому;

2) прогрес звички розвивається швидко, якщо вона слабко освоєна. Після її первинного розвитку, прогрес поступово уповільнюється при систематичному повторенні звички;

3) чим частіше звичка має виконуватись, тим швидше вона набувається, але й тим швидше вона втрачається, якщо її не дотримуватись;

4) оскільки складність кожної дії є різною й кожна особа має свої особливості, то мінімальна кількість повторень звички для її повної автоматизації має варіюватися.

Ці вимоги побудовані на основі статті [52]. Для вибору первинної моделі було досліджено додаток “Трекер звичок Loop” [53]. Це тривало протягом 1 року й 10 місяців і в результаті було чітко визначено, що його функція обчислення прогресу звички відповідає усім вище вказаним вимогам, окрім останньої.

Оскільки ця програма є у відкритому доступі [53], було описано її математичну модель і досліджено параметри та їх значення. Модель побудована на основі методу експоненціального згладжування й виглядає наступним чином.

$$P_t = (P_{t-1} * 0{,}5^{\frac{\sqrt{f}}{13}}) + (v * (1 - 0{,}5^{\frac{\sqrt{f}}{13}})), \quad (1)$$

де P – прогрес звички. Можливі значення – від 0 до 1, де 0 – немає жодного розвитку, 1 – звичка повністю автоматична.

t – кількісний номер повторення звички. Наприклад, якщо особа 10 разів мала виконати звичку, то $t = 10$.

0,5 - стандартний експоненціальний коефіцієнт згладжування.

f – частота звички. Наприклад, якщо звичка має повторюватися 3 рази протягом 8 днів, то $f = 3 / 8 = 0{,}375$;

13 - коефіцієнт масштабування, який корегує швидкість зростання чи спадання прогресу звички на основі квадратного кореня з частоти звички.

v – бал, на скільки добре набувач звички впорався з її повторенням під час поточного оцінювання. Якщо впорався, то $v = 1$, якщо – ні, то $v = 0$.

Тепер необхідно додати в цю модель коефіцієнт складності звички. Чим остання важча для конкретної особи, тим більше повторень слід виконати для її набуття [52]. Так, було проведено дослідження діапазону кількості днів, коли дія стає автоматичною [52]. Він складає 18-254 діб, якщо вона повторюється кожного дня [52]. Причому в середньому це відбувається через 66 днів [52].

Параметр складності повинен впливати саме на степінь числа 0,5 для корегування швидкості зростання чи спадання прогресу. Отож, далі було здійснено дослідження, для визначення відповідностей параметрів складності.

Таблиця 1

Відповідність різних параметрів складності

Складність	Мінімальна кількість повторень звички для її автоматизації	Коефіцієнт складності
1	18	5,6
2	30	3,33
3	44	2,3
4	59	1,7

5	66	1,525
6	71	1,4
7	100	1,0
8	130	0,77
9	190	0,524
10	254	0,392

Мінімальна кількість днів для автоматизації звички і її відповідна складність вибирались згідно з дослідженням про формування звичок [52]. Тобто чим звичка є складніша для конкретної особи, тим більше її повторень слід зробити. Тепер перейдемо до програмного методу.

Лістинг 3.1.1. Метод обчислення прогресу звички мовою C#.

```
88      public double ComputeScore( double frequency, double previousScore, double checkmarkValue, int complexity )
89      {
90          double coefOfComplexity = complexity switch
91          {
92              1 => 5.6,
93              2 => 3.33,
94              3 => 2.3,
95              4 => 1.7,
96              5 => 1.525,
97              6 => 1.4,
98              7 => 1.0,
99              8 => 0.77,
100             9 => 0.524,
101             10 => 0.392,
102             _ => throw new ArgumentException( message: $"Complexity {complexity} of habit is not supported",
103             paramName: nameof( complexity ) )
104         };
105
106         double exponentialSmoothingFactor = 0.5;
107         double scalingFactor = 13.0;
108         double decayFactor = Math.Pow( exponentialSmoothingFactor, Math.Sqrt( frequency ) * coefOfComplexity / scalingFactor );
109
110         double score = previousScore * decayFactor;
111         score += checkmarkValue * (1 - decayFactor);
112         return score;
113     }
```

Рис. 3.1.1. Функція ComputeScore

Спершу визначається коефіцієнт складності відповідно до complexity. Далі обчислюється фактор розпаду (decayFactor), який обраховується також відповідно до нового коефіцієнту. Після цього обчислюється поточне значення прогресу (score) відповідно до попереднього розвитку звички (previousScore), фактору розпаду та балу v (checkmarkValue).

Математична модель тепер виглядає наступним чином.

$$P_t = (P_{t-1} * 0{,}5^{\frac{\sqrt{f} * c}{13}}) + (v * (1 - 0{,}5^{\frac{\sqrt{f} * c}{13}})), \quad (2)$$

де с – коефіцієнт складності набуття звички для конкретної особи.

Як бачимо, тут було додано коефіцієнт складності до попередньої математичної моделі й тепер є можливість визначати прогрес звички відповідно до її важкості.

3.2. Генераторний попередньо навчений трансформатор

Він використовується для генерації відповідей ШІ-помічника та рекомендованих звичок на основі інших звичок користувача, вибраних сфер життя, його цілі на все життя, статі та основного гасла.

Generative Pre-trained Transformer (GPT) - це модель глибокого навчання, яка базується на трансформаторі, представленому в роботі "Attention is All You Need" Васілем Лінгом та іншими [54]. Модель використовує механізм великої кількості параметрів для генерації тексту та вирішення різних завдань, після того як вона була попередньо навчена на великих обсягах текстових даних.

У контексті обробки природної мови, термін "токен" вказує на найменшу одиницю, яку модель може розглядати. У випадку моделі трансформатора, такої як GPT, токени складають 0.75 слова й використовуються для представлення вхідної інформації у вигляді векторів, що подаються моделі для обробки. Такий підхід дозволяє моделі розуміти та генерувати текст на рівні токенів, що є більш кращим, ніж працювати з окремими символами.

Математично, трансформатор може бути описаний як мережа, що використовує механізм самоуваги для обробки послідовностей. Нехай маємо вхідний текст у вигляді послідовності токенів x_1, x_2, ..., x_n, де кожен x_i - це вектор, який представляє відповідний токен.

Модель трансформатора включає кілька шарів, кожен з яких має два підшари: механізм самоуваги та feedforward нейронну мережу. Позначимо ваги цих шарів як W.

Операція самоуваги визначається наступним чином.

$$SelfAttention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V, \quad (3)$$

де Q, K, V - проекції вхідних векторів x на простори запитань, ключів і значень відповідно;

d_k - розмірність простору ключів.

Feedforward шар використовується для нелінійного перетворення.

$$FeedForward(x) = \sigma(W2ReLU(W1x + b1) + b2), \quad (4)$$

де $W1$, $W2$, $b1$, $b2$ - параметри моделі.

Шари трансформатора оброблюють вхідну послідовність, дозволяючи моделі "дивитися" на різні частини введеного тексту. Результати обробки можуть бути використані для генерації тексту або вирішення інших завдань, для яких модель може бути попередньо навчена.

GPT базується на моделі трансформер, яка складається з багатьох шифрувальників і дешифрувальників у послідовних шарах. Основна концепція моделі трансформер полягає в тому, що кожне слово в текстовому вводі обробляється паралельно, на відміну від традиційних послідовних моделей, таких як LSTM або GRU. Крім того, трансформер використовує механізм уваги, який дозволяє моделі зосереджувати увагу на важливих словах при генерації виходу.

З точки зору математики, проведення обчислень в моделі GPT можна описати наступним чином.

1) Ініціалізація: кожне вхідне слово перетворюється на вектор за допомогою вбудовування, а позиції слів кодуються за допомогою позиційних кодувань. Вбудовування та позиційні кодування потім додаються одне до одного.
2) Перетворення: вхід перебуває крізь серію трансформаційних блоків, кожен з яких містить шари уваги та повнозв'язані шари.
3) Увага: слова, у яких треба зосередитись, обчислюються путем додавання ваг до відповідних векторів вбудовування.

4) Нормалізація: вхідний вектор нормалізується для стабілізації обчислень.
5) Сховище даних: вектори вбудовування зберігаються в сховище даних для подальшого використання під час декодування.
6) Декодування: за допомогою збережених векторів вбудовування генерується вихідна послідовність.

Цикл обчислення продовжується, поки не буде сгенеровано задану кількість вихідних слів.

3.3. Висновки до розділу

Успішно визначено модель GPT та досліджено існуючу функцію обчислення прогресу звички, побудовано відповідну до неї математичну модель. Визначено таблицю відповідностей складностей звичок; мінімальної кількості повторень для автоматизації звичок; коефіцієнтів, що регулюють швидкість зростання чи спадання прогресу. Внаслідок цього, реалізовано програмну функцію обчислення прогресу звички з врахуванням її важкості та змінено відповідну математичну модель.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Насамперед варто зазначити, що для написання чистого та оптимального конкурентного коду використовувались принципи з книги [45] та конвенції, які описані в документі [46]. Частину з них було автоматизовано й таким чином перенесено в файл .editorconfig.

4.1. Розробка основи серверного застосунку

4.1.1. Архітектура

Його архітектура в перегладачі рішення виглядає наступним чином.

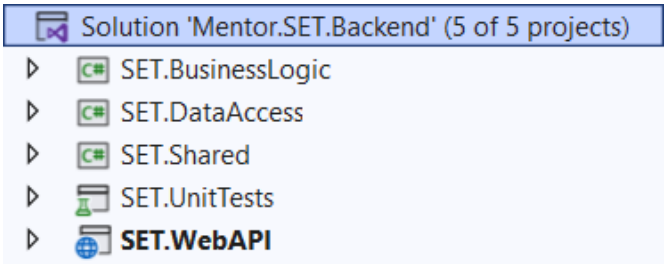

Рис. 4.1.1.1. Архітектура серверного застосунку в «Оглядачі рішень»

Кожен з підпунктів елементу «Solution ‘Mentor.SET.Backend’» виступає проєктом (збіркою) .NET Core або того, що створений на його основі. Оглянемо усіх їх.

- SET.BusinessLogic – .NET Core-проєкт, що містить реалізацію бізнес-логіки, тобто спеціальних правил і алгоритмів, які керують обміном інформацією між БД та інтерфейсом користувача [28]. Бізнес-логіка - це, по суті, частина комп'ютерної програми, яка містить інформацію (у вигляді бізнес-правил), що визначає або обмежує те, як працює бізнес. Наприклад, тут реалізовані сервіси регістрації, логінування, logout, звичок користувача і т.д.
- SET.DataAccess – .NET Core-збірка, яка створює, налаштовує та видозмінює схему БД; надає можливість керування її даними.
- SET.Shared - .NET Core-проєкт, що містить:
  - сервіси, розширення та утиліти, які використовуються іншими backend-збірками;
  - описи .NET-об’єктів, в які та з яких формуються записи таблиць БД.

- SET.UnitTests – .NET Core-збірка, що складається з модульних тестів backend-у. Вона використовується xUnit, тобто найпопулярніший і найзручніший інструментарій для виконання даного виду технічного випробування [36].
- SET.WebAPI – ASP.NET Core Web API-проєкт, який виступає проміжним шаром між frontend та backend. Ця збірка має унікальний процес створення, тому оглянемо більш детально.

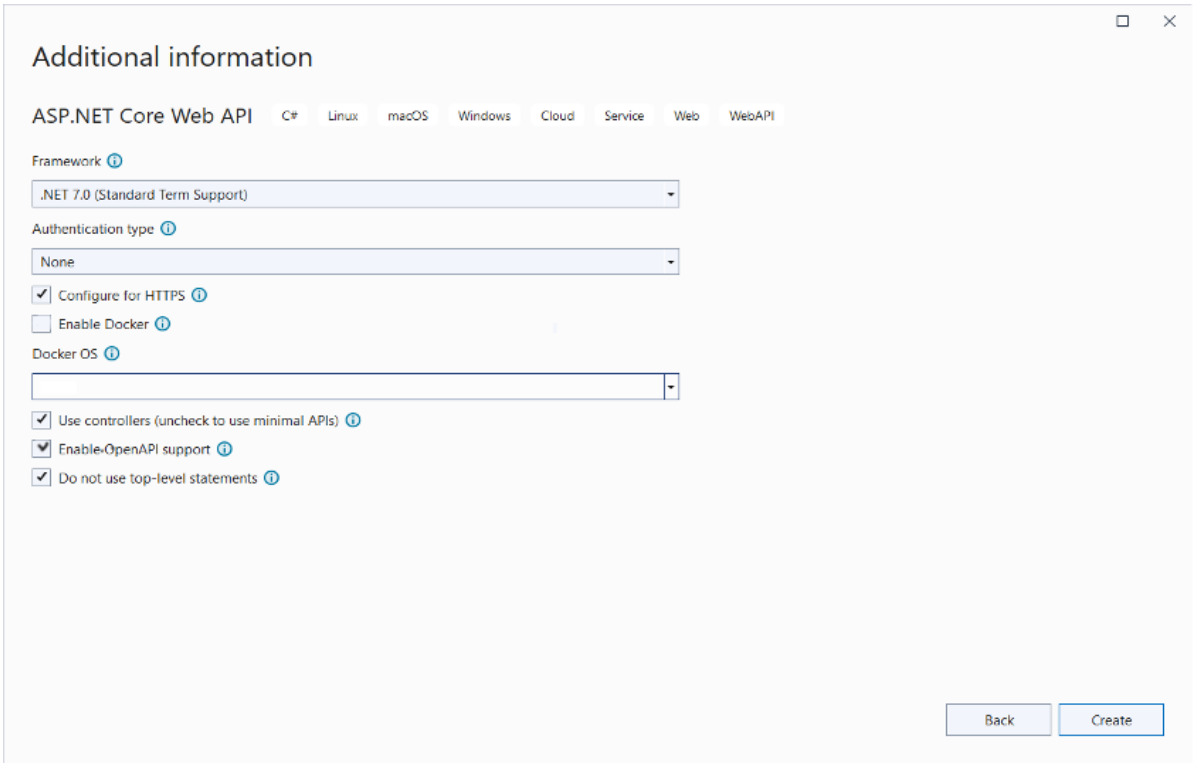

Рис. 4.1.1.2. Створення проєкту SET.WebAPI

Як бачимо, було вибрано не використовувати автентифікацію за замовчуванням, оскільки була розроблена власна з різноманітними токенами (див. підрозділ «Авторизація та автентифікація»). Docker не є потрібним для цього проєкту, оскільки для його публікації в Azure, перший сервіс не знадобиться. OpenAPI або Swagger використовується для спрощення тестування цього рішення шляхом виклику окремих URL.

Отже, для backend було використано стандартну архітектуру, без використання усіляких патернів проєктування. Це зроблено навмисно, оскільки тут потреби в них немає, а вони здебільшого лише ускладнюють розуміння програми.

Після опису архітектури тепер можна перейти до огляду реалізації проєкту.

4.1.2. Створення БД

Для взаємодії з нею використовується Entity Framework Core й модель Code First. Для цього слугували 2 причини:

1) ми маємо можливість повертатись до будь-якої версії створеної БД. Це досить корисно, особливо якщо ми підтримуємо різні версії додатку. Без Code First ми повинні використовувати окремий інструмент для MS SQL Server, який є платний.

2) продуктивність. Створювати щось за допомогою клавіатури є швидше, аніж за допомогою миші.

Для того, щоб створити моделі на основі описаних сутностей (див. підрозділ «Схема сутностей»), можна скористатись розширенням StarUML - C#. Для цього спершу його потрібно встановити, тому виберемо в пункті меню «Tools» елемент «Extension Manager», введемо в пошуку «C#» й натиснемо Install поряд з потрібним пунктом. Отримаємо наступне.

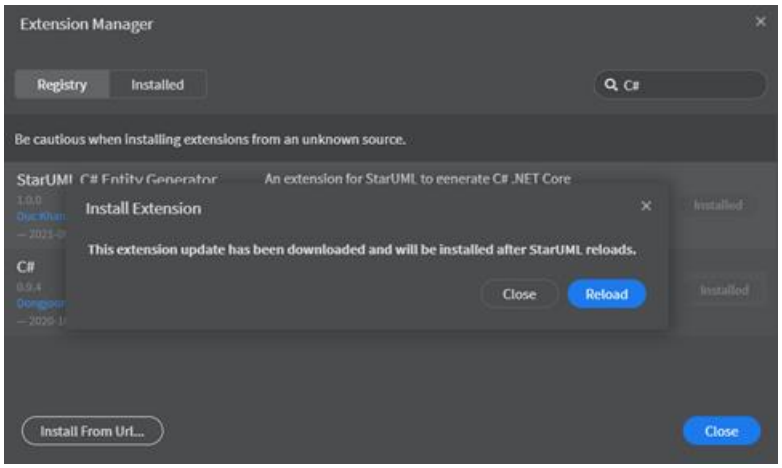

Рис. 4.1.2.1. Результат скачування StarUML-розширення «C#»

Перезапустимо StarUML, оскільки чомусь Reload не працює. Тепер пункти елемента меню Tools виклядають наступним чином.

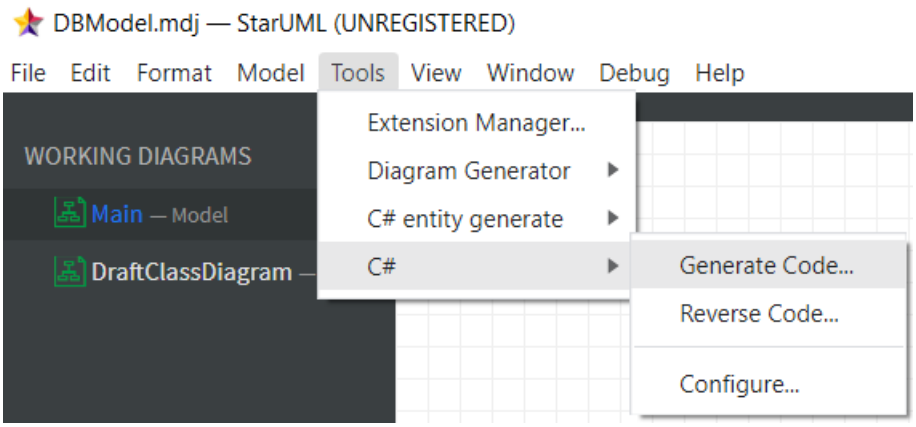

Рис. 4.1.2.2. Оновлений пункт Tools в StarUML

Натиснемо Configure й налаштуємо генерацію C# коду. Вони повинні виглядати наступним чином.

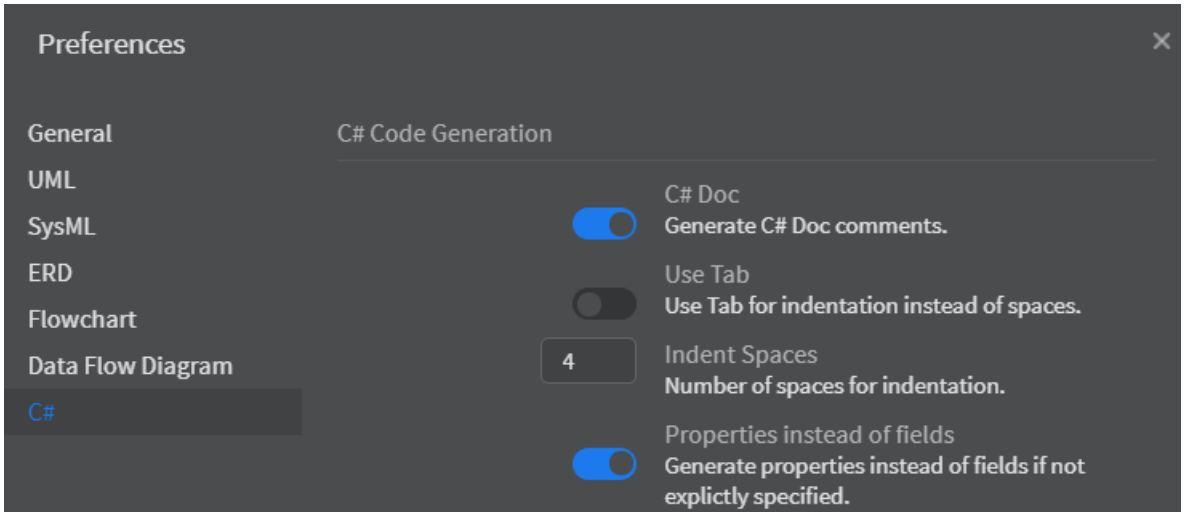

Рис. 4.1.2.3. Налаштування генерації C#-коду

Тепер знову натиснемо Tools → C# й виберемо Generate Code. Після цього вкажемо шлях, куди згенерувати код. В результаті отримаємо наступне.

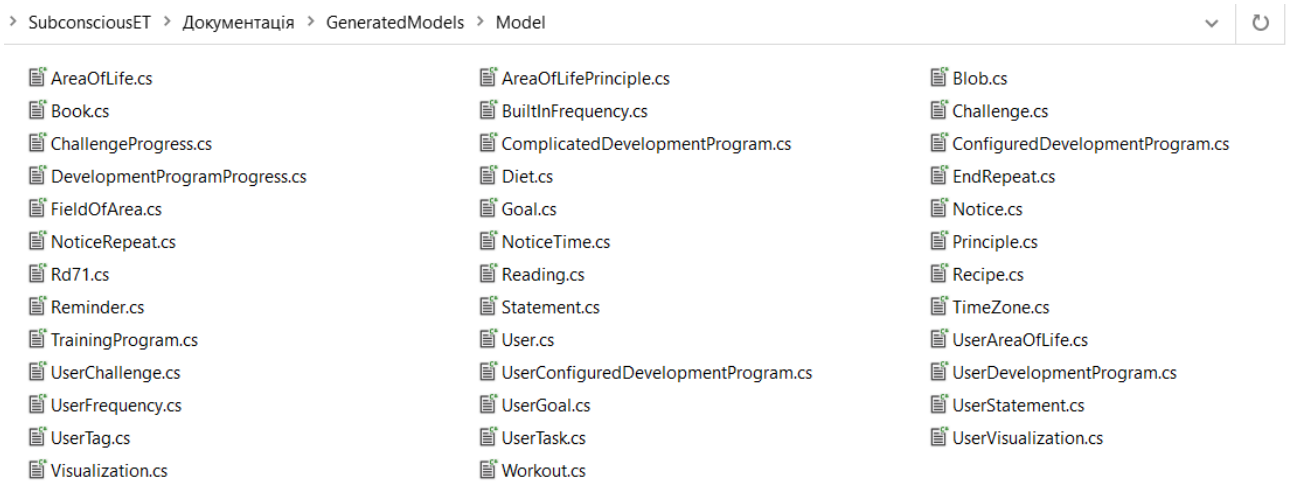

Рис. 4.1.2.4. Згенеровані сутності

Далі їх було перенесено в збірку SET.DataAccess, в папку Models, відформатовано та описано їх взаємодію. Після цього слід було створити міграції. Це виконується за допомогою команди add-migration в Package Manager Console. Щоб їх запустити слід скористатись update-database:

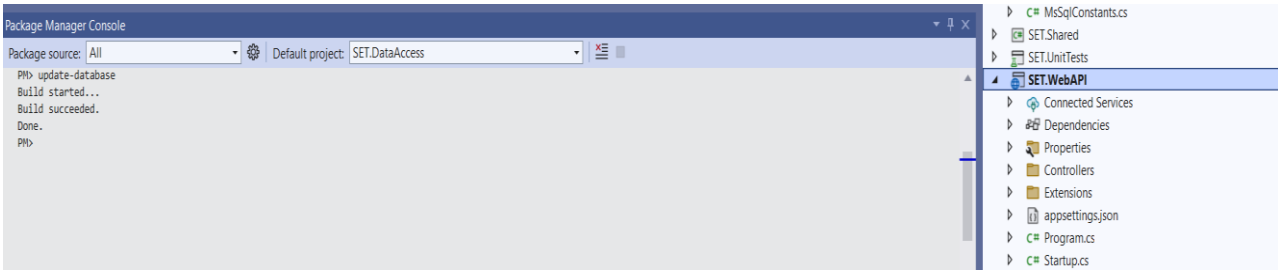

Рис. 4.1.2.5. Запуск міграцій

Результат створення міграцій можна переглянути на наступному рисунку.

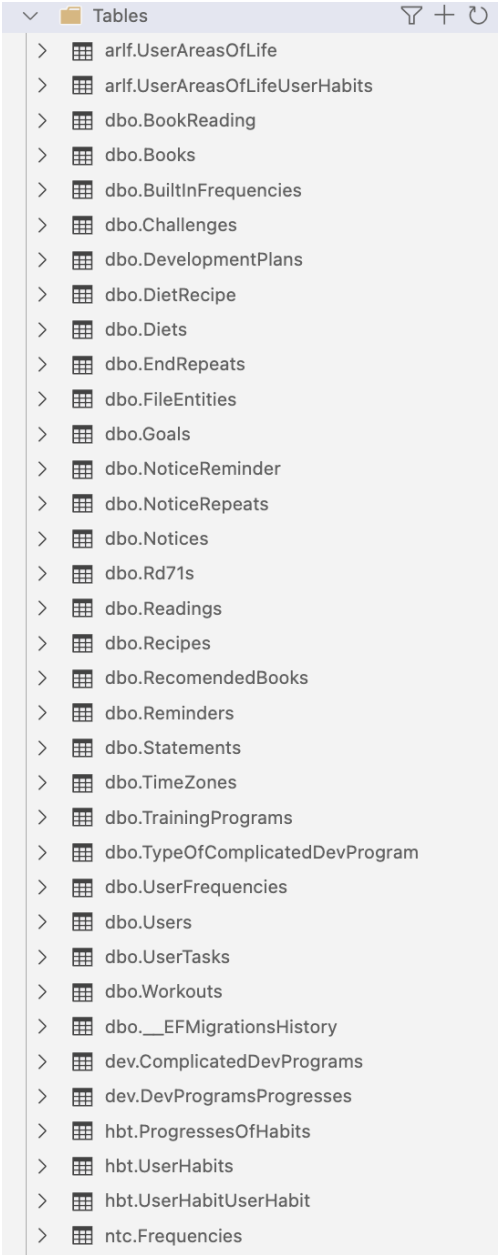

Рис. 4.1.2.6. Створені таблиць БД

4.1.3. Налаштування серверу

Практично будь-який Web-проєкт на платформі ASP .NET Core запускається з виконання методу Program.Main. Клас Program виглядає наступним чином.

```
7   public static class Program
8   {
        0 references
9       public static void Main(string[] args)
10      {
11          CreateHostBuilder(args).Build().Run();
12      }
13
        1 reference
14      public static IHostBuilder CreateHostBuilder(string[] args)
15      {
16          return Host.CreateDefaultBuilder(args)
17                     .ConfigureWebHostDefaults(webBuilder => webBuilder.UseStartup<Startup>());
18      }
19  }
```

Рис. 4.1.3.1. Клас-вхід в серверний застосунок

Тобто тут спершу виконується створення HostBuilder з конфігурації за замовчуванням (рядок 16). Це надало змогу значно швидше виконувати розробку програми за рахунок зручного логінування та простого отримання конфігурацій додатку.

Далі метод IHostBuilder.ConfigureWebHostDefaults конфігурує IHostBuilder зі значеннями за замовчуванням для розміщення веб-додатку. Це забезпечує наступні важливі пункти.

1) Використання Kestrel (підтримує HTTPS, аутентифікацію, засоби безпеки та інші функції, необхідні для розгортання безпечних та надійних веб-додатків [38]) як веб-серверу і налаштування його за допомогою провайдерів конфігурації програми;
2) інтеграція з IIS (гнучкий веб-сервер загального призначення від Microsoft [39]).

Цю функцію слід викликати перед налаштуванням конкретного додатка, щоб уникнути перезапису наданих послуг, джерел конфігурації, оточення, кореневого каталогу вмісту тощо.

В метод ConfigureWebHostDefaults ми передаємо функцію, яка встановлює клас, що використовуватиметься для налаштування та конфігурації. В даному випадку це Startup. Він містить методи, які дозволяють вам настроїти різні

аспекти додатка, такі як маршрутизація, сервіси, аутентифікація, авторизація та інші.

Методи, які реалізовані в класі Startup:

1) ConfigureServices - викликається при старті додатка та використовується для налаштування сервісів. Тут реєструються залежності, задаються особливості застосованих бібліотек (AutoMapper, JWT-авторизації, Swagger, контекст БД, Newtonsoft JSON).

```
28      public void ConfigureServices(IServiceCollection services)
29      {
30          services.
31              AddControllers().
32              AddNewtonsoftJson( options =>
33                  options.SerializerSettings.ReferenceLoopHandling = ReferenceLoopHandling.Ignore );
34
35          services.AddJwtAuthentication(() => Configuration[ key: "JwtSettings:Secret" ]);
36          services.AddSwaggerWithBearer();
37          services.AddAutoMapper();
38          services.AddDbContext<AppDbContext>(options =>
39          {
40              string? connectionString;
41  #if DEBUG
42              connectionString = Configuration.GetConnectionString( name: "DefaultConnection" );
43  #else
44              connectionString = Configuration.GetConnectionString( "Azure" );
45  #endif
46              options.UseSqlServer( connectionString );
47          });
48          services.AddScoped<IAuthService, AuthService>();
49          services.AddScoped<IGoalService, GoalService>();
50          services.AddScoped<IFileSystemService, FileSystemService>();
51          services.AddSingleton<IRandomService, RandomService>();
52          services.AddSingleton<IProgressOfHabitService, ProgressOfHabitService>();
53          services.AddSingleton<IServiceOfHabit, ServiceOfHabit>();
54      }
```

Рис. 4.1.3.2. Метод, який налаштовує сервіси

2) Configure - використовується для налаштування конвеєра обробки запитів (middleware pipeline). Тут додається та налаштовується проміжне програмне забезпечення (middleware) для обробки запитів, визначається маршрутизація, обробка статичних файлів, встановлюються користувацькі правила.

```
44      public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
45      {
46          if ( env.IsDevelopment() )
47          {
48              app.UseDeveloperExceptionPage();
49              app.UseSwagger();
50              app.UseSwaggerUI(c => c.SwaggerEndpoint(url: "/swagger/v1/swagger.json", name: "Mentor SET.WebAPI v1"));
51          }
52
53          app.UseHttpsRedirection();
54
55          app.UseRouting();
56
57          app.UseAuthentication();
58
59          app.UseAuthorization();
60
61          app.UseEndpoints(endpoints => endpoints.MapControllers());
62      }
```

Рис. 4.1.3.3. Налаштування конвеєра

Тепер слід протестувати запуск сервера. Для виконання цього пункту використовувався Swagger (фреймворк, що описує структуру API проєкту, який його використовує, та надає змогу розробляти, тестувати та документувати додаток [42]). При запуску проєкту SET.WebAPI, отримаємо наступне.

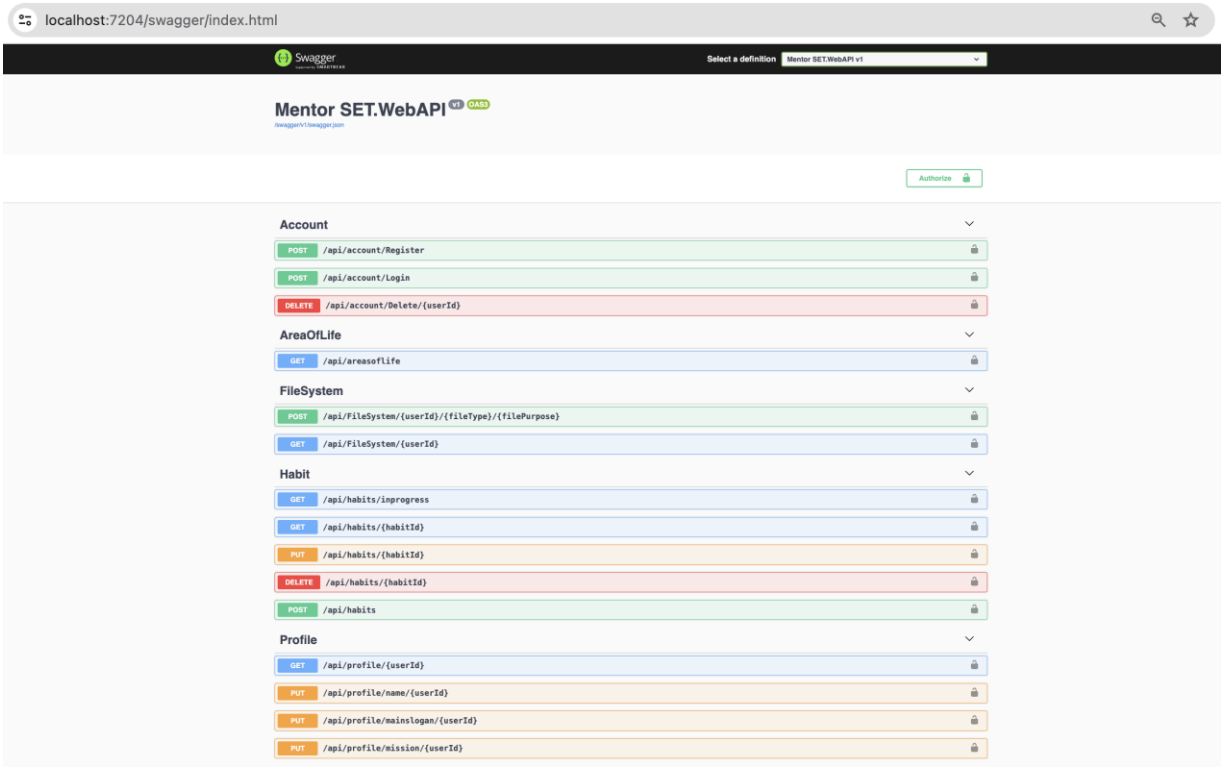

Рис. 4.1.3.4. Первинний вигляд Swagger UI

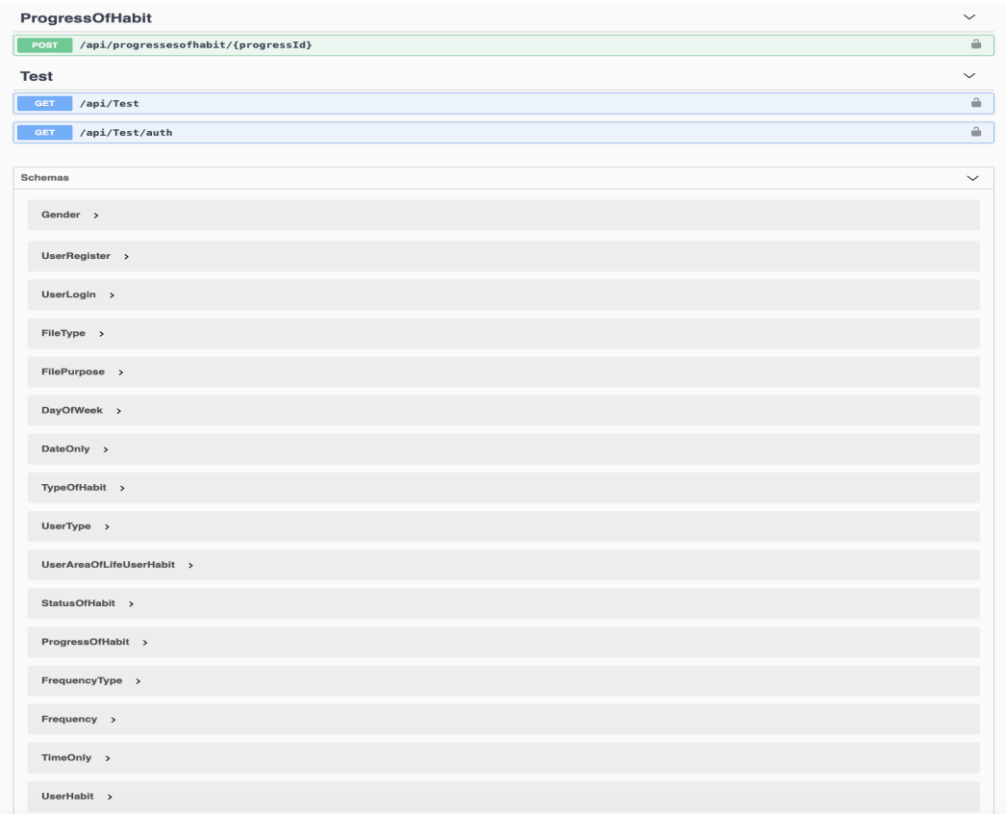

Рис. 4.1.3.5. Продовження

Протестуємо логінування незареєстрованого користувача.

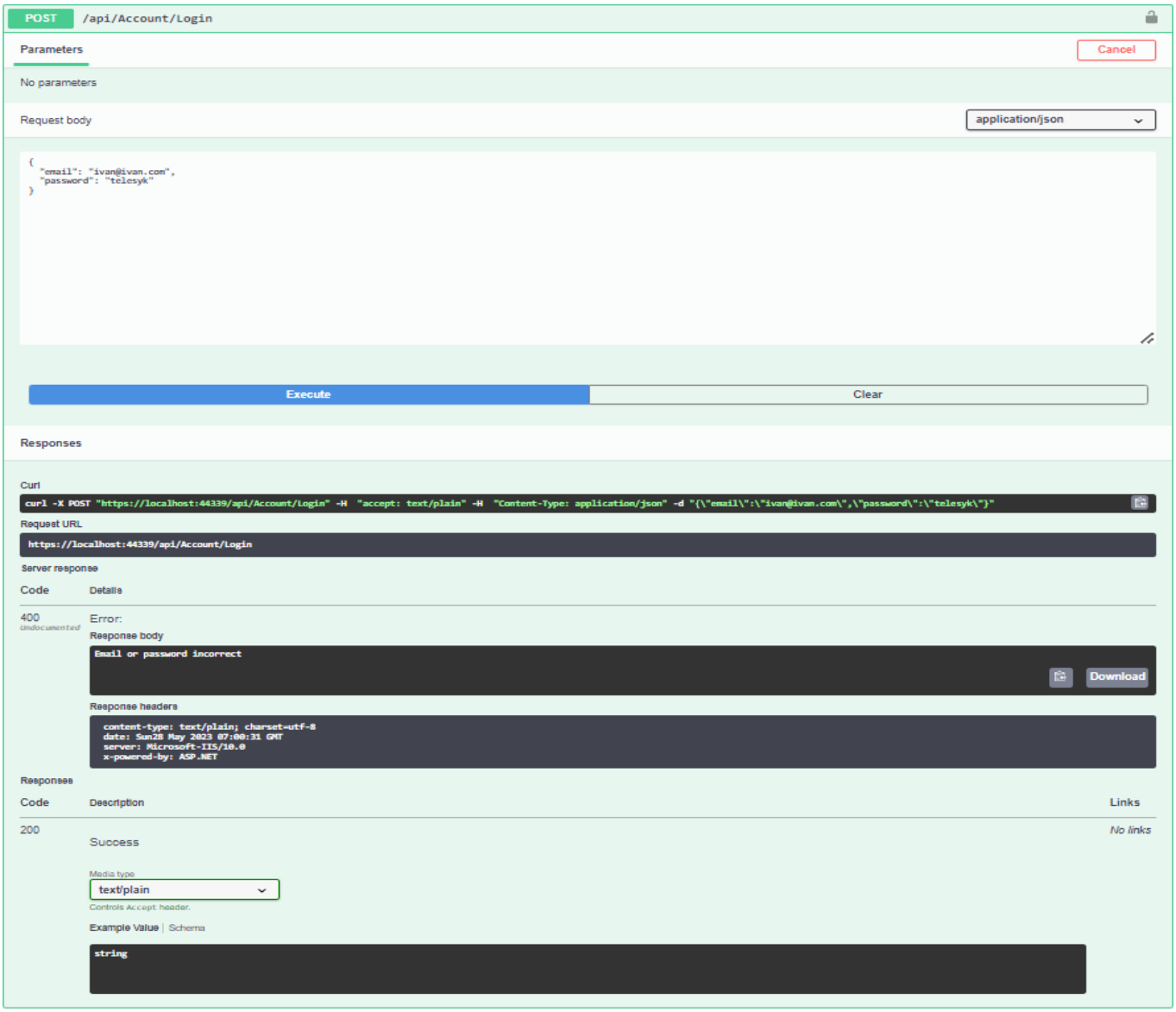

Рис. 4.1.3.6. Результат логінування не зареєстрованого користувача

Як бачимо, в Response body отримано повідомлення, що email чи пароль вказані некоректно. Тут навмисно не вказується точна помилка задля покращення безпеки додатку.

4.1.4. Розміщення серверу на хмарі

Оскільки вибрано Azure для цієї цілі, то необхідно було скористатись сайтом [56] для створення та надання доступу до Azure SQL БД та сервісу застосунку (App Service). Було виконано наступні кроки стосовно цього пункту.

1) Створення Azure SQL БД. Для цього слід вибрати SQL databases → Create → встановити параметри SQL БД.

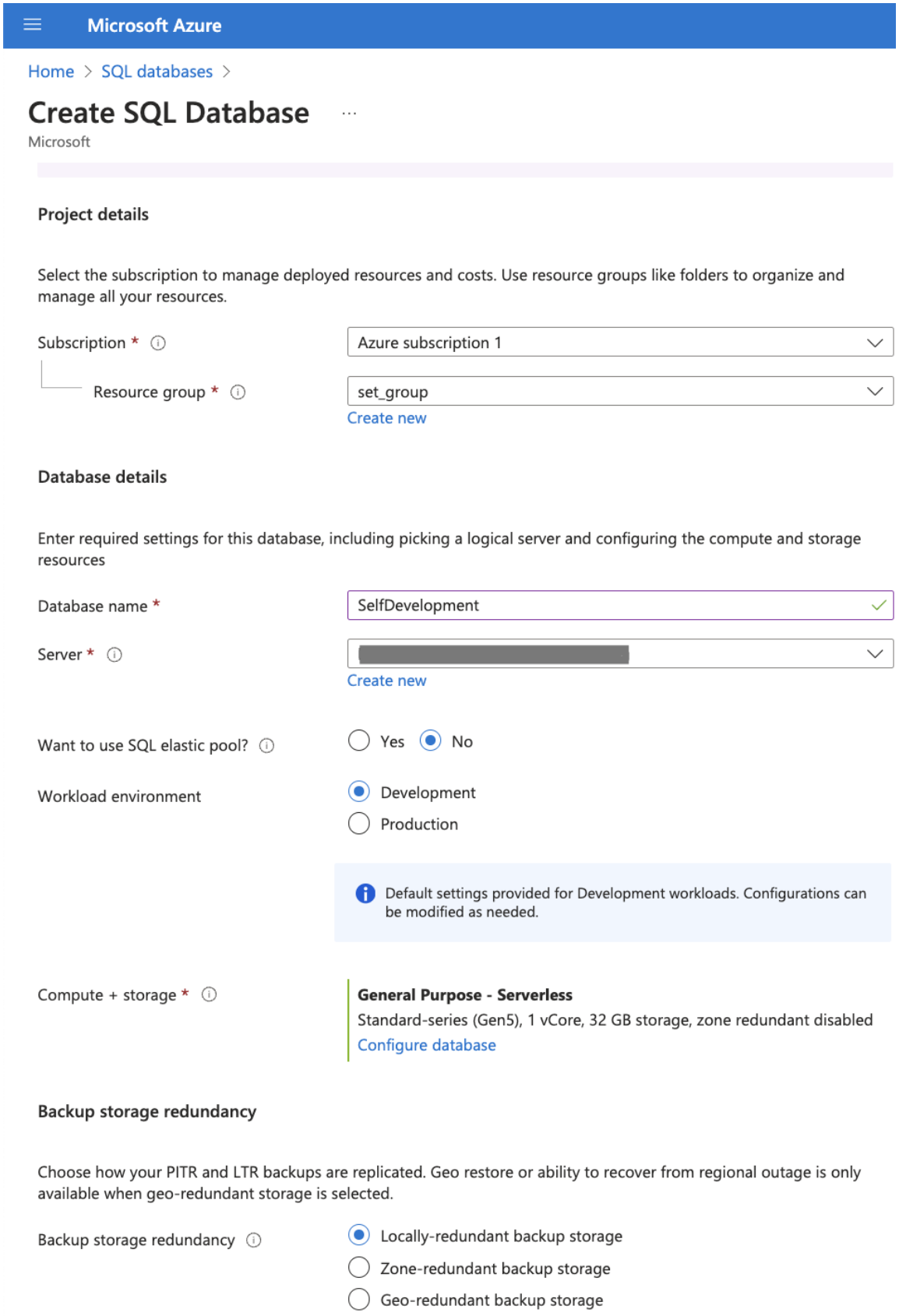

Рис. 4.1.4.1. Приклад створення Azure SQL БД.

2) Встановити наступні NuGet-пакети в проєкт SET.DataAccess для використання EF Core, SQL Server та типів DateOnly та TimeOnly.

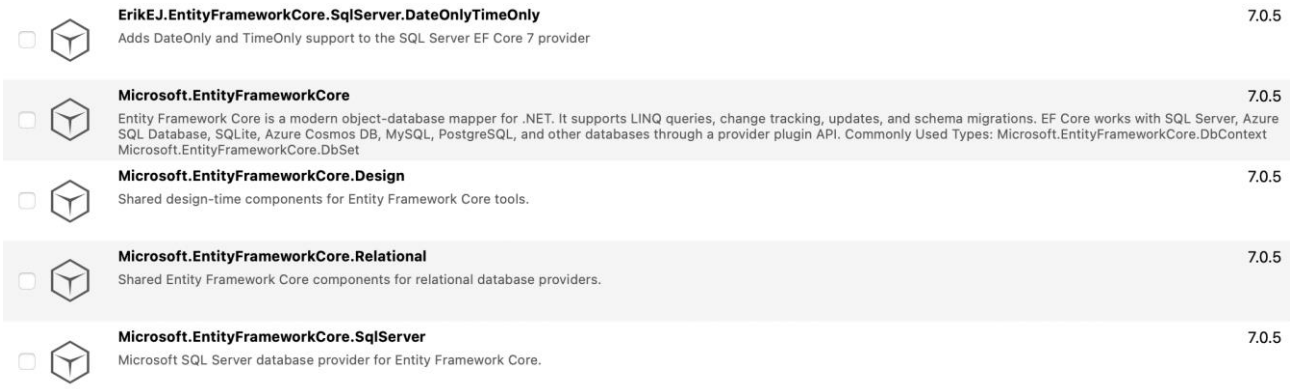

Рис. 4.1.4.2. NuGet-пакети для використання
EF Core та SQL Server в проєкті SET.DataAccess

3) Додати в файл appsettings.json рядок з’єднання до Azure SQL БД.
4) Якщо проєкт запускається в режимі Debug, то використовувати рядок під’єднання до локальної БД, якщо в якомусь іншому режимі (наприклад, Release), то до нового з’єднання, оскільки проєкт в Azure запускається саме в режимі Release й відлагоджувати його немає можливості.

Лістинг 4.1.4.1. Визначення, який рядок з’єднання до БД обирати в проєкті SET.WebAPI виглядає наступним чином.

```
36          services.AddDbContext<AppDbContext>(options =>
37          {
38              string? connectionString;
39      #if DEBUG
40              connectionString = Configuration.GetConnectionString( name: "DefaultConnection" );
41      #else
42              connectionString = Configuration.GetConnectionString( "Azure" );
43      #endif
44              options.UseSqlServer( connectionString );
45          });
```

Рис. 4.1.4.3. Визначення рядка з’єднання

5) Створити App Service (сервіс додатку). Для цього слід на сайті [55] обрати App Services → Create → Web App → задати параметри застосунку.

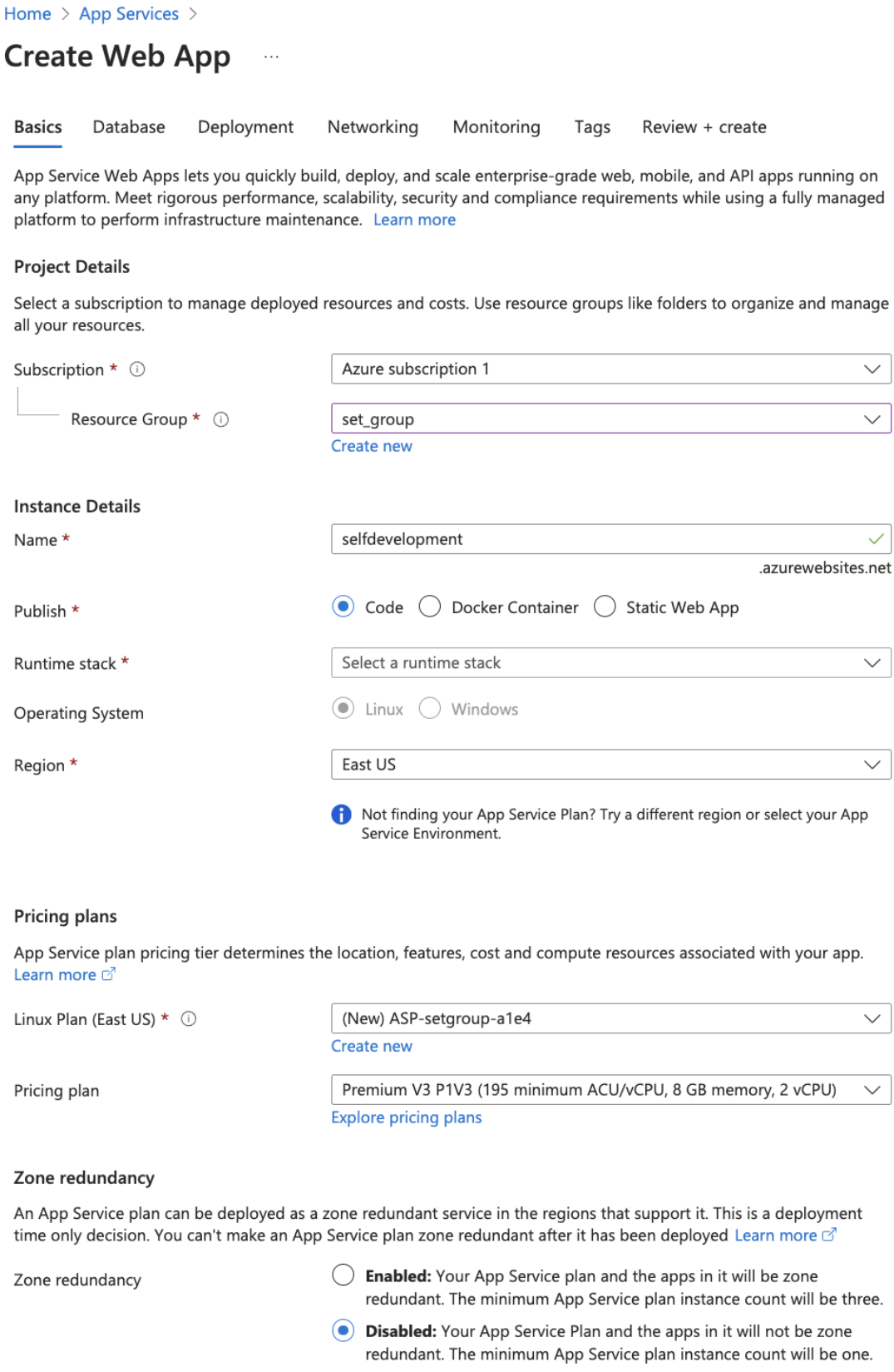

Рис. 4.1.4.4. Приклад задання параметрів для web додатку в Azure

6) Після того, як було задано значення аргументів, слід підготувати проєкт до публікації його на хмарі. Для цього необхідно:

6.1) виправити компіляційні помилки в рішенні;

6.2) автоматично виконувати міграцію БД до найновішої версії.

Лістинг 4.1.4.2. Автоматична міграція БД до найновішої версії в процедурі Startup.Configure в проєкті SET.WebAPI.

```
75          using IServiceScope scope = app.ApplicationServices.GetService<IServiceScopeFactory>().CreateScope();
76          using AppDbContext dbContext = scope.ServiceProvider.GetRequiredService<AppDbContext>();
77          dbContext.Database.Migrate();
```

Рис. 4.1.4.5. Автоматичне виконання
міграції до найновішої версії БД

Таким чином тут отримується AppDbContext та виконується міграція БД.

7) Виконати публікацію серверного додатку на Azure. Для цього слід у VS у вікні «Оглядач рішень» ПКМ натиснути на SET.WebAPI → Publish (опублікувати) → Publish to Azure → зайти в обліковий запис Microsoft, за допомогою якого було створено Azure веб-сервіс та SQL БД → вибрати відповідну Azure підписку.
8) Якщо публікація автоматично не відбулась, то необхідно знову у VS у вікні «Оглядач рішень» ПКМ натиснути на SET.WebAPI → Publish (опублікувати) → Publish to selfdevelopment (назва Azure веб-додатку).

Таким чином серверний застосунок був успішно розміщений у хмарі й за допомогою останнього пункту він постійно оновлювався по мірі його розробки.

4.2. Розробка основи клієнтського застосунку

Для його реалізації використовувалась книга по .NET MAUI [47] з метою застосування оптимальних комерційних шаблонів для цієї платформи.

4.2.1. Архітектура

Оскільки .NET MAUI розміщує код для різних ОС в одній збірці, то архітектура є досить простою. Для даного проєкту в Оглядачі рішень вона виглядає наступним чином.

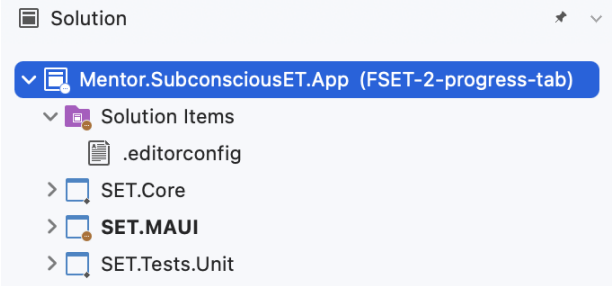

Рис. 4.2.1.1. Архітектура клієнтського застосунку в «Оглядачі рішень»

Файл .editorconfig слугує для автоматизації конвенцій коду. Оглянемо інші елементи. Отож, усього тут є 3 збірки.

1) SET.Core – бібліотека класів, розроблена на платформі .NET Core 7.0. Вона містить платформно-незалежні класи й методи, що надасть змогу при потребі реалізувати веб-застосунок. Її зміст виглядає наступним чином.

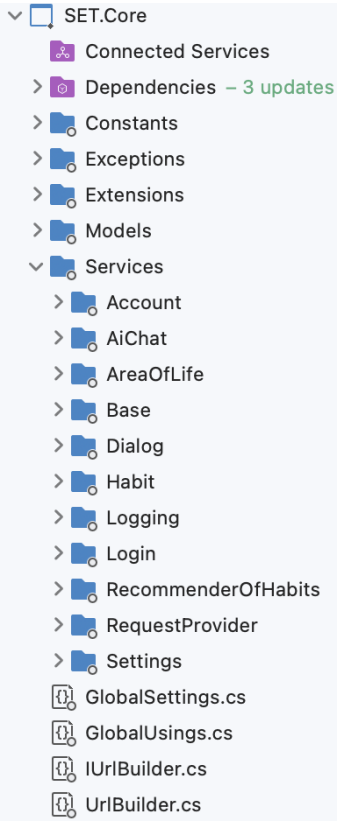

Рис. 4.2.1.2. Структура проєкту SET.Core

Як бачимо, тут міститься чимало сервісів і вони всі застосовуються в наступній збірці.

2) SET.MAUI – власне клієнтський застосунок, розроблений на платформі .NET MAUI версії 8.0. Підтримує 2 ОС: Android та iOS. Для його створення було спершу встановлено розширення DevExpress .NET MAUI Project Templates. Щоб це зробити потрібно у VS вибрати серед пунктів меню справа «Розширення» → «Керувати розширеннями» й скачати те, яке наведено нижче.

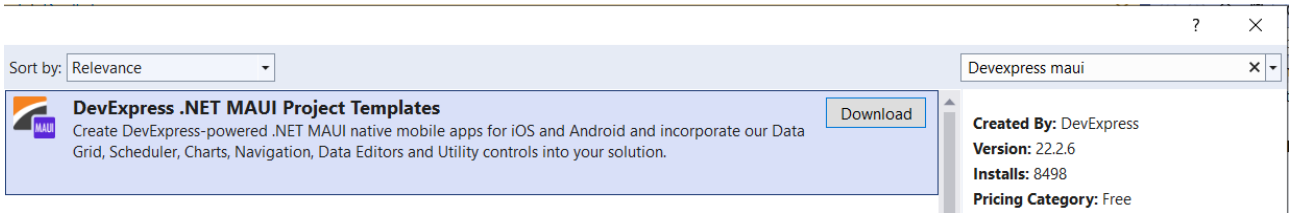

Рис. 4.2.1.3. Розширення DevExpress .NET MAUI Project Templates

Далі потрібно закрити VS → зачекати поки виконається встановлення → знову відкрити VS → «Новий Проєкт» → вибрати наступний вид проєкту.

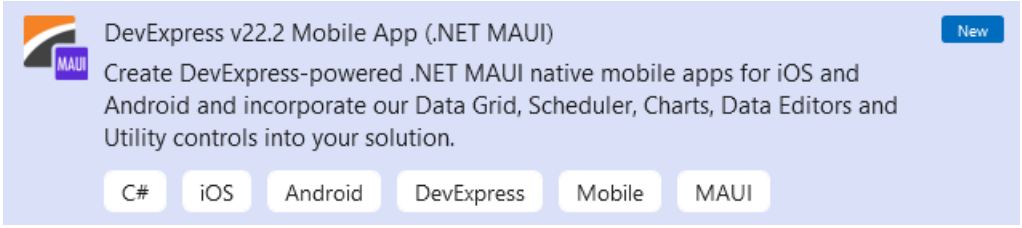

Рис. 4.2.1.4. Проєкт для створення DevExpress .NET MAUI-застосунку

Далі необхідно вибрати назву та шлях до проєкту, після чого його налаштувати за допомогою наступного вікна. Тут показуються одразу ж вибрані конфігурації.

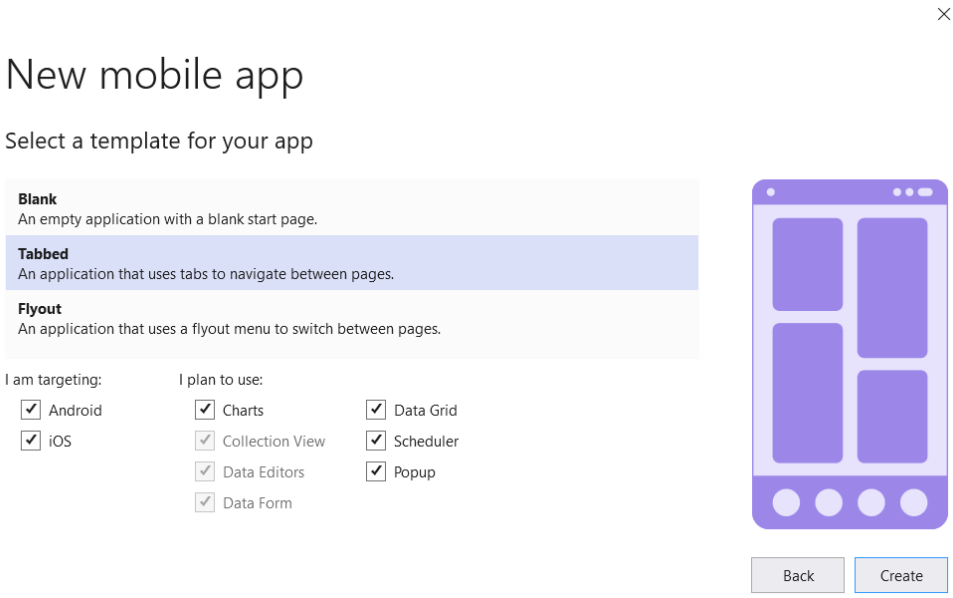

Рис. 4.2.1.5. Вибір специфічного шаблону для розробки додатку.

В результаті даний проєкт одразу ж налаштовує .NET MAUI для розробки лише на вибраних ОС і також надає вибір між різноманітними шаблонами застосунків для швидшої їх розробки та представляє конкретні приклади використання DevExpress (детальніше про це див. підрозділ «Авторизація та автентифікація»).

3) SET.Tests.Unit – проєкт, що містить модульні тести збірки SET.Core. Тут використовується xUnit в якості інструментарію для виконання цього виду тестування [36], тобто як і в backend.

Отож, було вдало побудовано архітектуру клієнтського застосунку, що надало значний поштовх для його подальшої розробки.

4.2.2. Базові сервіси

Базові сервіси - це ті послуги, якими користується більшість додатків. Оглянемо поступово їх.

1) Локалізація

Локалізація – це налаштування додатку з врахуванням культурних і мовних особливостей цільової аудиторії [56]. Додаток наданий час підтримує 2 мови: англійську та українську. Якщо в користувача пристрій є на якійсь іншій мові, то застосунок буде перекладений на англійську. Локалізація даного застосунку складалась з наступних частин.

1.1) Створити файли з ресурсами для зберігання рядків

Вони містяться в директорії SET.MAUI/Resources/AppStrings. Щоб їх створити слід на неї натиснути ПКМ → Додати → Новий файл → Файл ресурсів.

Відповідні файли було вирішено називати з початком LocStrings. В результаті ми отримаємо файл LocStrings.resx. Там ми зберігатимемо переклади на англійську мову. Далі необхідно було додати файл LocStrings.uk.resx і описати українські переклади.

1.2) Додати переклад назви додатку для ОС Android.

1.3) Додати переклад назви додатку для iOS.

Для прикладу, якщо пристрій є на англійській мові застосунок виглядатиме наступним чином.

Рис. 4.2.2.1. Вкладка «Helper», коли пристрій на англійській мові

Проте вона працює таким чином, що відповідає на тій мові, на якій було задано запит. Наприклад, оглянемо наступний рисунок.

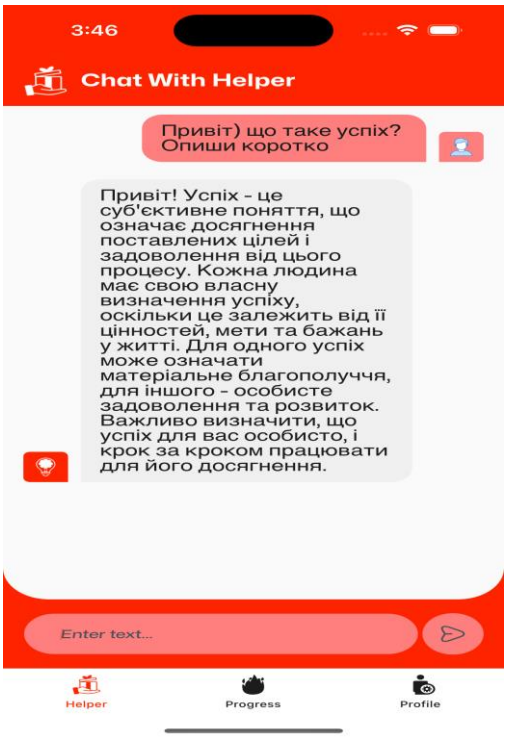

Рис. 4.2.2.2. Локалізація у вкладці «Helper»

Як бачимо, хоч додаток зараз є на англійській мові, проте через те, що запит задано на українській, то й відповідь приходить на ній.

Якщо задати українську назву додатку наступним чином. Відкриємо застосунок Settings → в пошуку введемо “Atomic” → виберемо додаток «Atomic Habits» → натиснемо послідовно на «Language», потім на Українська. В результаті отримаємо наступне.

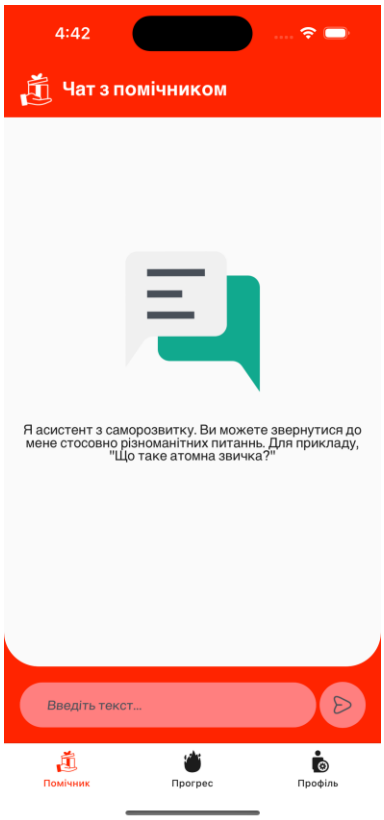

Рис. 4.2.2.3. Вкладка «Помічник» після змінення мови лише для застосунку

Оглянемо локалізацію назви додатку. Якщо задано мову пристрою чи застосунку на англійській мові, то й назва буде на ній.

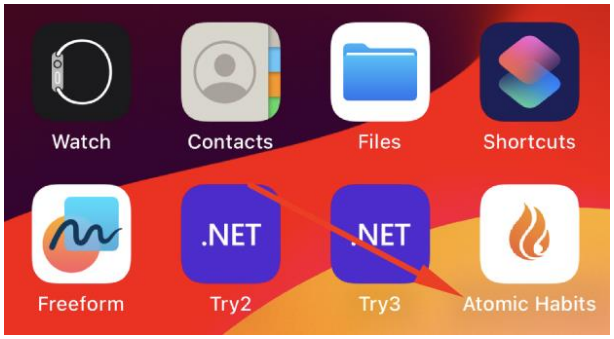

Рис. 4.2.2.4. Назва додатку на англійській мові

Тепер змінимо мову пристрою на українську. В результаті отримаємо наступне.

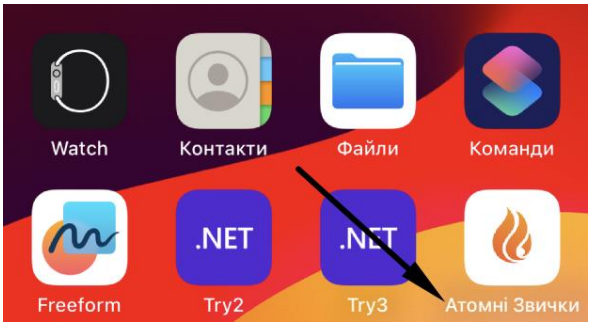

Рис. 4.2.2.5. Назва додатку на українській мові

Отже, застосунок було успішно локалізовано.

2) Діалоговий сервіс

Він надає можливість показувати стандартні діалоги з будь-якого місця виклику. Для прикладу, представлення помилки, вікна-підтвердження. Реалізація діалогового сервісу на платформі .NET MAUI виглядає наступним чином.

```
public class DialogService : IDialogService
{
    public Task ShowAlertAsync( string msg, string title, string buttonLabel )
    {
        return Application.Current.MainPage.DisplayAlert( title, msg, buttonLabel );
    }

    public Task ShowErrorAsync( string msg )
    {
        return Application.Current.MainPage.DisplayAlert( title: LocStrings.Error, msg, cancel: LocStrings.OK );
    }

    public Task<bool> ShowAlertWithTwoBtnsAsync( string msg, string title, string accept, string cancel )
    {
        return Application.Current.MainPage.DisplayAlert( title, msg, accept, cancel );
    }

    public Task<bool> ShowConfirmAsync( string msg, string title )
    {
        return Application.Current.MainPage.DisplayAlert( title, msg, LocStrings.Yes, LocStrings.No );
    }
}
```

Рис. 4.2.2.6. Реалізація діалогового сервісу

В майбутньому планується використати пакет CommunityToolkit та реалізувати власні діалоги для представлення більш привабливих вікон.

3) Кешування налаштувань клієнтського застосунку

Для цього існує власноруч реалізований клас SettingsService, який виконує зберігання на пристрої відповідні налаштування з використанням класу Preferences.

4) Взаємодія зі сервером

З цією метою був імплементований клас RequestProvider, який використовуючи протокол HTTPS, виконує обмін повідомленнями зі сервером.

5) Журнальний сервіс

Для зберігання інформації роботи додатку використовується журнальний сервіс. Він виконує представлення відповідних даних в консолі VS.

4.3. Авторизація та автентифікація

Якщо користувач не зайшов в свій обліковий запис, то при відкритті застосунку відкриється форма логінування.

1) Сторінка логінування.

Вона виглядає наступним чином і відповідає за авторизацію та перехід на сторінку реєстрації.

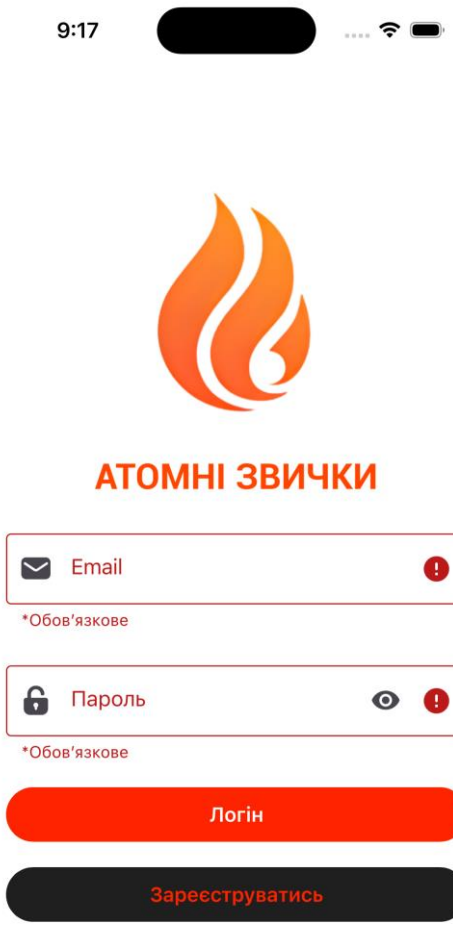

Рис. 4.3.1. Первинний вигляд сторінки логінування

Як бачимо, тут виконується валідація полів Email та Пароль і зазначається, що вони є обов’язковими.

Лістинг 4.3.1. Реалізація поля Email на мові розмітки XAML.

```
<dxe:TextEdit Text="{Binding Email.Value}"
              LabelText="{x:Static loc:LocStrings.Email}"
              AutofillContentType="EmailAddress"
              HelpText="{x:Static loc:LocStrings.Correct}"
              StartIcon="email"
              Keyboard="Email"
              ClearIconVisibility="Never"
              HasError="{Binding Email.IsValid,
                                 Converter={StaticResource IsFalse}}"
              ErrorText="{Binding Email.Errors,
                                  Converter={StaticResource FirstValidationError}}"
              TextChangedCommand="{Binding ValidateEmailCommand}" />
```

Рис. 4.3.2. Контрол поля Email на мові розмітки XAML

По мірі зміни тексту в кожному з них, вона повторно виконується. Наприклад, якщо користувач неправильно заповнить поле Email, то повідомлення під ним зміниться на наступне.

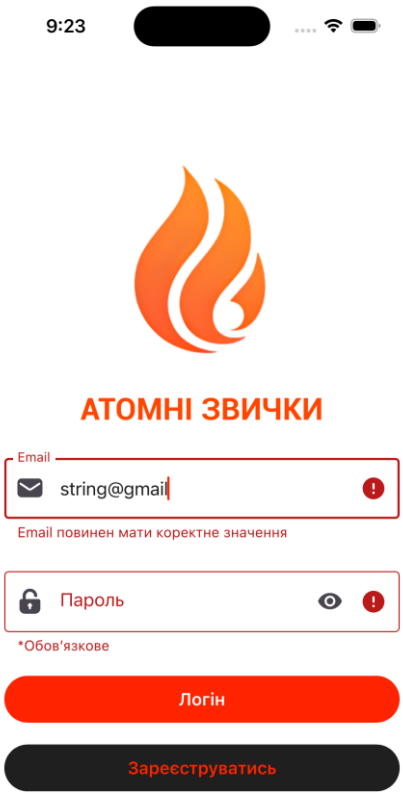

Рис. 4.3.3. Повідомлення, коли користувач неправильно вкаже електронну пошту

Тут також можна помітити, що товщина рамки для виділеного поля збільшується. Це забезпечує бібліотека DevExpress. Коли клієнт коректно заповнить це поле, то форма виглядатиме наступним чином.

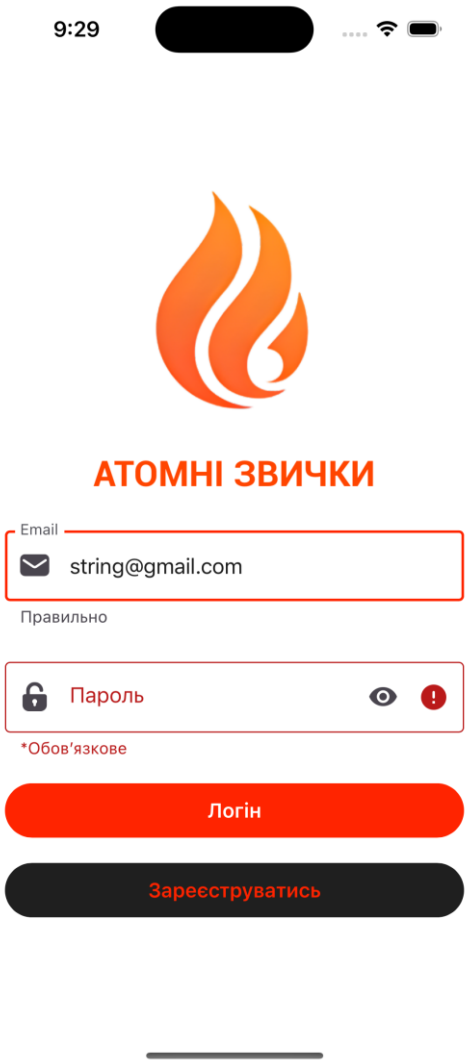

Рис. 4.3.4. Поле Email після правильного його задання

Було вирішено вказувати під полями будь-який текст для того, щоб усі поля не почали рухатись в залежності від його існування.

Валідація поля «Пароль» виконується таким чином, що відбувається лише перевірка на те, чи введено хоча б якийсь символ.

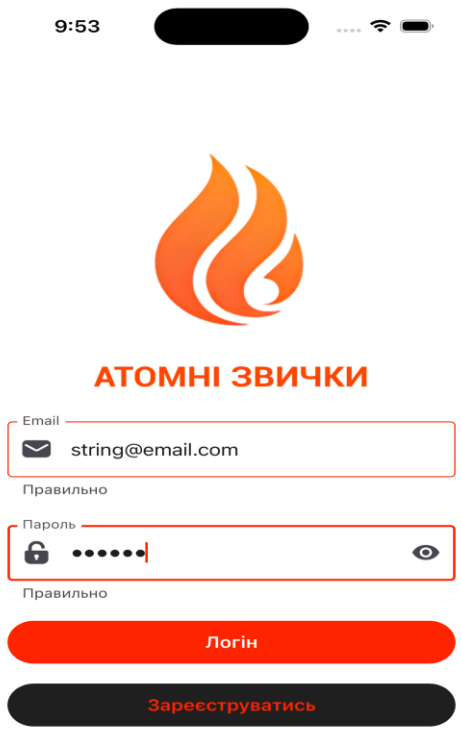

Рис. 4.3.5. Поле введеного паролю

Користувач може натиснути на іконку ока й таким чином побачити введений текст. Для прикладу, це виглядає наступним чином.

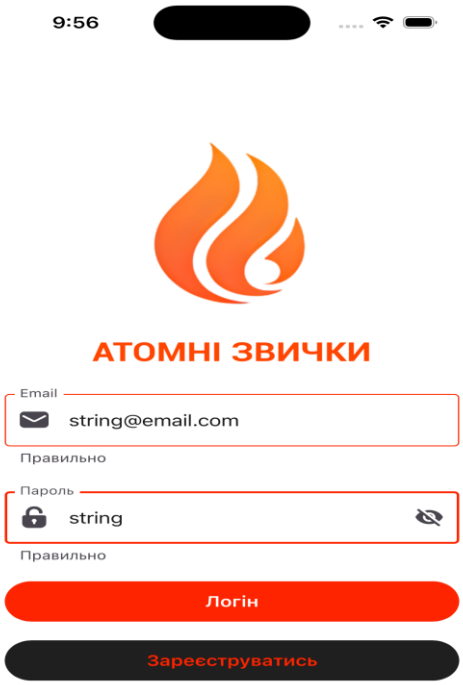

Рис. 4.3.6. Показ паролю

Як бачимо, тут автоматично виконується зміна іконки справа, яка відповідає за приховування паролю.

Якщо користувач натисне на кнопку Логін, коли хоча б одне з полів неправильно заповнене, то жодна з дій не виконається. В протилежному випадку виконається запит на сервер для отримання JW-токену, за допомогою якого користувач матиме змогу отримувати доступ до ресурсів серверу.

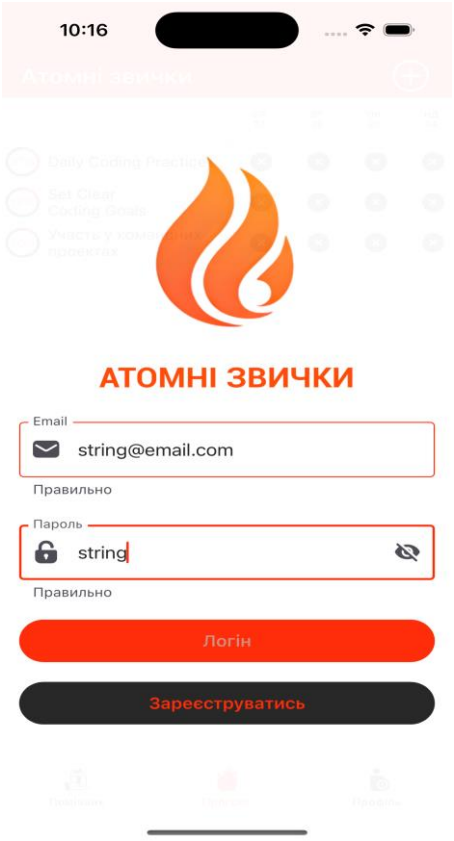

Рис. 4.3.7. Плавний перехід на основну
вкладку авторизованого користувача

Як бачимо, тут перехід відбувається з деякою анімацією й змінюється колір тексту «Логін».

2) Сторінка реєстрації.

Клієнт також може створити новий обліковий запис, натиснувши на сторінці логінування кнопку «Зареєструватись». Первинно відповідна форма виглядає наступним чином.

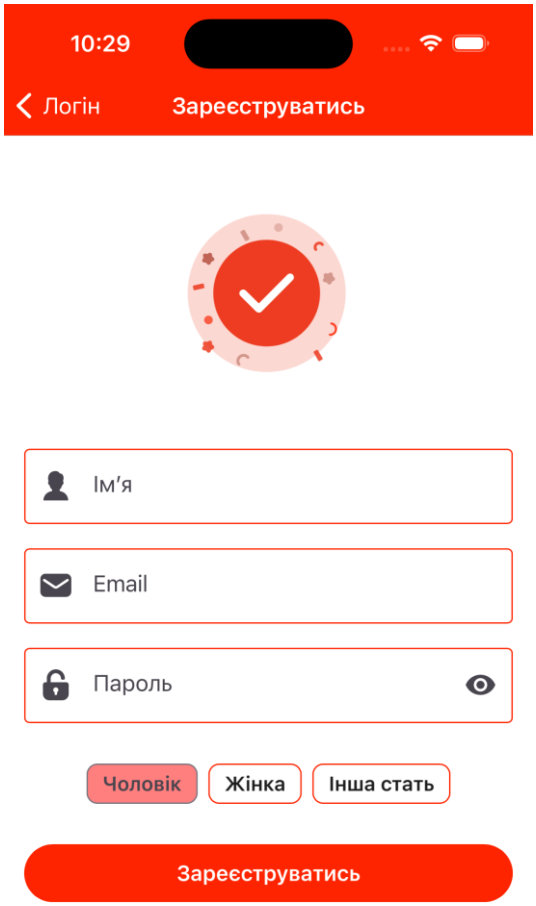

Рис. 4.3.8. Початковий вигляд сторінки реєстрації

Як бачимо, зліва зверху розміщується кнопка для повернення назад на сторінку логінування. Для цього необхідно було задати властивість Shell.NavBarIsVisible="True" для цієї об'єкту цієї сторінки. Зображення зверху є анімованим і безперервно змінюється. Це було досягнуто за допомогою бібліотеки SkiaSharp.

Лістинг 4.3.2. Анімація з використанням NuGet-пакету SkiaSharp.

```
<skia:SKLottieView IsAnimationEnabled="True"
                   RepeatCount="-1"
                   WidthRequest="500"
                   HeightRequest="200"
                   Source="one_check.json" />
```

Рис. 4.3.9. Анімація на сторінці реєстрації

Отож, ця бібліотека надає досить простий спосіб додання анімації, слід лише її стягнути з наступного сайту [57] й додати в папку Resources/Raw.

Тут також відбувається валідація усіх полів по мірі їх зміни. Кнопка «Зареєструватись» виконується лише тоді, коли всі поля були успішно заповнені. Важливо відмінити, що тут пропонується вказати стать користувача, оскільки це впливатиме на те, як система штучного інтелекту буде підбирати рекомендовані звички.

Для представлення роботи форми виконуємо її заповнення.

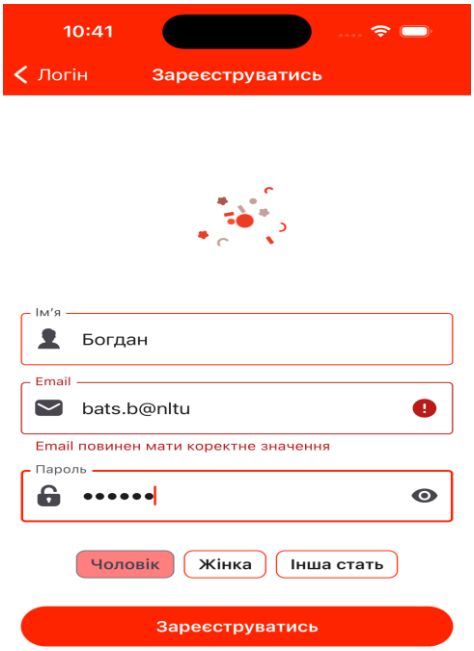

Рис. 4.3.10. Приклад некоректно заповненої форми реєстрації

Як бачимо, тут email є неправильно заповнений і виводиться відповідне повідомлення. Коли користувач введе всі дані коректно й натисне на кнопку «Зареєструватись», то відбувається 2 запити на сервер: реєстрація користувача в додатку та логінування й таким чином отримання JW-токену для подальшого отримання доступу до ресурсів серверу.

4.4. Вкладки клієнтського додатку

Для спрощення переходу між різними сторінками використовується саме навігація за допомогою вкладок і таким чином це спрощує отримання чи заповнення потрібної інформації.

4.4.1. Профіль

На даній вкладці користувач має змогу заповнити такі дані, як своє ім’я (це пригодиться для розробки додатку в майбутньому, коли буде додана його гейміфікація), основний слоган по житті та місію, тобто ціль на усе життя. Останні 2 види інформації використовуються для більш доцільної генерації рекомендованих звичок. Первинно вкладка виглядає наступним чином.

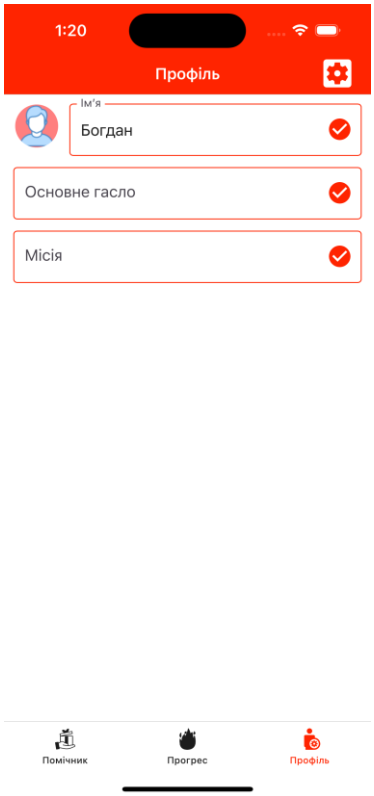

Рис. 4.3.1.1. Первинний вигляд вкладки «Профіль»

Коли користувач хоче відредагувати певне поле й натискає на нього, то відкривається окреме вікно для його зміни. Це зроблено з тією метою, щоб клієнт міг скасувати введені дані й повернути попереднє значення поля. Щоб це зробити йому слід натиснути будь-де поза межами вікна.

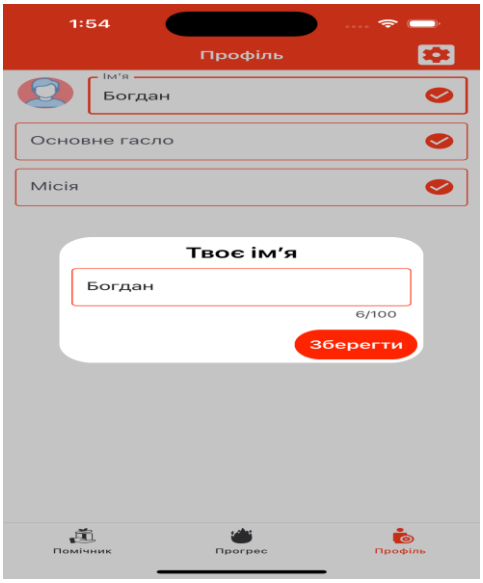

Рис. 4.3.1.2. При натиску на поле «Ім’я»

Спливаюче вікно відображається та зникає з анімацією й відповідно змінюється фоновий колір сторінки. Як бачимо, тут є обмеження на довжину поля, щоб не перезаповнювати БД і таким чином не збільшувати зайвий раз кількість ресурсів на хмарі. Це лімітування присутнє у кожному з полів даної вкладки. Коли користувач вводить значення в нього, то воно автоматично перераховує кількість символів в полі. Для прикладу, оглянемо редагування поля «Місія».

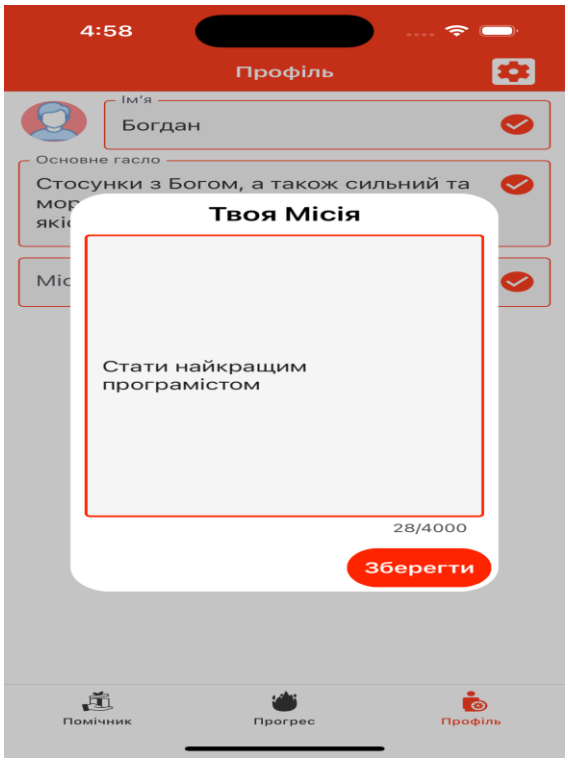

Рис. 4.3.1.3. Редагування поля «Місія»

Як бачимо, на вкладці це значення ще не відображається. Щоб це виправити, слід натиснути кнопку «Зберегти». Таким чином внизу відобразиться повідомлення про успішне оновлення.

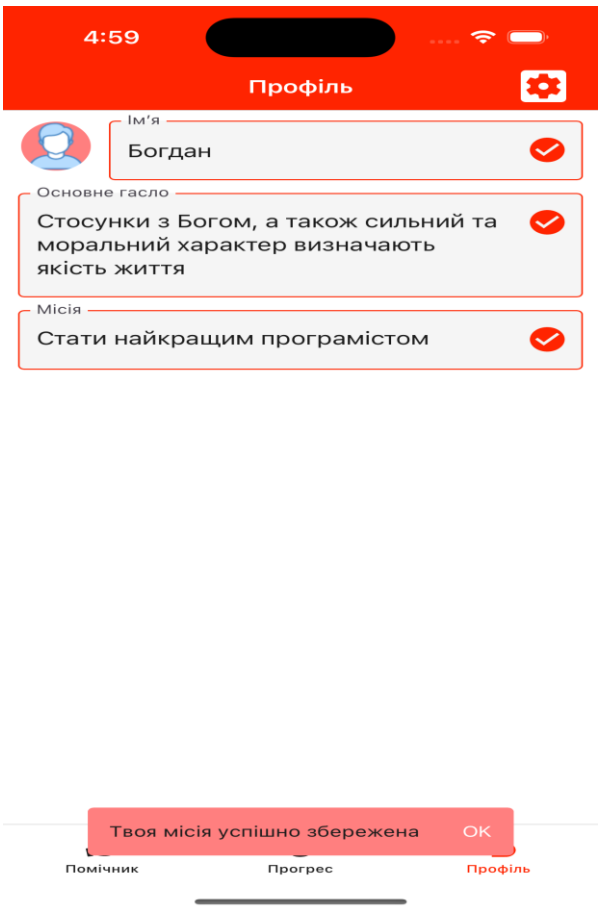

Рис. 4.3.1.4. Результат збереження місії

Після кількох секунд воно автоматично зникає або після натиснення на кнопку «ОК». Стосовно поля «Основне гасло», то воно є дещо меншим, проте працює ідентичним чином.

Лістинг 4.3.1.1. Збереження даних на вкладці «Профіль» з перевіркою, чи правильно заповнене дане поле.

```
[RelayCommand(CanExecute = nameof(CanSaveUserName))]
private async Task SaveUserNameAsync()
{
    bool isSuccess = false;
    await IsUiBusyFor( async () =>
    {
        string url = $"{UrlBuilder.UserName}/{SettingsService.UserId}";
        await RequestProvider.PutAsync( url, UserName.Value, SettingsService.AuthAccessToken );
        isSuccess = true;
    } );

    if (isSuccess)
    {
        await Snackbar.Make(
            LocStrings.YourNameSuccessfullySaved,
            visualOptions: SnackbarHelper.DefaultOptions()
        ).Show();
    }
}

private bool CanSaveUserName()
{
    return UserName.IsValid;
}
```

Рис. 4.3.1.5. Команда збереження ім’я користувача

Отож, тут задається атрибут RelayCommand, який створює команду SaveUserNameCommand, назва якої визначається з імені методу, до якого прикліплюється. Також вказується назва функції, яка буде визначати, чи може команда бути виконана. В тілі основного методу відбувається вказівка, що зараз користувацький інтерфейс зайнятий виконанням відправки запиту на сервер, що стимулює показувати індикатор завантаження.

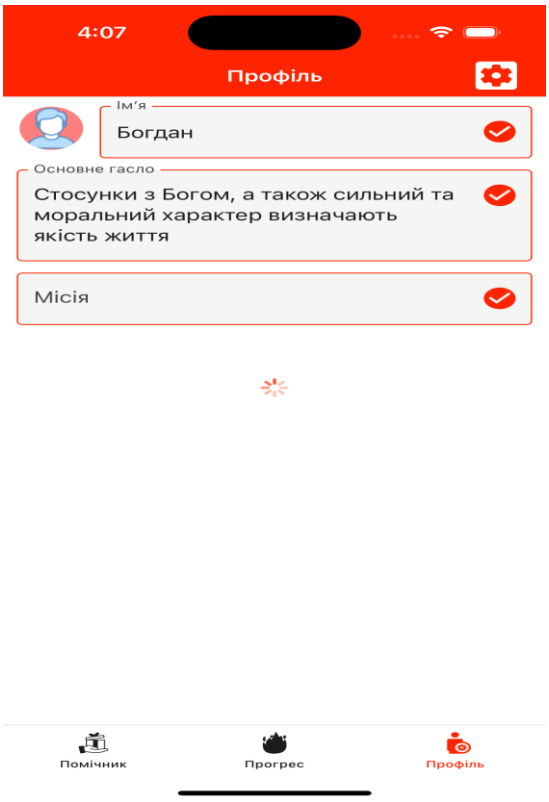

Рис. 4.3.1.6. Представлення індикатору завантаження

Після цього відбувається представлення повідомлення про успішне збереження.

Також, як можна помітити, справа вверху є кнопка налаштувань. Якщо її натиснути, то відкриється відповідна сторінка. Вона виглядає наступним чином.

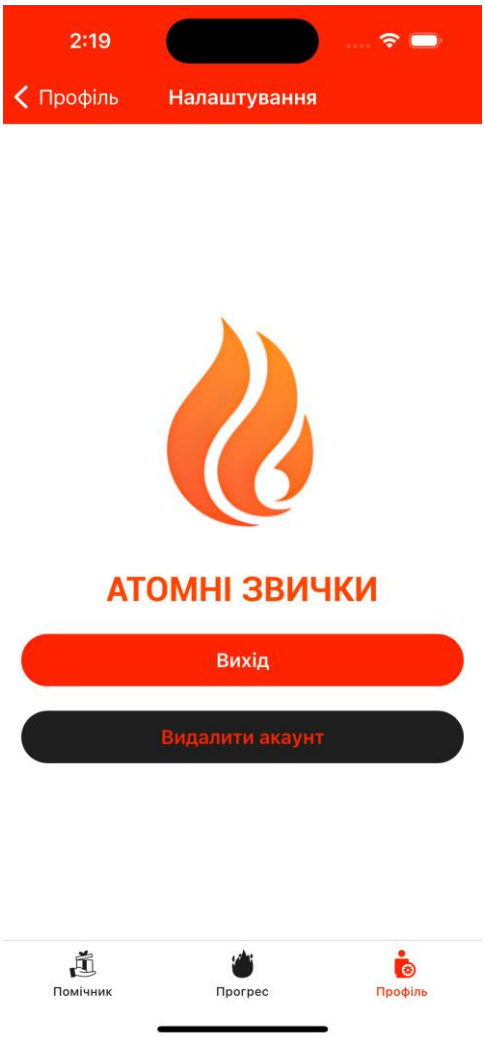

Рис. 4.3.1.7. Сторінка налаштувань

На даний час вона підтримує вихід з акаунту та його видалення, тобто очищення усіх даних користувача з серверу, що покращує конфіденційність даних клієнта. При натисненні на будь-яку кнопку спершу користувач має підтвердити свої наміри. Наприклад, якщо він натисне на кнопку «Вихід», то відобразиться наступне вікно.

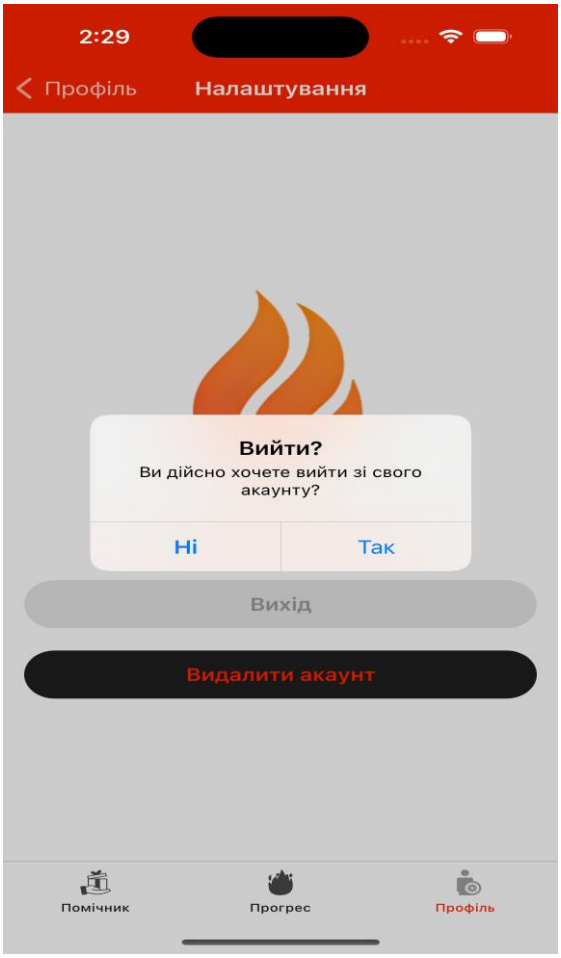

Рис. 4.3.1.8. Підтвердження виходу з облікового запису

Якщо користувач натискає «Так», то виконується відповідна дія й відбувається перехід на сторінку логінування.

4.4.2. Прогрес

Це найосновніша вкладка додатку й розробка саме на ній здебільшого концентрувалась. Вона відповідає за представлення атомних звичок користувача та їхнього прогресу. Первинно вона виглядає наступним чином.

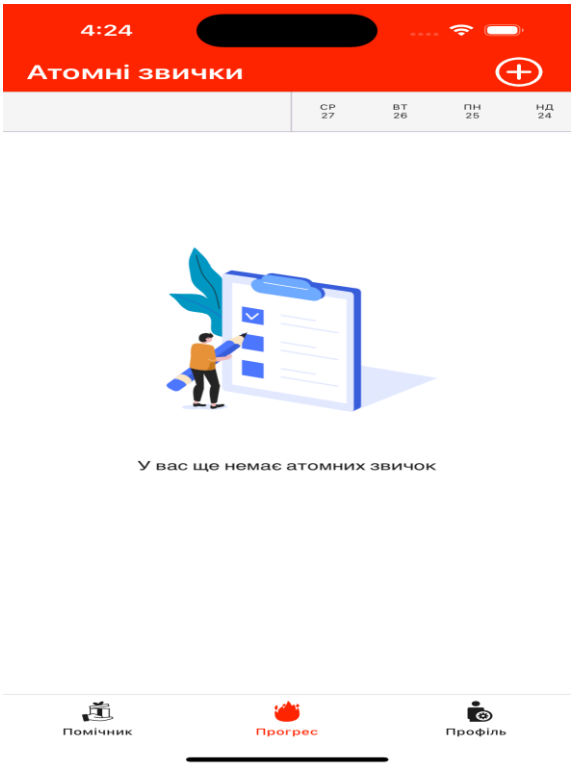

Рис. 4.3.2.1. Початковий вигляд вкладки «Прогрес»

Як видно, якщо користувач немає ще створених звичок, то відбувається представлення анімації й відповідний текст. Якщо користувач натисне кнопку плюс справа зверху, то відкриється наступна форма.

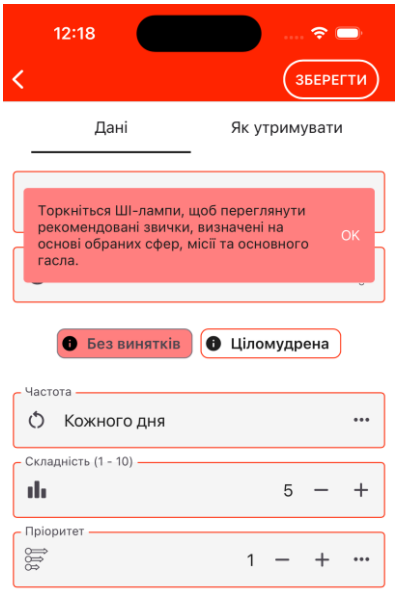

Рис. 4.3.2.2. Вигляд форми створення звички

Як бачимо, тут відображається на кілька секунд повідомлення, як завантажити рекомендовані звички й на основі чого вони будуть генеруватись. Оскільки, там вказано, що також на основі обраних сфер, то виберемо спершу їх. При натисненні на три крапочки в полі «Сфери звички», відповідний список відобразиться наступним чином.

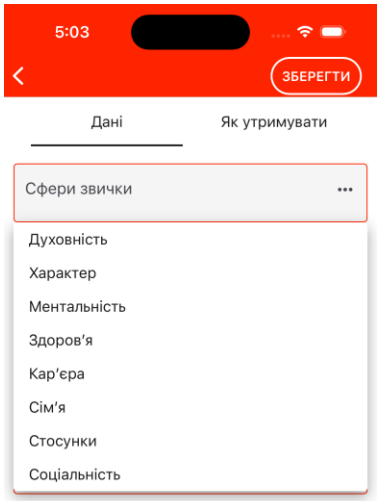

Рис. 4.3.2.3. Список можливих сфер звичок

Тут можливо вибрати одразу кілька сфер звичок. З точки зору особистісного розвитку одні й ті ж самі звички не так часто варто застосовувати в усіх частинах життя, оскільки це часто може заважати стати успішним у всіх них, тому важливо конкретно їх зазначати. Виберемо для прикладу, сферу звички «Кар'єра».

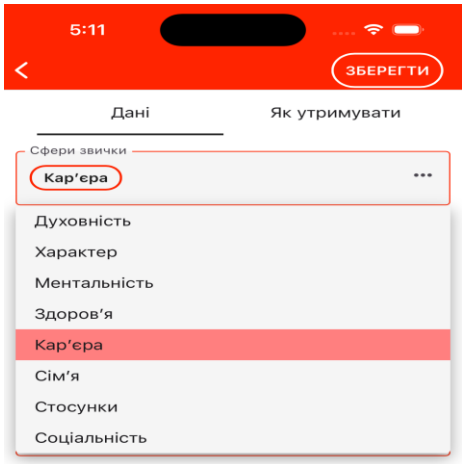

Рис. 4.3.2.4. Вибір сфери звички «Кар’єра»

Щоб сховати цей список слід будь-де натиснути поза цим списокм. Усі ці кольори та представлення окремо прописуються за допомогою відповідних властивостей. Отож, тепер натиснемо на ШІ-лампу в полі «Назва» (звички). Отримаємо наступне.

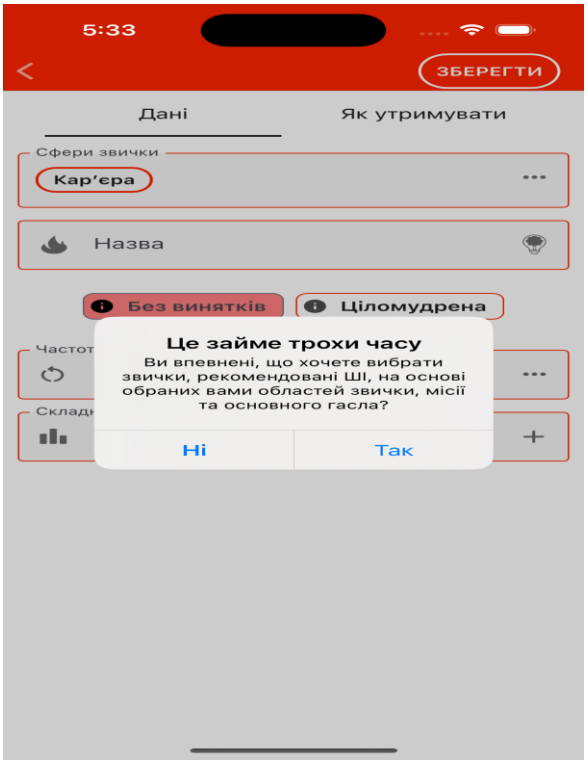

Рис. 4.3.2.5. Підтвердження генерації рекомендованих звичок

Це вікно необхідне для того, щоб користувач зайвий раз не навантжував сервер й був впевнений, що задав усе правильно. Якщо користувач натисне «Так», то представляється наступний лист знизу.

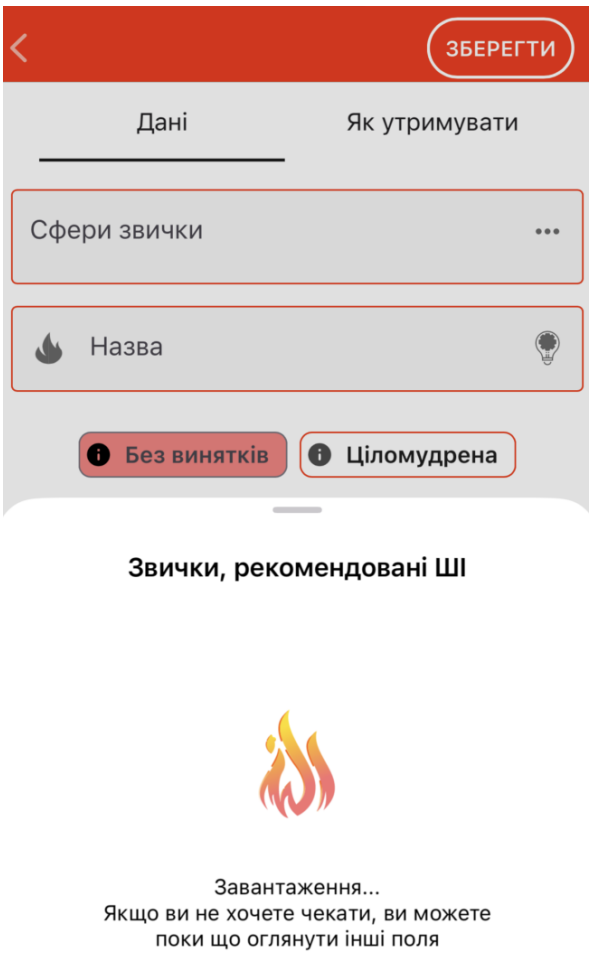

Рис. 4.3.2.6. Завантаження рекомендованих звичок

Зверху, як бачимо, представляється елемент для збільшення й зменшення розміру цього листа. Поки система штучного інтелекту виконує генерацію рекомендованих звичок, користувач може змінити інші поля. Коли вона завершить цей процес відображається на кілька секунд відповідне повідомлення, натиснувши на яке представляться рекомендовані звички. Або ж користувач може знову натиснути на відповідну іконку у вигляді лампочки.

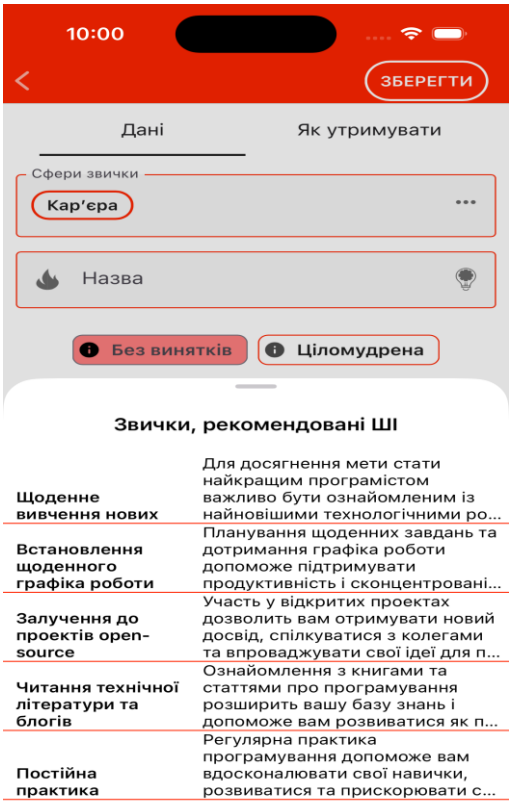

Рис. 4.3.2.7. Згенеровані рекомендовані звички

Як бачимо, тут пропонується одразу 5 різних звичок, причому такі, яких ще немає в користувача й які відповідають його місії. Якщо користувач клікне на один з цих елементів, то одразу заповняться поля «Назва» та «Причина слідувати» й сховається відповідний список.

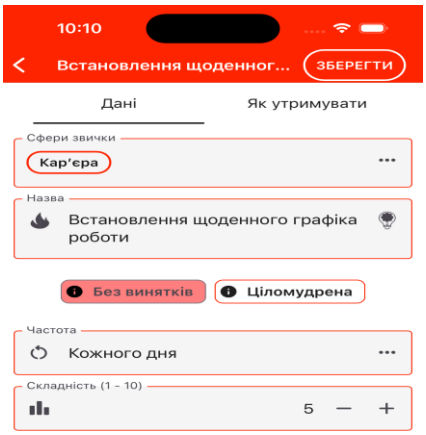

Рис. 4.3.2.8. Результат вибору рекомендованої звички

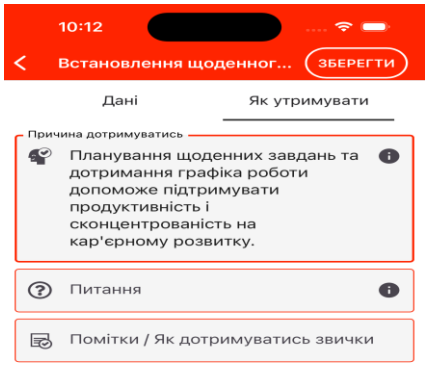

Рис. 4.3.2.9. Продовження

Як можна помітити, звички бувають 2-х типів: без винятків та ціломудрені. При натиску на кожен з цих елементів. Появляється повідомлення з описом кожної з цих звичок причому на позиції відповідного елементу.

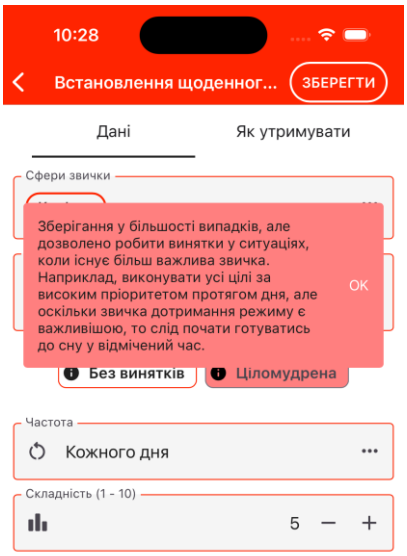

Рис. 4.3.2.10. Повідомлення з описом ціломудреної звички

Тепер щоб вказати частоту звички слід натиснути на поле «Частота». Відповідно до неї система особистісного розвитку визначає, чи є у певний день дедлайн для виконання звички, якщо - так і звичка не є дотримана, то зменшується прогрес її автоматизації відповідно до алгоритму експоненціального згладжування. Якщо користувач натисне на це поле, то виконається анімоване представлення наступного вікна.

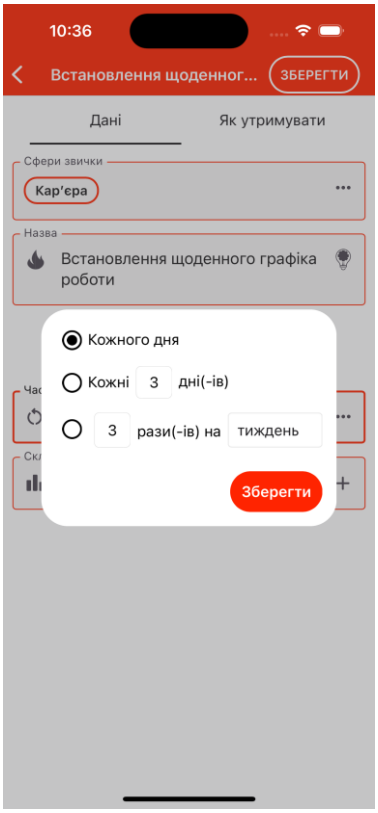

Рис. 4.3.2.11. Спливаюче вікно для визначення частоти звички

При натиску на будь-який елемент поряд з radio button виконається його встановлення. Якщо користувач натисне на «тиждень», то відобразиться наступний список.

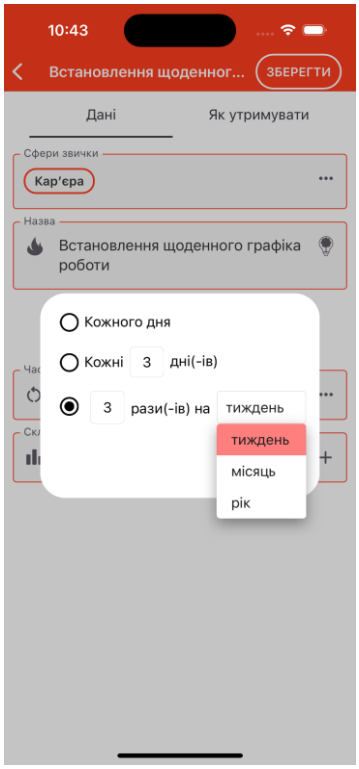

Рис. 4.3.2.12. Доступні періоди частоти виконання звички

Якщо користувач некоректно зазначить кількість повторень звички для вибраної періодичності, то відобразиться вікно з помилкою.

Тільки при натиску на кнопку «Зберегти» виконується встановлення частоти звички.

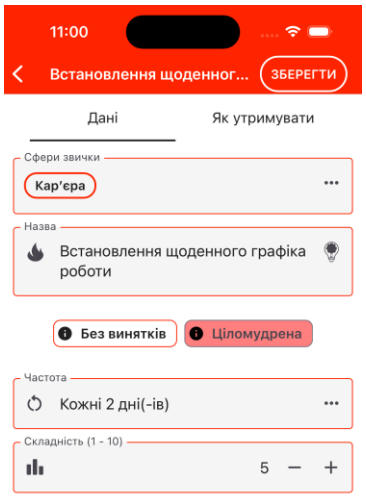

Рис. 4.3.2.13. Встановлена частота звички

Як бачимо, тут також визначається текст відповідно до різної частоти звички.

Тепер перейдемо до поля «Складність». В нього користувач вказує, наскільки є важко для виконання поточна звичка, оцінюючи це від 1 до 10, де 1 – дуже легко, 10 – надзвичайно складно. В залежності від значення цього поля буде визначатись кількість днів для повної автоматизації звички. Тобто чим більша є складність, тим повільніше буде йти відповідний прогрес. В майбутньому планується реалізувати можливість проходження тесту для визначення важкості звички.

Оглянемо вкладку «Як утримувати», яка надає можливість вказати поля, що підсилюватимуть мотивацію, розуміння звички та зазначатимуть як саме їх дотримуватись. Вона виглядає наступним чином.

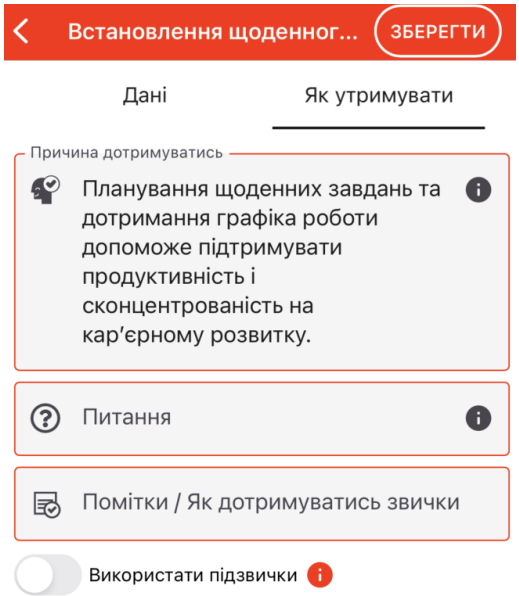

Рис. 4.3.2.14. Вкладка «Як утримувати»

При натиску на іконку інформації показується більш детальний опис поля. Наприклад, для поля «Питання» це виглядатиме наступним чином.

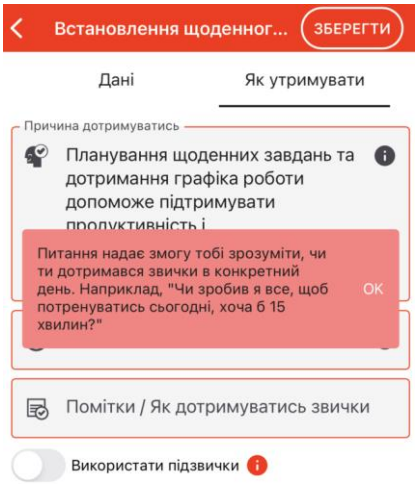

Рис. 4.3.2.15. Більш детальна інформація до поля «Питання»

На цьому ми оглянули усі неочевидні поля. Якщо користувач не заповнить хоча б одне з обов’язкових полів, тобто «Назву» або «Причину дотримуватись», то представиться наступне спливаюче вікно.

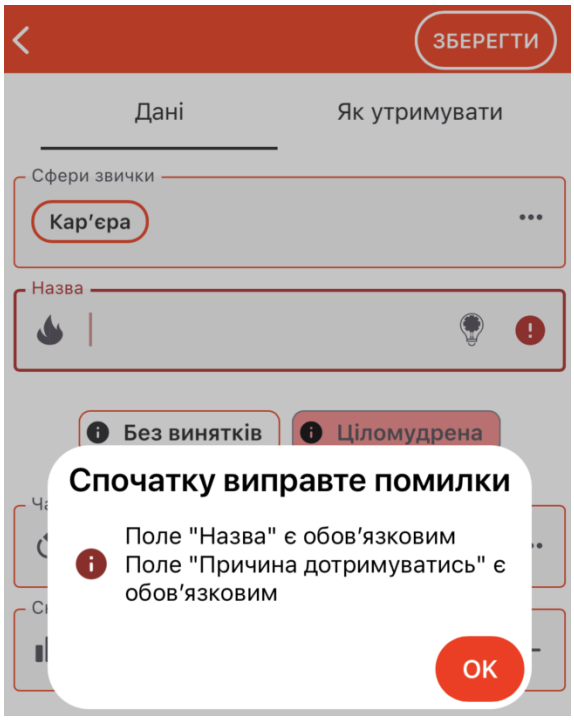

Рис. 4.3.2.16. Спливаюче вікно й зміна відображення поля «Назва» при натисненні кнопки «Зберегти»

Натиснемо ще раз на лампочку в полі «Назва». Одразу ж отримаємо наступне.

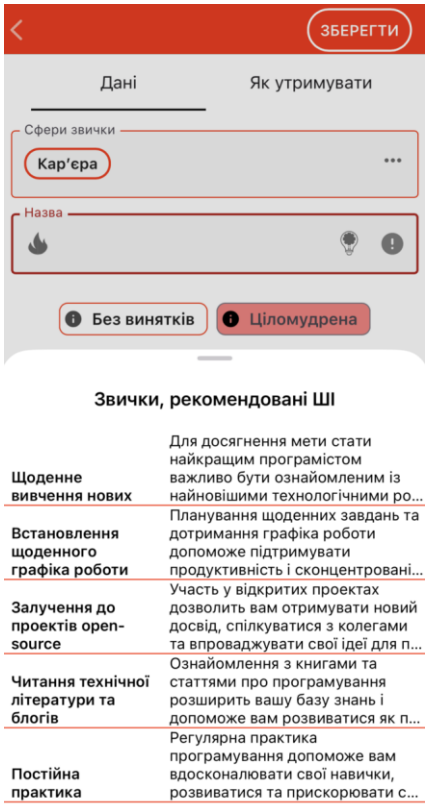

Рис. 4.3.2.17. Повторне завантаження рекомендованих звичок для ідентичної сфери

Оскільки користувач не змінював сфер звичок, то рекомендовані звички одразу підвантажуються з кешу. Виберемо ще раз 2-ий елемент і натиснемо кнопку «Зберегти». В результаті отримаємо наступне.

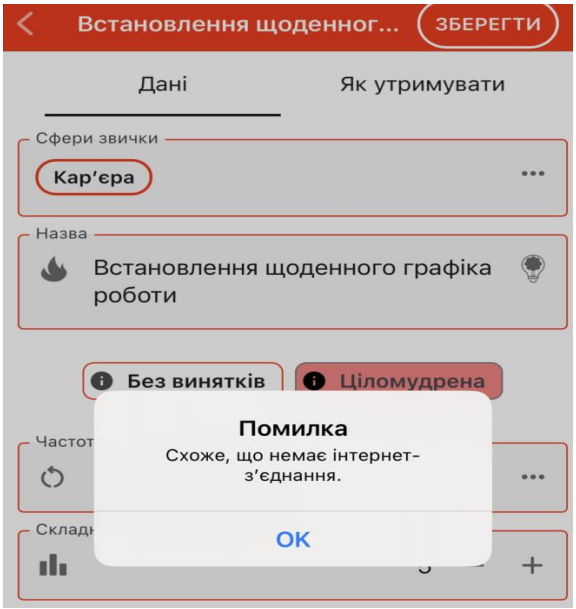

Рис. 4.3.2.18. Вікно з обробленою помилкою мережі

В цей час було вимкнене світло. Як бачимо, помилка відсутності інтернет-з’єднання коректно відображається без затримки. Після його успішного відновлення, натиснемо «Зберегти». В результаті отримаємо.

Рис. 4.3.2.19. Результат збереження першої атомної звички

Як бачимо, тут відображається на скільки відсотків звичка є автоматизована, її назва та, чи була вона дотримана в конкретний день. Тут одразу генерується 66 днів, оскільки це є середня кількість днів для набуття звички згідно з дослідженням [52]. Якщо телефон повернути в горизонтальне положення, то вигляд додатку зміниться на наступний.

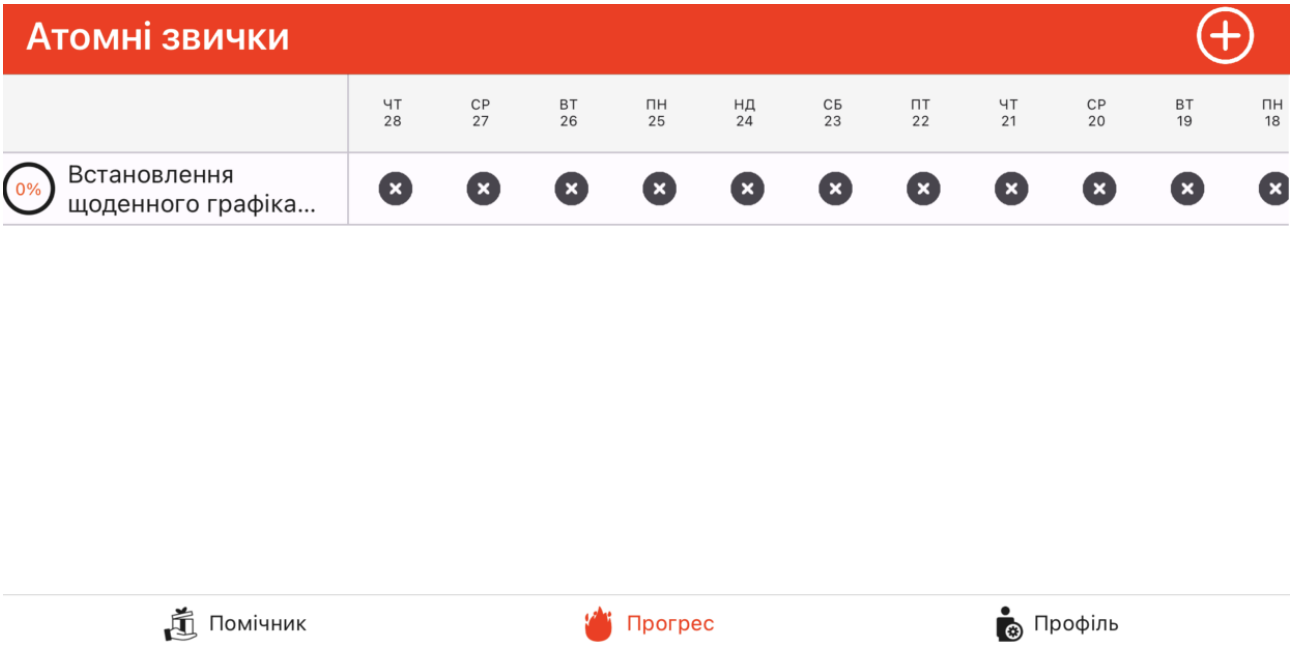

Рис. 4.3.2.20. Вигляд вкладки «Прогрес» при горизонтальному положенні

Якщо користувач натисне, наприклад, на check box в стовпці «СБ 23» (субота, 23 число), то отримаємо наступне.

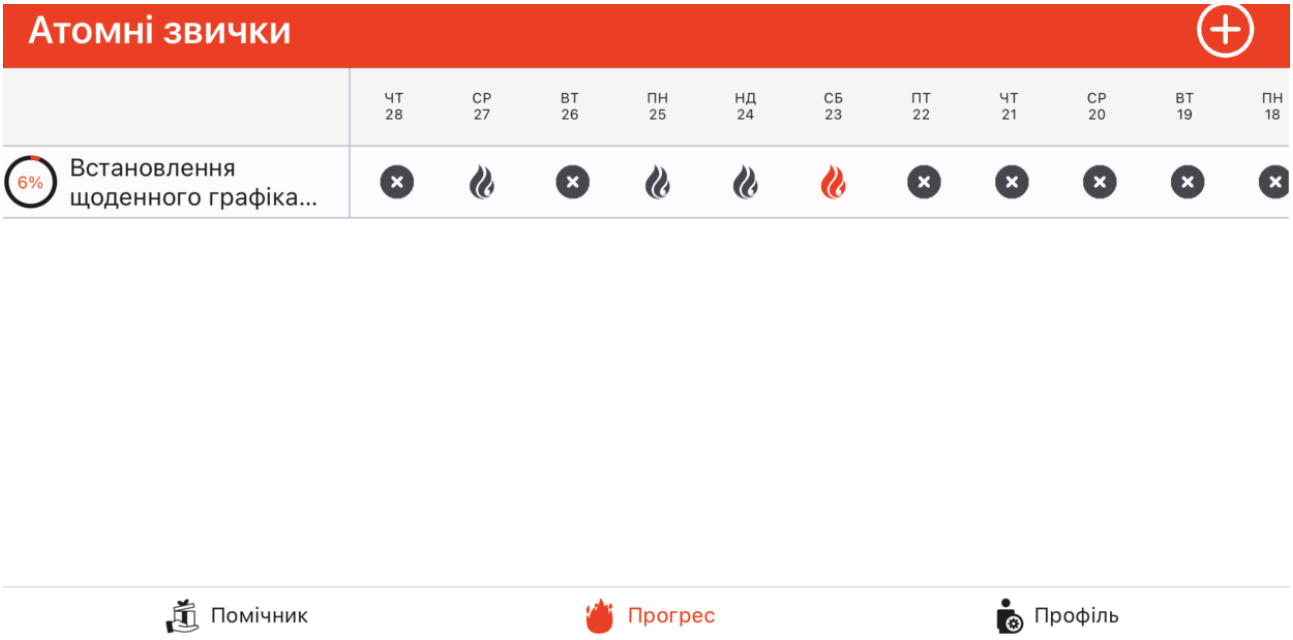

Рис. 4.3.2.21. Зміна таблиці відповідно до дотримання звички в певний день

Лістинг 4.3.2.1. Оброблення випадку, коли користувач безперервно натискає на check box-и дотримання звичок.

```
[RelayCommand]
private async Task ChangeValueOfProgressOfHabitAsync( ProgressOfHabit progressOfHabit )
{
    if (progressOfHabit == null)
    {
        return;
    }

    UserHabit habit = progressOfHabit.Habit!;

    try
    {
        SemaphoreSlim locker = m_isBusyForChangeCompleted[habit];

        await locker.WaitAsync();
        bool previousValueOfIsCompleted = progressOfHabit.IsCompleted;

        try
        {
            progressOfHabit.IsCompleted = !progressOfHabit.IsCompleted;
            await ProgressOfHabitService.UpdateAsync( progressOfHabit );
        }
        catch
        {
            progressOfHabit.IsCompleted = previousValueOfIsCompleted;
        }
        finally
        {
            locker.Release();
        }
    }
    catch ( Exception ex )
    {
        await DialogService.ShowErrorAsync( ex.Message );
        LoggingService.LogCriticalError( ex );
    }
}
```

Рис. 4.3.2.22. Команда ChangeValueOfProgressOfHabitCommand

Як бачимо, тут виконується синхронізація потоків. Причому береться SemaphoreSlim конкретної звички, щоб не заважити користувачу швидко вказувати, що він дотримав усі необхідні звички.

Check box з логотипом і сірим кольором позначається тоді, коли дотримання звички є необов'язковим відповідно до встановленої частоти. Дотримаємо тепер звичку в дні, коли це є необхідно.

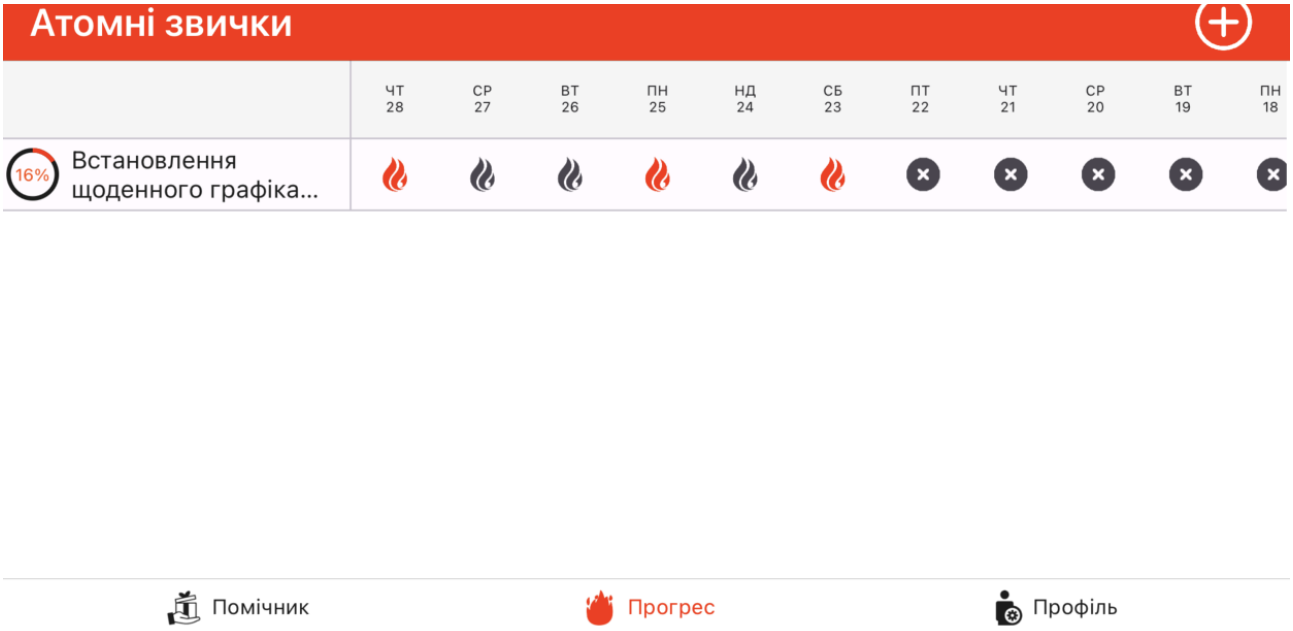

Рис. 4.3.2.23. Дотримання звички в обов'язкові дні

Як бачимо, прогрес автоматизації звички нелінійно змінюється, що відповідає науковому дослідженню [52]. Якщо користувач виконає long press (зажимання пальця протягом кількох секунд) на певній звичці, то вона виділиться наступним чином.

Рис. 4.3.2.24. Результат виділення звички користувачем

Як бачимо, тут також відображаються інші кнопки в панелі інструментів. Тобто це редагування та видалення. Для прикладу, натиснемо редагування. В результаті отримаємо.

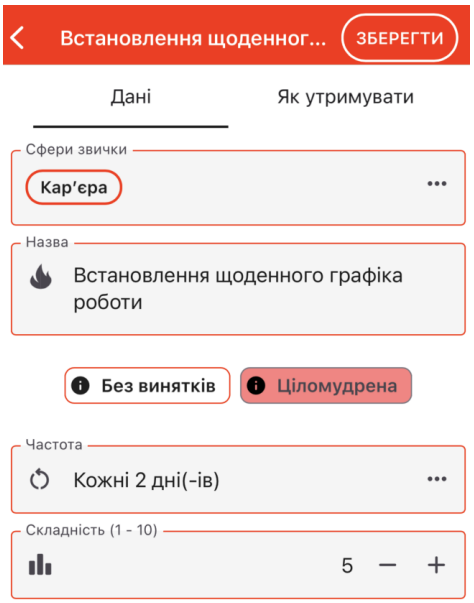

Рис. 4.3.2.25. Завантажені дані зі серверу про звичку

Як бачимо, тут успішно завантажуються дані звички зі серверу. Повернемося тепер назад і спробуємо виконати видалення. Це можна зробити також провівши пальцем зліва-направо на назві звички.

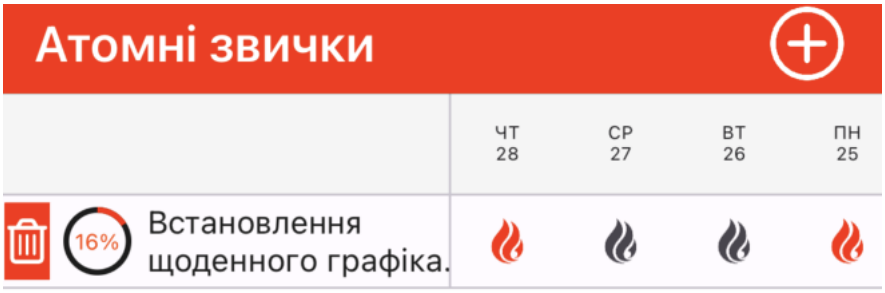

Рис. 4.3.2.26. Іконка видалення звички

Далі слід натиснути на відповідну іконку. В результаті отримаємо підтвердження.

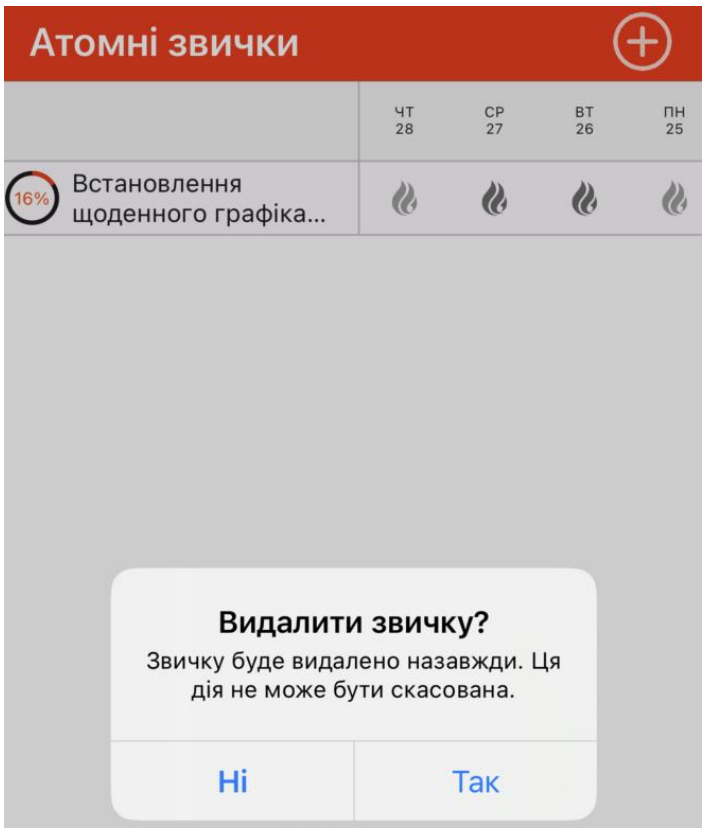

Рис. 4.3.2.27. Підтвердження видалення звички

Натиснемо «Так». Таким чином видалиться звичка з відповідної таблиці на клієнті й виконається видалення усіх даних з серверу, які напряму пов’язані з нею. В результаті отримаємо після перезавантаження додатку (щоб переконатись, що зі серверу вона також видалена).

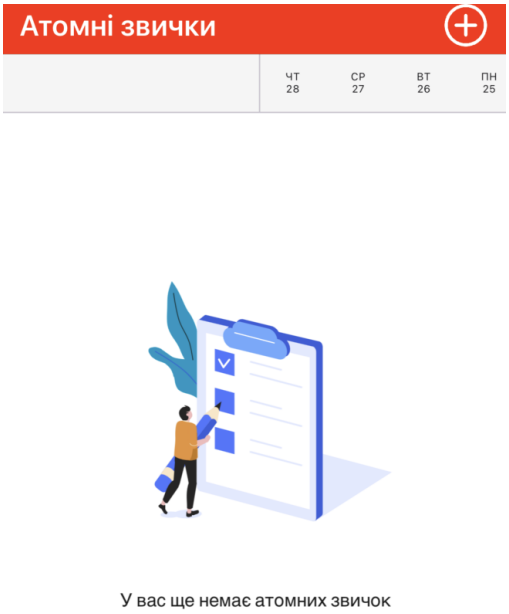

Рис. 4.3.2.28. Результат видалення єдиної звички

На цьому завершується функціональність вкладки «Прогрес». Тепер перейдемо до «Помічник», останню сторінку.

4.4.3. Помічник

Ця закладка була розроблена з метою надання користувачу інформації стосовно саморозвитку. Первинно вона виглядає наступним чином.

Рис. 4.3.3.1. Первинний вигляд вкладки «Чат з помічником»

Тут знову ж таки відбувається анімація зображення, яке видно на середині сторінки. Досить довгим був процес визначення, як найкраще генерувати відповіді на запитання, оскільки більшість з відповідних сервісів є платними. Проте було знайдено деякі безкоштовні, але вони генерують зовсім некоректні відповіді. Отож, все-таки було знайдено компромісне рішення й вибрано Chat GPT версії 3.5-turbo, який вимагає низьку оплату й генерує чудові відповіді на різноманітні запитання. Також його можна налаштувати спеціальним чином, щоб він розумів контекст додатку. Щоб переконатись, що це було вдало зроблено, привітаємося з ним.

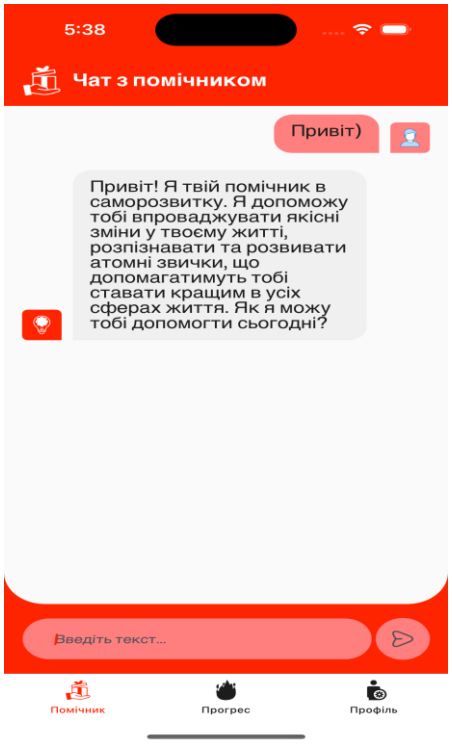

Рис. 4.3.3.2. Привітання з помічником,
створеного на основі штучного інтелекту

Як бачимо, у цій відповіді помічника міститься короткий опис того, на чому він концентрується. Причому відповідь генерується кожного разу в різному вигляді, хоч суть є дуже схожою. Запитаємо його щось про звички.

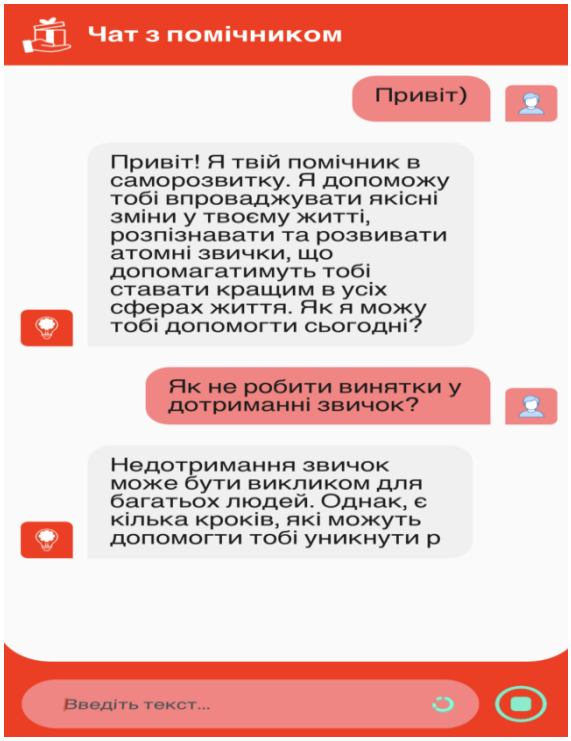

Рис. 4.3.3.3. Процес генерації відповіді

Варто відмітити, що тут відповідь помічника показується не відразу вся, а формується поступово таким чином надаючи можливість користувачу одразу з нею починати ознайомлюватись.

Лістинг 4.3.3.1. Представлення повідомлення помічника.

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xaml-comp compile="true" ?>
<Grid xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
      xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
      x:Class="SET.MAUI.Views.Templates.HelperMessageTemplate"
      xmlns:msgs="clr-namespace:SET.MAUI.Messages"
      xmlns:dxe="clr-namespace:DevExpress.Maui.Editors;assembly=DevExpress.Maui.Editors"
      xmlns:dx="clr-namespace:DevExpress.Maui.Core;assembly=DevExpress.Maui.Core"
      x:DataType="msgs:DisplayMessage"
      ColumnDefinitions="35, 10, Auto, *"
      Padding="16, 0, 0, 0"
      x:Name="G_HelperMessageTemplate">

    <Grid Grid.Column="0"
          x:Name="G_HelperIcon"
          WidthRequest="35"
          HeightRequest="35"
          VerticalOptions="End">
        <BoxView Color="{AppThemeBinding Light={StaticResource Primary}, Dark={StaticResource DarkPrimary}}"
                 CornerRadius="5, 5, 5, 0" />
        <dx:DXImage Source="ai"
                    TintColor="White"
                    WidthRequest="22"
                    HeightRequest="22"/>
    </Grid>

    <Grid Grid.Column="2"
          MaximumWidthRequest="260"
          x:Name="G_Answer">
        <BoxView CornerRadius="16, 16, 0, 16"
                 Color="{AppThemeBinding Light={StaticResource TertiaryLight}, Dark={StaticResource TertiaryDark}}"/>
        <dxe:MultilineEdit BackgroundColor="{AppThemeBinding Light={StaticResource TertiaryLight}, Dark={StaticResource TertiaryDark}}"
                           TextFontSize="18"
                           TextColor="{AppThemeBinding Light={StaticResource LightNormalText}, Dark={StaticResource DarkNormalText}}"
                           x:Name="ME_Answer"
                           Text="{Binding Text}"
                           TextFontFamily="MonaSansMedium"
                           BorderThickness="0"
                           ClearIconVisibility="Never"
                           IsReadOnly="True"
                           BoxPadding="5"
                           Margin="10, 5"
                           VerticalOptions="Start"
                           BoxMinHeight="10"
                           MinimumHeightRequest="18"
                           TextChanged="ME_Answer_TextChanged"/>
    </Grid>
</Grid>
```

Рис. 4.3.3.4. XAML-код представлення відповіді помічника

Для того, щоб відповідь поступово генерувалась, слід обробляти подію зміни тексту й дивитись, чи необхідно змінювати висоту контейнеру елементів. Тут BoxView використовується для представлення фонового вигляду повідомлення та іконки.

Варто відмітити, що при створені відповіді користувач може відмінити її, натиснувши кнопку поряд з полем введення тексту. Для прикладу, введемо наступний запит і зупинемо його.

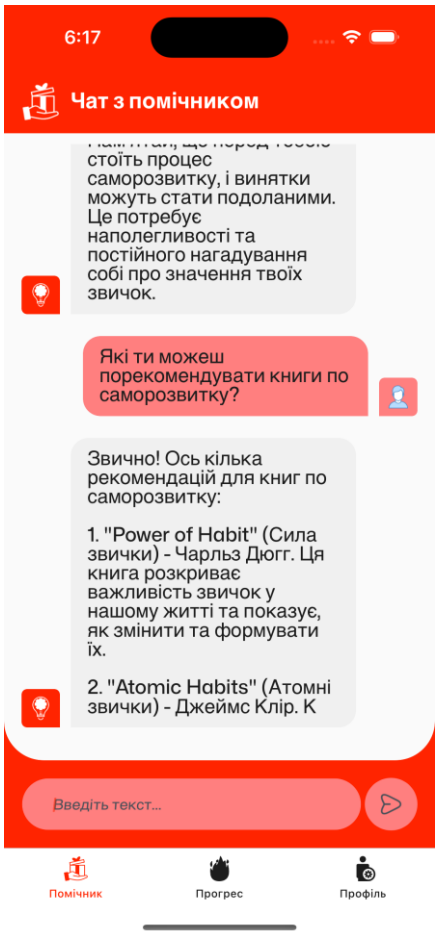

Рис. 4.3.3.5. Зупинена відповідь помічника

Як бачимо, тут в процесі зупинено генерацію відповіді. Це легко досягається за допомогою об’єкту CancellationTokenSource та його методу Cancel. Звісно, після того, як цей екземпляр було переведено у стан відміни, його слід перестворити, що мати можливість далі генерувати відповіді та відміняти їх.

Також варто відмітити, що помічник запам’ятовує увесь чат повідомлень, навіть відмінених. Наприклад, попросимо ще раз порекомендувати книги по саморозвитку.

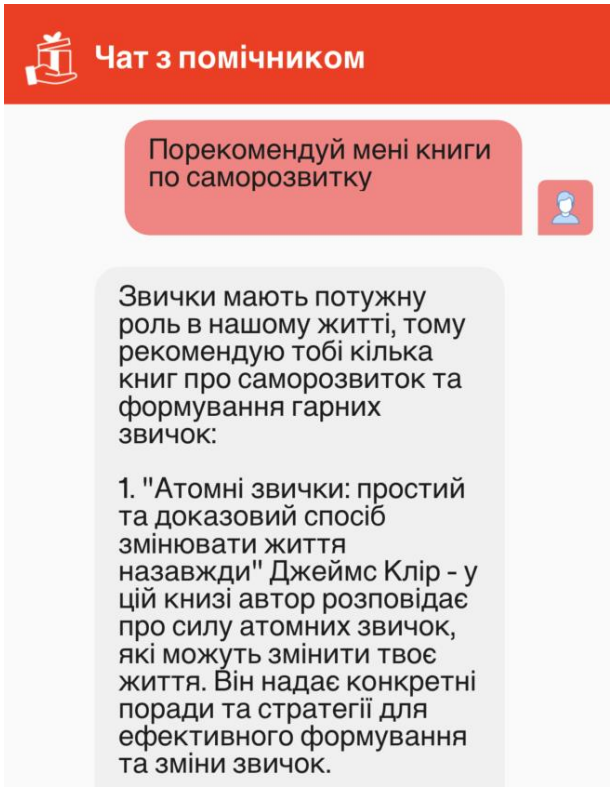

Рис. 4.3.3.6. Запит та частина відповіді помічника

Тепер задамо запит для детальнішого опису першої рекомендованої книгу. В результаті отримаємо наступне.

Рис. 4.3.3.7. Приклад запам’ятовування контексту помічником.

Як бачимо, помічник вдало розуміє контекст повідомлень. Також ми можемо задати запитання, не пов’язана зі саморозвитком.

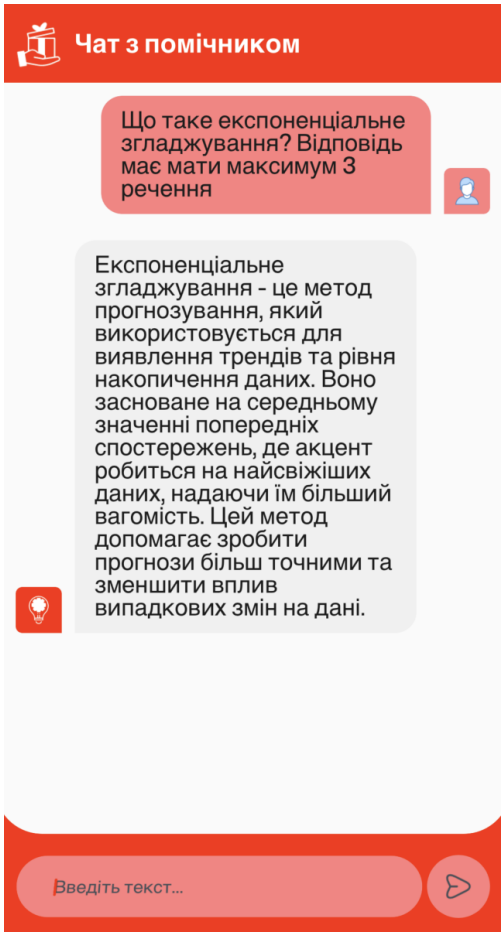

Рис. 4.3.3.8. Питання, що напряму
непов’язане зі саморозвитком, і відповідь на нього

Отож, помічник здатен вдало запам’ятовувати контекст, генерувати питання, як по саморозвитку, так і по іншим сферам.

4.5. Висновки до розділу

Описавши все програмне забезпечення додатку, можна дійти підсумку, що всі відмічені цілі було досягнуто. Розроблена основа застосунку надасть змогу більш просто й швидко додавати нову функціональність додатку. Також можна дійти висновку, що хоч .NET MAUI є новою платформою в розробці кросплатформних застосунків і має чимало недоопрацювань, вона надає досить широкий спектр можливостей. Створені безкоштовні бібліотеки (наприклад, DevExpress.Maui, CommunityToolkit.Maui) їх ще додатково розширюють, виправляють помилки і допомагають реалізовувати справжні комерційні проєкти.

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ

Мультиплатформна система особистісного розвитку «Атомні звички» є відповіддю на актуальні виклики у сфері саморозвитку, де надзвичайно швидко зростає кількість відповідної інформації, що неминуче створює хаос, а, отже, й перегорання, невдоволення, виснаження та зниження рівня мотивації (детальніше див. підрозділ «Аналіз предметної області). На даний час не існує жодного додатку, який би допомагав знизити усі ці пункти за рахунок обмеження кількості нових звичок для опрацювання на один період часу.

5.1. Опис ідеї проєкту

Ідея проєкту полягає в допомозі користувач створювати звички відповідно до їхньої статі, місії (цілі на життя), основного гасла, попередньо визначених звичок, а також вибраних сфер, до яких клієнт хоче її застосовувати. Наприклад, якщо в користувача місією є «Створити найуспішнішу ІТ-компанію по розробці корисних застосунків», а сфера звички – «Кар’єра», то серед запропонованих може бути «Складати графік на день».

Функціональність додатку також намагається допомогти користувачу дотримувати визначені звички. Для цього існують наступні поля.

- «Складність» - в залежності від неї по-різному змінюється прогрес автоматизації звички. Таким чином застосунок надає змогу краще визначити, чи звичка стала набутою, чи ні.
- «Причина дотримуватись» - підвищує мотивацію клієнта слідувати звичці.
- «Питання» - за допомогою нього користувач може зрозуміти, чи дотримана відповідна звичка, чи ні.
- «Помітки або як дотримуватись звички» - особисті нотатки користувача стосовно того, як ефективно дотримуватись звички й робити мінімум винятків.

В цей же час клієнт має змогу отримати важливу інформацію про дотримання звичок, саморозвиток та інші сфери, скориставшись вкладкою помічник.

Рис. 5.1.1. Первинний вигляд вкладки «Помічник»

Він працює таким чином, що запам’ятовує контекст усіх повідомлень і може давати зворотній відклик на відповідні запити користувача (див. детальніше підрозділ «Помічник»).

Звісно ж, окрім створення й редагування звичок, клієнт ще бажає відслідковувати їхній прогрес та знати, чи може він приступати до наступної звички та чи має він дотриматись її у певний день відповідно до встановленої частоти. Задля цього існує вкладка «Прогрес» (див. детальніше відповідний підрозділ).

Отже, функціональність додатку є досить непростою для реалізації, проте дуже зручною й корисною для користувача. Оглянемо тепер ситуацію на ринку та виконаємо спілкування з клієнтами та психологами.

5.2. Дослідження ринку й спілкування з клієнтами та психологами

1) Дослідження ринку

Згідно з ситуацією на ринку мобільних додатків (в App Store та Google Play), категорія «Саморозвиток» чи її подібні не є найпопулярнішими, проте зустрічаються застосунки, які мають велику кількість завантажень (більше 1 млн).

Якщо оглянути рейтинг додатків в Google Play [7], то можна побачити, що є 1 схожий додаток – «Трекер звичок Loop» [49], що представляє розвиток автоматизації звичок. Але там не має обмеження по максимальній кількості нових звичок, що може створити значні проблеми для користувача, можливості отримати рекомендовані звички відповідно до особливостей кожної людини (ціль на життя, основне гасло, стать, інші звички), одержати цінну інформацію стосовно саморозвитку та інших сфер. Він має понад 5 млн. завантажень, що є дуже високим показником. Ідея цих застосунків є досить схожою до поточного проєкту, тому потенціал останнього є чималим.

Згідно з рейтингом сайтів [43], веб-ресурсів по особистісному розвитку є не дуже велика кількість в порівнянні з іншими й в глобальному списку вони займають досить низьке місце (3182 - найвищий рейтинг сайту такої категорії).

Отже, хоч мобільних додатків по саморозвитку є не так багато й вони не є дуже популярними, але якщо вдало реалізувати проект, то можна зацікавити велику кількість клієнтів.

2) Аналіз спілкування з клієнтами та психологом

- Який досвід в комунікації з різними можливими користувачами?

Багато отримано позитивних відгуків й певний досвід щодо того, яккраще спілкуватись з потенційними клієнтами:

a) Краще одразу пропонувати ідею проекту, а не спершу запитувати чи вони мають змогу надати feedback. Якщо вони в конкретний момент не зможуть, то дадуть відгук у вільний час.

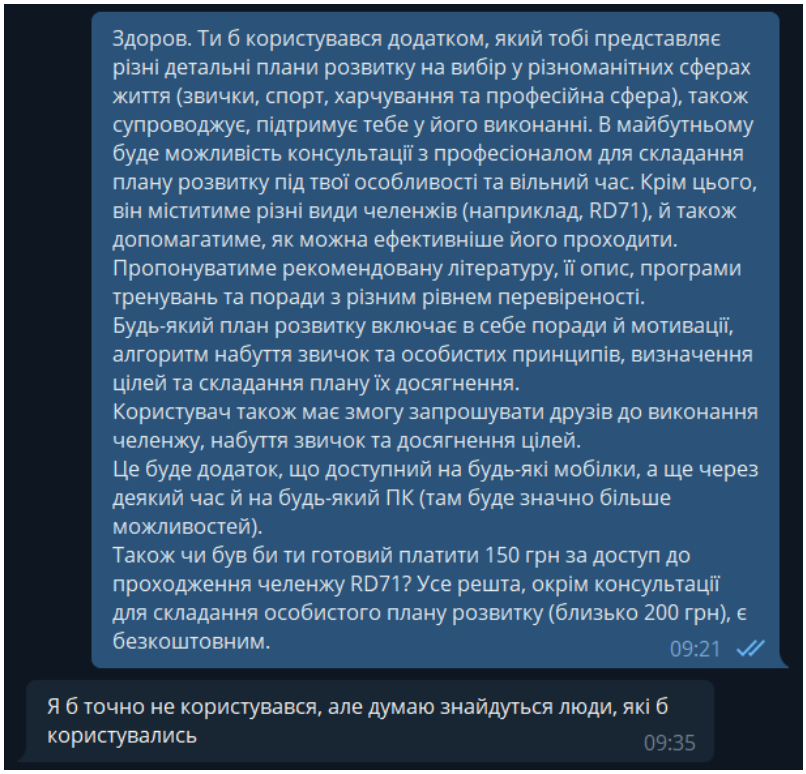

Рис. 5.2.1. Одразу повне формулювання ідеї проєкту користувачам.

b) Варто просити відгуки у людей з дещо різними парадигмами (уявленнями, думками), щоб знати, чи захопить цей додаток велике коло людей.
   Було виявлено, що це вдасться, якщо зважати на різні фактори (вік, стать).
c) Краще наголошувати на те, що мені цікава лише їхня думка щодо ідеї й я нічого їм поки не пропоную: люди не люблять, коли їх переконують чи одразу хочуть щось продати без їхнього запиту.
d) Також не варто одразу задавати багато питань одному й тому ж самому клієнту. Краще задати менше, щоб не дуже напружувати його й мати змогу ще раз з ним поспілкуватись.

- Якщо можливі клієнти вважають ідею хорошою, то чи готові вони оплатити перший чек?

В усіх позитивні відгуки щодо ідеї проекту, деякі вважають, що зараз для нього слушний час.

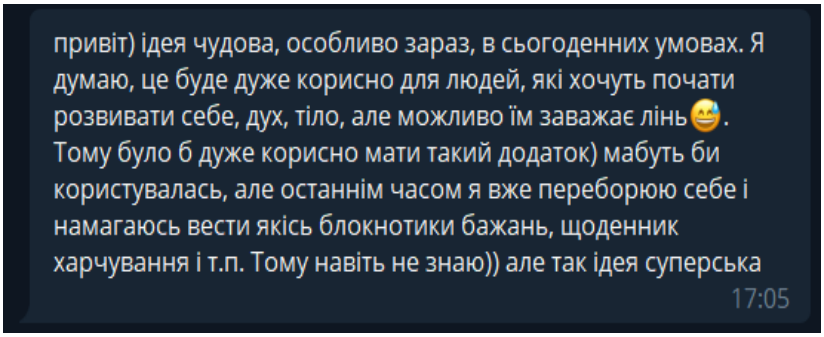

Рис. 5.2.2. Позитивний відгук стосовно ідеї проєкту.

- Чи зацікавили можливих користувачів якісь специфічні особливості вашого продукту, чи просили вони додати чи забрати щось?

Люди зацікавлені у більшості властивостях додатку й також їм імпонує їх поєднання, оскільки можуть зберігати усі цілі, звички, принципи та результати своєї праці в одному місці.

Щодо видалення певних можливостей застосунку, то деякі не зовсім зацікавлені в консультації з професіоналом для складання особистого плану розвитку. Тому було вирішено надати можливість спілкуватись з асистентом по саморозвитку, який розроблений на основі штучного інтелекту.

Щодо додання особливостей додатку казали, що вказаних можливостей є цілком достатньо й все чудово продумано.

- Чи можете розраховувати на спілкування з ними у майбутньому з приводу продукту?

Так, більшість користувачів радо відповіли на ідею проекту й хочуть спробувати використати описані можливості.

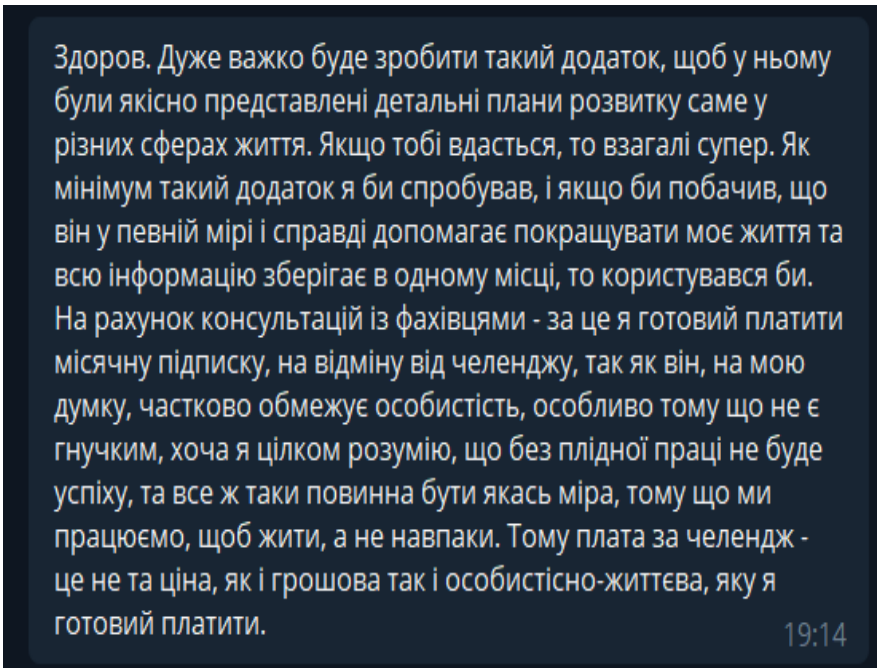

Рис. 5.2.3. Відгук про бажання використання проєкту

- Який досвід в спілкуванні з спеціалістами вибраної предметної області стосовно додатку?

В результаті спілкування з психологом було визначено, що варто саме не створювати свою ідеологію й навіювати її користувачам, а допомогти їм визначити свою. Саме під це підлаштовується поточний застосунок.

Було помічено, що більшість таких спеціалістів не працюють лише заради грошей. Це й логічно, оскільки це є найнижчий рівень мотивації і хто, як не вони про це знають. Психологи також допомогли мені сконцентруватись над унікальністю додатку. Підказали кілька виразів й порад, які будуть надаватись користувачу.

Оскільки з ними вдало було побудовано комунікацію, було отримано змогу одержувати консультації на теми психології.

5.3. Розроблення ринкової стратегії проєкту

Спершу для проєкту "Атомні звички" важливо правильно визначити цільові групи споживачів, які мають найбільший потенціал використання додатку та зацікавлені у особистісному розвитку.

1) Молодь та студенти.
    1.1) Особи віком від 18 до 30 років, що активно працюють над освітою та кар'єрним ростом.
    1.2) Зацікавлені в покращенні робочої ефективності та особистісному розвитку.
2) Професіонали та бізнес-користувачі.
    2.1) Працівники бізнесу, фрілансери, та підприємці (зазвичай віком від 25 до 45 років).
    2.2) Зацікавлені у збалансованому особистісному та професійному розвитку.
3) Люди, що активно працюють над саморозвитком.
    3.1) Особи будь-якого віку, які приділяють увагу здоров'ю, освіті, зміні своїх звичок та розвитку особистості.
    3.2) Зацікавлені в системному підході до створення та зберігання корисних звичок.

Розподілення на цільові групи дозволить максимально задовольнити потреби споживачів у кожному сегменті та забезпечити ефективну рекламу. Враховуючи різноманітність груп, варто використовувати стратегію диференційованого маркетингу.

Також було вибрано відповідні стратегічні напрями.

1) Створення персоналізованих пропозицій.
    - Адаптація функціоналу додатку під конкретні потреби кожної цільової групи.
    - Унікальні можливості для кожного сегменту (навчальні, професійні, бізнесові).
2) Маркетингові кампанії, спрямовані на специфіку сегменту.
    - Спеціальні рекламні акції для кожного сегменту (наприклад, акцент на академічних успіхах для студентів та молоді).
    - Партнерства та співпраця з організаціями, що спеціалізуються на конкретному сегменті.
3) Співпраця з психологами та експертами в галузі особистісного розвитку.
    - Впровадження рекомендацій та практик від професійних психологів для кожного сегменту.
    - Взаємодія з експертами для розробки унікальних програм.
4) Використання соціальних мереж та спільнот.
    - Створення спеціалізованих спільнот для обміну досвідом та мотивацією в межах кожного сегменту.
    - Залучення впливових особистостей в сфері особистісного розвитку для реклами та рекомендацій.
    - Активна присутність в соціальних мережах та спільнотах, спрямованих на саморозвиток, особистісний розвиток та продуктивність.
5) Запровадження акцій та знижок для привертання нових користувачів.

"Атомні звички" ставлять за мету створення унікального простору для особистісного розвитку, а стратегія диференційованого маркетингу дозволяє ефективно охоплювати та задовольняти потреби різноманітних груп користувачів. Посильною метою є збільшення інформації про додаток та його популярності серед широкого кола аудиторії, сприяючи тим самим їхньому розвитку особистості та досягненню успіху в різних сферах життя.

5.4. Маркетингова програма стартап-проєкту

Маркетингова програма для "Атомних Звичок" базується на глибокому розумінні потреб та цінностей цільових груп користувачів у сфері особистісного розвитку. Заснована на інноваційному підході до формування звичок та індивідуалізації досвіду користувачів, маркетингова стратегія враховує конкурентні переваги проєкту та потреби сучасного ринку.

1) Концепція товару

"Атомні Звички" - це інтегрована мультиплатформна система, що допомагає користувачам створювати та утримувати позитивні звички шляхом індивідуалізованого підходу та ефективного використання даних. Додаток пропонує стратегічне обмеження кількості нових звичок, персоналізовані підходи до створення звичок та підтримку від помічника, реалізованого на основі штучного інтелекту.

2) Збут

Продаж та розповсюдження "Атомних Звичок" здійснюватиметься через офіційні магазини додатків Google Play та App Store. Також передбачається співпраця зі спеціалізованими веб-платформами та партнерами в галузі особистісного розвитку.

3) Просування

Маркетингова кампанія буде орієнтована на підвищення усвідомленості та визнання "Атомних Звичок" як ключового інструменту для особистісного росту. Включатиме соціальні мережі, онлайн-рекламу, партнерства з блогерами та експертами, а також участь у конференціях та подіях зі сфери особистісного розвитку.

4) Ціноутворення

Модель ціноутворення може бути базована на фріміум-підході, де базовий функціонал є безкоштовним, а розширені функції доступні за плату. При цьому можлива індивідуалізація цін для різних сегментів користувачів. Спеціальні пакети та підписки можуть включати розширені можливості, такі як прискорена швидкість генерації рекомендованих звичок, розширені аналітичні звіти, нагадування виконання звичок та інші додаткові функції.

5) Попередній аналіз можливостей

Аналіз ринку особистісного розвитку підтверджує зростання попиту на інструменти для ефективного управління звичками та особистісним розвитком. Визначено відсутність подібних інновацій, що робить "Атомні Звички" унікальними на ринку. Аналіз ринкового середовища показує зростання попиту на засоби саморозвитку та планування. Швидке темпове життя, великий обсяг інформації та необхідність управління часом створюють сприятливі умови для успіху "Атомних Звичок".

6) Обрана альтернатива ринкової поведінки

Стратегія диференційованого маркетингу обрана для підходу до кількох сегментів ринку з індивідуалізованими програмами, що враховують специфіку потреб кожного сегменту.

Маркетингова програма "Атомних Звичок" спрямована на створення цінності для користувачів, визначення продукту як невід'ємної частини їхнього особистісного розвитку та надання конкурентних переваг на ринку.

5.5. Висновки до розділу

"Атомні Звички" виявляють потенціал для успішної ринкової комерціалізації на основі проведеного аналізу. Ринок особистісного розвитку динамічний та вимагає інноваційних рішень, що робить проєкт актуальним і конкурентоспроможним.

Попит на подібні додатки у сфері саморозвитку несе потенціал для значного зростання, оскільки сучасні люди все більше цікавляться власним

розвитком. Динаміка ринку свідчить про те, що користувачі шукають ефективні інструменти для створення та утримання позитивних звичок.

Розглядаючи потенційні групи клієнтів (студенти, професіонали, підприємці, вмотивовані та спортивні групи), "Атомні Звички" мають широкий спектр охоплення ринку, оскільки вони пропонують індивідуалізовані підходи для кожного сегменту. Це забезпечує залучення різних категорій користувачів.

Конкурентоспроможність проєкту підтверджується відмінностями від існуючих аналогів: стратегічне обмеження кількості нових звичок, персоналізована стратегія для кожного користувача та вбудований помічник по саморозвитку.

Успішність ринкової комерціалізації також зумовлена диференційованою маркетинговою стратегією, яка надає можливість адаптуватися до потреб різних сегментів. Модель ціноутворення фріміум може бути ефективною, забезпечуючи безкоштовний базовий функціонал та опцію платного розширення для зацікавлених користувачів.

З огляду на високий попит, конкурентоспроможність та відсутність подібних інновацій на ринку, "Атомні Звички" мають значний потенціал для успішного впровадження та рентабельності на ринку особистісного розвитку. Доцільність подальшого розвитку проєкту підтверджується потребами сучасного суспільства у зручних та ефективних інструментах для саморозвитку.

ВИСНОВКИ

В результаті виконання дипломної роботи були виконані наступні завдання.

1) Проаналізовано стан предметної області, тобто сферу особистісного розвитку;
2) досліджено ринок з додатками за даною тематикою;
3) перевірено різні програми розвитку;
4) проведено спілкування з клієнтами та психологом, що допомогло визначити вимоги до проєкту та його принципи функціонування;
5) визначено глосарій додатку для кращого розуміння його термінів;
6) побудовано діаграми варіантів використання, що пришвидшило розробку додатку за рахунок специфікації більш конкретних вимог;
7) описано схему сутностей;
8) визначено та автоматизовано деякі правила написання коду;
9) створено БД за допомогою EF Core та підходу Code First;
10) розміщено сервер на платформі Azure;
11) розроблено авторизацію, автентифікацію;
12) додано зручний відслідковувач звичок користувача;
13) реалізовано мультиплатформний інтерфейс програми засобами .NET MAUI та DevExpress;
14) розроблено генерацію рекомендованих звичок штучним інтелектом відповідно до вказаних її сфер, місії користувача (ціль на життя), статі та основного гасла;
15) реалізовано асистента по саморозвитку на основі штучного інтелекту, який може відповідати на різноманітні питання користувача.

Також були отримані наступні важливі відомості під час виконання курсового проєкту.

а) Людина – це не механізм: якщо просто вказати, що можна змінити – це рідко виправить ситуацію. Тому важливо не навіювати їй свою ідеологію, а навпаки лише допомагати створити свою;

б) дані користувача краще розміщувати на локальній БД з можливістю синхронізації на сервер;
в) сфера особистісного розвитку є досить актуальною, оскільки ми природньо хочемо розвиватись;
г) архітектура додатку є унікальною, універсальних алгоритмів немає в її розробці;
д) узагальненість в розробці ПЗ часто є обманливою, тому слід коректно користуватись принципами наслідування, поліморфізму та інкапсуляції задля прискореної реалізації додатку;
е) бібліотека DevExpress.Maui зручна в експлуатації. Містить також окремий сайт для швидкої допомоги у її використанні [44].

Отже, я зумів досягнути усіх цілей, які були поставлені на дипломну роботу, та в процесі цього отримати цінний досвід та знання, які зможу використати в подальшій кар’єрі та в покращенні даного застосунку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сватенко І. 10 найкращих додатків 2022 року за версією Google [Електронний ресурс] / І. Сватенко // Laba. – 2022. Режим доступу : https://laba.ua/blog/3311-10-luchshih-prilozheniy-2022-goda-po-versii-google.
2. MediaSapiens. У Google Play – у 3 рази більше завантажень додатків, ніж в App Store. При цьому Apple заробила в 2 рази більше [Електронний ресурс] / MediaSapiens // MediaSapiens. – 2020. Режим доступу : https://ms.detector.media/it-kompanii/post/25734/2020-10-15-u-google-play-u-3-razy-bilshe-zavantazhen-dodatkiv-nizh-v-app-store-pry-tsomu-apple-zarobyla-v-2-razy-bilshe/.
3. Kromme C. Unlocking the Potential of AI for personal development. [Electronic resource] / C. Kromme // LinkedIn. – 2022. Available from : https://www.linkedin.com/pulse/unlocking-potential-ai-personal-development-christian-kromme/.
4. BetterHelp Editorial Team. The Visualization Definition And How It Can Transform Your Life [Electronic resource] : - Available from : https://www.betterhelp.com/advice/visualization/the-visualization-definition-and-how-it-transforms-your-life/.
5. Найдіть свіжі ідеї [Електронний ресурс] : - Режим доступу : https://www.pinterest.ca/.
6. П’ять додатків з Play Market для саморозвитку: досвід Сватове.City [Електронний ресурс] : - Режим доступу : https://svatove.city/articles/88991/dlya-novogo-nikoli-ne-pizno-pyat-dodatkiv-z-play-market-dlya-vashogo-samorozvitku.
7. Саморозвиток [Електронний ресурс] : - Режим доступу : https://play.google.com/store/search?q=саморозвиток&c=apps&hl=uk&gl=US.
8. Архітектура програмного забезпечення [Електронний ресурс] : - Режим доступу : https://uk.wikipedia.org/wiki/Архітектура_програмного_забезпечення.
9. Несанкціонований доступ до інформації і його мета [Електронний ресурс] : - Режим доступу : https://studfile.net/preview/7435414/page:11/.

10. Архітектура клієнт-сервер [Електронний ресурс] : - Режим доступу : http://inter.ptngu.com/kompyuterni-merezhi/arhitektura-kliyent-server.
11. Tour of C# [Electronic resource] : - Available from : https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/.
12. What is .NET [Electronic resource] : - Available from : https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet.
13. Visual Studio 2022 Preview [Electronic resource] : - Available from : https://visualstudio.microsoft.com/vs/preview/.
14. What is .NET MAUI? [Electronic resource] : - Available from : https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui.
15. XAML [Electronic resource] : - Available from : https://learn.microsoft.com/en-us/dotnet/maui/xaml/.
16. Оцініть переваги мобільних додатків [Електронний ресурс] : - Режим доступу : https://smile-ukraine.com/ua/mobile-apps/mobile-apps-types.
17. Названо найдорожчі бренди світу. У першій трійці без сюрпризів, Tesla піднялася на 26 рядків https://biz.nv.ua/ukr/markets/reyting-brendiv-naydorozhchiy-u-apple-naybilshe-podorozhchala-tesla-poselednie-novini-50190597.html.
18. Is Xamarin dead ? [Electronic resource] : - Available from : https://www.manchesterdigital.com/post/foresight-mobile/is-xamarin-dead.
19. Handlers [Electronic resource] : - Available from : https://learn.microsoft.com/en-us/dotnet/maui/user-interface/handlers/.
20. 5 Advantages of .NET MAUI Over Xamarin [Electronic resource] : - Available from : https://www.syncfusion.com/blogs/post/advantages-net-maui-over-xamarin.aspx#unification%20of%20libraries.
21. Demystifying .NET Core and .NET Standard [Electronic resource] : - Available from : https://learn.microsoft.com/en-us/archive/msdn-magazine/2017/september/net-standard-demystifying-net-core-and-net-standard.
22. .NET Core Overview [Electronic resource] : - Available from : https://www.tutorialsteacher.com/core/dotnet-core.

23. ASP.NET Core Overview [Electronic resource] : - Available from : https://www.tutorialsteacher.com/core/aspnet-core-introduction.
24. C# Versions [Electronic resource] : - Available from : https://www.c-sharpcorner.com/article/c-sharp-versions/.
25. An introduction to NuGet [Electronic resource] : - Available from : https://learn.microsoft.com/en-us/nuget/what-is-nuget.
26. What is SQL Server Management Studio (SSMS)? [Electronic resource] : - Available from : https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16.
27. MySQL Workbench [Electronic resource] : - Available from : https://www.mysql.com/products/workbench/.
28. Business Logic: Definition, Benefits, and Example [Electronic resource] : - Available from : https://www.investopedia.com/terms/b/businesslogic.asp.
29. The Repository Design Pattern in C# [Electronic resource] : - Available from : https://www.codeguru.com/csharp/repository-pattern-c-sharp/.
30. Entity Framework Core [Electronic resource] : - Available from : https://learn.microsoft.com/en-us/ef/core/.
31. ASP.NET Core 5.0 Web API [Electronic resource] : - Available from : https://www.c-sharpcorner.com/article/asp-net-core-5-0-web-api/.
32. Docker overview [Electronic resource] : - Available from : https://docs.docker.com/get-started/overview/.
33. What is JSON [Electronic resource] : - Available from : https://www.w3schools.com/whatis/whatis_json.asp.
34. StarUML A sophisticated software modeler for agile and concise modeling [Electronic resource] : - Available from : https://staruml.io/.
35. What is Class Diagram [Electronic resource] : - Available from : https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/.
36. Unit Test using xUnit In .NET Core 6 [Electronic resource] : - Available from : https://www.c-sharpcorner.com/blogs/unit-test-using-xunit-in-net-core-6.

37. ЗАСТОСУВАННЯ UML В ДИПЛОМНИХ РОБОТАХ [Електронний ресурс] : - Режим доступу : https://dut.edu.ua/ua/news-1-626-7758-zastosuvannya-uml-v-diplomnih-robotah_kafedra-kompyuternih-nauk-ta-informaciynih-tehnologiy.
38. Kestrel web server in ASP.NET Core [Electronic resource] : - Available from : https://learn.microsoft.com/en-us/aspnet/core/fundamentals/servers/kestrel?view=aspnetcore-5.0.
39. Internet Information Services (IIS) [Electronic resource] : - Available from : https://www.techtarget.com/searchwindowsserver/definition/IIS.
40. GUID (global unique identifier) [Electronic resource] : - Available from : https://www.techtarget.com/searchwindowsserver/definition/GUID-global-unique-identifier.
41. 60 Секунд, Які Змінять Твоє Життя [Електронний ресурс] : - Режим доступу : https://www.youtube.com/watch?v=MbvOxFOSpTg.
42. Website Glossary [Electronic resource] : - Available from : https://blog.hubspot.com/website/glossary?hubs_content=blog.hubspot.com%2Fwebsite%2Fwhat-is-swagger&hubs_content-cta=Swagger.
43. Рейтинг топ веб-сайтів [Електронний ресурс] : - Режим доступу : https://www.similarweb.com/top-websites/.
44. .NET MAUI TreeView [Electronic resource] : - Available from : https://supportcenter.devexpress.com/ticket/details/t1168576/net-maui-treeview.
45. Cleary St. Concurrency in C# Cookbook: Asynchronous, Parallel, and Multithreaded Programming / Cleary St. 2nd ed. : Publication O'Reilly Media, 2019. – 254 p.
46. Конвенції в Mentor SET [Електронний ресурс] : - Режим доступу : https://docs.google.com/document/d/1glA6m-ax8Be7p0jDsJduGVXWldCBCJQI/edit?usp=sharing&ouid=110006529788053704421&rtpof=true&sd=true.
47. Stonis M. Enterprise Application Patterns using .NET MAUI / Stonis M. 1st ed.: Published by Microsoft Developer Division, .NET, and Visual Studio product teams, 2022 – 101p.

48. Секрети успішного життя [Електронний ресурс] : - Режим доступу : https://play.google.com/store/apps/details?id=com.inspiriongames.lifesecrets.lifesecrets&hl=uk&gl=US.
49. Трекер звичок Loop [Електронний ресурс] : - Режим доступу : https://play.google.com/store/apps/details?id=org.isoron.uhabits&hl=uk&gl=US.
50. Clear J. Atomic habits / Clear J. 1st ed.: Published by Penguin Random House, New York, 2018. – 320 p.
51. Bargh J. A. The four horsemen of automaticity: awareness, intention, efficiency, and control in social cognition. J. A. Bargh // Handbook of social cognition: Vol 1 basic processes. – 1994. – PP. 1–40.
52. Lally P. How are habits formed: Modelling habit formation in the real world. [Electronic resource] / P. Lally, C. H. M. Van Jaarsveld, H. W. W. Potts, J. Wardle // Wiley Online Library. – 2009. – Available From : https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.695.830&rep=rep1&type=pdf.
53. Xavier A. S. Loop Habit Tracker. [Electronic resource] / A. S. Xavier. - Available From : https://github.com/iSoron/uhabits.
54. Андрєєв А. GPT: що це таке, способи застосування, розвиток [Електронний ресурс] / Андреєєв А. : - Режим доступу : https://apix-drive.com/ua/blog/useful/gpt-sho-ce-take#sho-take-gpt-i-jak-ce-pracjue.
55. Azure Services [Electronic resource] : - Available from : https://portal.azure.com/#home.
56. Венцковська Ю. Словничок перекладача: локалізація додатків [Електронний ресурс] / Венцковська Ю. : - Режим доступу : https://mk-translations.ua/ua/blog/slovnichok-perekladacha-lokalizaciya-dodatkiv/.
57. Lightweight, scalable animations for your websites and apps [Electronic resource] : - Available from : https://lottiefiles.com/.

ДОДАТКИ

Файл: CircularProgressBarDrawable.cs

```
using Microsoft.Maui.Graphics.Text;

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SET.MAUI.Drawables
{
   public class CircularProgressBarDrawable : BindableObject, IDrawable
   {
      public static readonly BindableProperty ProgressProperty = BindableProperty.Create( nameof( Progress ), typeof( int ), typeof( CircularProgressBarDrawable ), defaultBindingMode: BindingMode.TwoWay );
      public static readonly BindableProperty SizeProperty = BindableProperty.Create( nameof( Size ), typeof( int ), typeof( CircularProgressBarDrawable ) );
      public static readonly BindableProperty ThicknessProperty = BindableProperty.Create( nameof( Thickness ), typeof( float ), typeof( CircularProgressBarDrawable ) );
      public static readonly BindableProperty ProgressColorProperty = BindableProperty.Create( nameof( ProgressColor ), typeof( Color ), typeof( CircularProgressBarDrawable ) );
      public static readonly BindableProperty ProgressLeftColorProperty = BindableProperty.Create( nameof( ProgressLeftColor ), typeof( Color ), typeof( CircularProgressBarDrawable ) );
      public static readonly BindableProperty TextColorProperty = BindableProperty.Create( nameof( TextColor ), typeof( Color ), typeof( CircularProgressBarDrawable ) );

      public int Progress
      {
         get => (int)GetValue( ProgressProperty );
         set => SetValue( ProgressProperty, value );
      }

      public int Size
      {
         get { return (int)GetValue( SizeProperty ); }
         set { SetValue( SizeProperty, value ); }
      }

      public float Thickness
      {
         get { return (float)GetValue( ThicknessProperty ); }
         set { SetValue( ThicknessProperty, value ); }
      }

      public Color ProgressColor
      {
         get { return (Color)GetValue( ProgressColorProperty ); }
         set { SetValue( ProgressColorProperty, value ); }
      }

      public Color ProgressLeftColor
      {
         get { return (Color)GetValue( ProgressLeftColorProperty ); }
         set { SetValue( ProgressLeftColorProperty, value ); }
      }

      public Color TextColor
      {
         get { return (Color)GetValue( TextColorProperty ); }
         set { SetValue( TextColorProperty, value ); }
      }
```

```
    public void Draw( ICanvas canvas, RectF dirtyRect )
    {
        float effectiveSize = Size - Thickness;
        float x = Thickness / 2;
        float y = Thickness / 2;

        if (Progress < 0)
        {
            Progress = 0;
        }
        else if (Progress > 100)
        {
            Progress = 100;
        }

        if (Progress < 100)
        {
            float angle = GetAngle( Progress );

            canvas.StrokeColor = ProgressLeftColor;
            canvas.StrokeSize = Thickness;
            canvas.DrawEllipse( x, y, effectiveSize, effectiveSize );

            // Draw arc
            canvas.StrokeColor = ProgressColor;
            canvas.StrokeSize = Thickness;
            canvas.DrawArc( x, y, effectiveSize, effectiveSize, 90, angle, true, false );
        }
        else
        {
            // Draw circle
            canvas.StrokeColor = ProgressColor;
            canvas.StrokeSize = Thickness;
            canvas.DrawEllipse( x, y, effectiveSize, effectiveSize );
        }

        // Make the percentage always the same size in relation to the size of the progress bar
        float fontSize = effectiveSize / 2.86f;
        canvas.FontSize = fontSize;
        canvas.FontColor = TextColor;
        canvas.FillColor = TextColor;

        // Vertical text align the text, and we need a correction factor of around 1.15 to have it aligned properly
        // Note: The VerticalAlignment.Center property of the DrawString method seems to have no effect
        float verticalPosition = ((Size / 2) - (fontSize / 2)) * 0.25f;
        string progress = $"{Progress}%";
        canvas.DrawString( progress, x, verticalPosition, effectiveSize, effectiveSize, HorizontalAlignment.Center,
VerticalAlignment.Center );
    }

    private float GetAngle( int progress )
    {
        float factor = 90f / 25f;

        if (progress > 75)
        {
            return -180 - ((progress - 75) * factor);
        }
        else if (progress > 50)
        {
            return -90 - ((progress - 50) * factor);
        }
        else if (progress > 25)
        {
```

```
            return 0 - ((progress - 25) * factor);
        }
        else
        {
            return 90 - (progress * factor);
        }
    }
  }
}
```

Файл: EditHabitView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<v:ContentPageBase xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
                   xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
                   x:Class="SET.MAUI.Views.EditHabitView"
                   xmlns:loc="clr-namespace:SET.MAUI.Resources.AppStrings"
                   xmlns:dxco="clr-namespace:DevExpress.Maui.Controls;assembly=DevExpress.Maui.Controls"
                   xmlns:dx="clr-namespace:DevExpress.Maui.Core;assembly=DevExpress.Maui.Core"
                   xmlns:dxe="clr-namespace:DevExpress.Maui.Editors;assembly=DevExpress.Maui.Editors"
                   xmlns:dxcv="clr-
namespace:DevExpress.Maui.CollectionView;assembly=DevExpress.Maui.CollectionView"
                   xmlns:skia="clr-namespace:SkiaSharp.Extended.UI.Controls;assembly=SkiaSharp.Extended.UI"
                   xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"
                   xmlns:v="clr-namespace:SET.MAUI.Views"
                   xmlns:vm="clr-namespace:SET.MAUI.ViewModels"
                   xmlns:m="clr-namespace:SET.Core.Models;assembly=SET.Core"
                   xmlns:templates="clr-namespace:SET.MAUI.Views.Templates"
                   xmlns:controls="clr-namespace:SET.MAUI.Controls"
                   x:DataType="vm:EditHabitViewModel"
                   x:Name="CPB_EditHabit"
                   Shell.TabBarIsVisible="False">
    <Shell.BackButtonBehavior>
        <BackButtonBehavior Command="{Binding BackCommand}" />
    </Shell.BackButtonBehavior>

    <Shell.TitleView>
        <Grid ColumnDefinitions="Auto, *, Auto"
              x:Name="G_Title">
            <Label Text="{Binding NameOfHabit.Value}"
                   x:Name="L_TitleText"
                   VerticalOptions="Center"
                   TextColor="{StaticResource TitleTextColor}"
                   FontAttributes="Bold"
                   LineBreakMode="TailTruncation"
                   FontSize="17"
                   Grid.Column="0"/>
            <Grid x:Name="G_SaveHabit"
                      HorizontalOptions="End"
                      VerticalOptions="CenterAndExpand"
                      SizeChanged="G_SaveHabit_SizeChanged"
                      Grid.Column="2">
                <dxco:SimpleButton Text="{Binding LocSave, Converter={StaticResource ToUpperCase}}"
                           Style="{StaticResource TitlePrimaryButton}"
                           x:Name="SB_SaveHabit"
                           Clicked="SB_Save_Clicked"
                           Margin="0, 2, 0, 2"
                           FontSize="14"
                           BorderThickness="2"/>
            </Grid>
        </Grid>
    </Shell.TitleView>
```

```
    <dxco:TabView HeaderPanelPosition="Top"
                  x:Name="TB_Tabs"
                  SwipeEnabled="True"
                  BackgroundColor="White"
                  HeaderPanelBackgroundColor="White"
                  SelectedItemHeaderBackgroundColor="White"
                  Background="White">
        <dxco:TabViewItem HeaderText="{x:Static loc:LocStrings.Info}"
                          HeaderIconPosition="Right"
                          HeaderWidth="160">
            <VerticalStackLayout Padding="10"
                                 Spacing="6">
                <dxe:AutoCompleteTokenEdit LabelText="{x:Static loc:LocStrings.AreasOfHabit}"
                                           EndIcon="dotshorizontal"
                                           ItemsSource="{Binding AllUserAreasOfLife}"
                                           EndIconClicked="TE_AreasOfLife_EndIconClicked"
                                           StartIcon="rules"
                                           IsStartIconVisible="True"
                                           DisplayMember="Name"
                                           SelectedItems="{Binding Habit.AreasOfLife, Mode=TwoWay}"
                                           SelectionChanged="TE_AreasOfLife_SelectionChanged"
                                           DropDownSelectedItemBackgroundColor="{AppThemeBinding Light={StaticResource
LightPrimary}, Dark={StaticResource LightPrimary}}"
                                           x:Name="TE_AreasOfLife">
                    <dxe:AutoCompleteTokenEdit.TokenAppearance>
                        <dxe:TokenAppearance BorderColor="{StaticResource Primary}"
                                             BorderThickness="2"
                                             CornerRadius="14"
                                             FontSize="14"
                                             TextColor="{AppThemeBinding Light={StaticResource LightNormalText},
Dark={StaticResource DarkNormalText}}"
                                             PressedBorderColor="{StaticResource Primary}"/>
                    </dxe:AutoCompleteTokenEdit.TokenAppearance>
                    <dxe:AutoCompleteTokenEdit.GestureRecognizers>
                        <TapGestureRecognizer NumberOfTapsRequired="1"
                                              x:Name="TGR_AreasOfHabit"
                                              Tapped="TGR_AreasOfHabit_Tapped"/>
                    </dxe:AutoCompleteTokenEdit.GestureRecognizers>
                </dxe:AutoCompleteTokenEdit>

                <dxe:MultilineEdit Text="{Binding NameOfHabit.Value}"
                                   LabelText="{x:Static loc:LocStrings.Name}"
                                   HasError="{Binding NameOfHabit.IsValid, Converter={StaticResource IsFalse}}"
                                   IsErrorIconVisible="True"
                                   x:Name="ME_NameOfHabit"
                                   MaxLineCount="5"
                                   StartIcon="fire_first"
                                   EndIcon="ai"
                                   IsEndIconVisible="{Binding IsNewHabit}"
                                   EndIconClicked="ME_NameOfHabit_AiIconClicked"
                                   ClearIconVisibility="Never" />

                <dxe:ChoiceChipGroup x:Name="CCG_Types"
                                     HorizontalOptions="CenterAndExpand">
                    <dxe:Chip Text="{x:Static loc:LocStrings.WithoutExceptions}"
                              Icon="info"
                              x:Name="C_WithoutExceptionsType"
                              TapCommand="{Binding TapWithoutExceptionsTypeCommand,
                                                   Source={RelativeSource AncestorType={x:Type vm:EditHabitViewModel}}}"
                              TapCommandParameter="{x:Reference C_WithoutExceptionsType}">
                    </dxe:Chip>
                    <dxe:Chip Text="{x:Static loc:LocStrings.IntegrallyWise}"
                              Icon="info"
                              x:Name="C_IntegrallyWiseType"
                              TapCommand="{Binding TapIntegrallyWiseTypeCommand,
```

```
                        Source={RelativeSource AncestorType={x:Type vm:EditHabitViewModel}}}"
            TapCommandParameter="{x:Reference C_IntegrallyWiseType}"/>
</dxe:ChoiceChipGroup>

<dxe:TextEdit IsReadOnly="True"
            StartIcon="restart"
            StartIconClicked="FrequencyIcon_Clicked"
            EndIcon="dotshorizontal"
            EndIconClicked="FrequencyIcon_Clicked"
            Text="{Binding FrequencyRepresentation}"
            LabelText="{x:Static loc:LocStrings.Frequency}">
   <dxe:TextEdit.GestureRecognizers>
      <TapGestureRecognizer Tapped="Frequency_Tapped"/>
   </dxe:TextEdit.GestureRecognizers>
</dxe:TextEdit>
<dxe:NumericEdit Value="{Binding Habit.Complexity}"
              LabelText="{x:Static loc:LocStrings.ComplexityWithRange}"
              MinValue="1"
              MaxDecimalDigitCount="0"
              MaxValue="10"
              IsReadOnly="{Binding IsNewHabit, Converter={StaticResource IsFalse}}"
              StartIcon="charts"
              IsUpDownIconVisible="True" />

<dxco:DXPopup x:Name="DXP_Errors"
            AllowScrim="True"
            AllowAnimation="True"
            AllowShadow="True"
            CloseOnScrimTap="True"
            MinimumWidthRequest="230"
            HeightRequest="200">
   <dx:DXStackLayout Orientation="Vertical">
      <Label Text="{x:Static loc:LocStrings.FixErrorsFirst}"
             HorizontalOptions="Center"
             VerticalOptions="Center"
             Margin="10, 10, 10, 0"
             FontSize="Subtitle"
             FontAttributes="Bold"/>

      <dx:DXStackLayout Margin="15"
                     Orientation="Horizontal"
                         ItemSpacing="10"
                         x:Name="HSL_ErrorMessages">
         <dx:DXImage Source="info"
                     x:Name="DXI_Error"
                     WidthRequest="25"
                     HeightRequest="25"
                        TintColor="{StaticResource Error}"
                        VerticalOptions="Center"/>
         <dx:DXStackLayout Orientation="Vertical"
                        x:Name="DXS_ErrorMessages"
                        VerticalOptions="Center">


         </dx:DXStackLayout>
         <!--<Label x:Name="L_ErrorMessages"
               FontSize="12"
               MaxLines="2"
               LineBreakMode="TailTruncation"
                  VerticalOptions="Center"
                  VerticalTextAlignment="Center"/>-->
      </dx:DXStackLayout>

      <dxco:SimpleButton x:Name="SB_CloseErrorsPopup"
                     Clicked="SB_CloseErrorsPopup_Clicked"
```

```
                    Text="{x:Static loc:LocStrings.OK}"
                    FontSize="Medium"
                    HorizontalOptions="End"
                    MinimumWidthRequest="60"
                    Margin="0, 5, 5, 5"
                    Style="{StaticResource PrimaryButton}"/>
    </dx:DXStackLayout>
</dxco:DXPopup>
<dxco:DXPopup x:Name="DXP_Frequency"
            AllowScrim="True"
            AllowAnimation="True"
            MinimumWidthRequest="250"
            AllowShadow="True"
            CloseOnScrimTap="True">
    <VerticalStackLayout Margin="15">
        <RadioButton Content="{x:Static loc:LocStrings.EveryDay}"
                FontSize="16"
                x:Name="RB_EveryDay"
                    GroupName="Frequency"/>

        <HorizontalStackLayout Margin="0, 10, 0, 10"
                        Focused="RepeatsOfSeveralDays_Focused">
            <RadioButton x:Name="RB_EverySeveralDays"
                    Content="{x:Static loc:LocStrings.Every}"
                    FontSize="16"
                    GroupName="Frequency"/>
            <Entry Keyboard="Numeric"
                HorizontalTextAlignment="Center"
                Margin="0, 0, 0, 5"
                FontSize="16"
                x:Name="E_RepeatsOfSeveralDays"
                Focused="RepeatsOfSeveralDays_Focused"
                MaxLength="2" />
            <Label Text="{x:Static loc:LocStrings.days}"
                FontSize="16"
                Margin="7, 6, 0, 0">
              <Label.GestureRecognizers>
                <TapGestureRecognizer Tapped="EverySeveralDaysFrequency_Tapped" />
              </Label.GestureRecognizers>
            </Label>
        </HorizontalStackLayout>

        <HorizontalStackLayout>
            <HorizontalStackLayout.GestureRecognizers>
              <TapGestureRecognizer Tapped="SeveralTimesPerPeriodFrequency_Tapped"/>
            </HorizontalStackLayout.GestureRecognizers>
            <RadioButton x:Name="RB_SeveralTimesPerPeriod"
                    GroupName="Frequency"/>
            <Entry Keyboard="Numeric"
                Margin="0, 0, 0, 15"
                MaxLength="3"
                HorizontalTextAlignment="Center"
                FontSize="16"
                x:Name="E_RepeatsOfSeveralTimesPerPeriod"
                Focused="E_RepeatsOfSeveralTimesPerPeriod_Focused">
            </Entry>
            <Label Text="{x:Static loc:LocStrings.timesPer}"
                FontSize="16"
                Margin="7, 10, 0, 0"/>
            <dxe:ComboBoxEdit ItemsSource="{Binding PeriodsOfHabit}"
                            SelectedItem="{Binding SelectedPeriodOfHabit}"
                    DisplayMember="Name"
                    WidthRequest="100"
                    TextFontSize="16"
                    HeightRequest="37"
```

```
                        BoxPadding="10, 10, 0, 10"
                        IsDropDownIconVisible="False"
                        BorderThickness="1"
                        FocusedBorderThickness="1"
                        DropDownBackgroundColor="{AppThemeBinding Light={StaticResource Background},
Dark={StaticResource DarkBackground}}"
                        DropDownSelectedItemBackgroundColor="{StaticResource LightPrimary}"
                            x:Name="CBE_Period"
                    Margin="7, 0, 0, 15">
                  <dxe:ComboBoxEdit.GestureRecognizers>
                    <TapGestureRecognizer Tapped="SeveralTimesPerPeriodFrequency_Tapped"/>
                  </dxe:ComboBoxEdit.GestureRecognizers>
                  <dxe:ComboBoxEdit.BorderColor>
                    <Color>#e9e9e9</Color>
                  </dxe:ComboBoxEdit.BorderColor>
                  <dxe:ComboBoxEdit.FocusedBorderColor>
                    <Color>#e9e9e9</Color>
                  </dxe:ComboBoxEdit.FocusedBorderColor>
                </dxe:ComboBoxEdit>
              </HorizontalStackLayout>

              <dxco:SimpleButton x:Name="SB_SaveFrequencyPopup"
                        Clicked="SB_SaveFrequencyPopup_Clicked"
                        Text="{Binding LocSave}"
                        HorizontalOptions="End"
                        FontSize="16"
                        Margin="0, 10, 0, 0"
                        Style="{StaticResource PrimaryButton}"/>
            </VerticalStackLayout>
          </dxco:DXPopup>

          <dxco:BottomSheet x:Name="BS_RecommendedHabits"
                    BackgroundColor="{AppThemeBinding Light={StaticResource Background},
Dark={StaticResource DarkBackground}}"
              ShowGrabber="True"
              AllowDismiss="True"
                    HalfExpandedRatio="0.6">
            <Grid>
              <VerticalStackLayout Padding="0, 7, 0, 15"
                       Spacing="20"
                       VerticalOptions="CenterAndExpand">
                <Label Text="{x:Static loc:LocStrings.RecommendedHabitsByAi}"
                        Style="{StaticResource CentralHeader}"/>

                <dxcv:DXCollectionView ItemsSource="{Binding RecommendedHabits}"
                            x:Name="DXC_RecommendedHabits"
                            Margin="0, 0, 0, 0">
                  <dxcv:DXCollectionView.ItemTemplate>
                    <DataTemplate x:DataType="m:RecommendedHabit">
                      <Grid RowDefinitions="Auto, Auto"
                          ColumnDefinitions="140, *"
                          ColumnSpacing="15">
                        <Grid.GestureRecognizers>
                          <TapGestureRecognizer Tapped="TGR_RecommendedHabit_Tapped"
                                    x:Name="TGR_RecommendedHabit"
                                    CommandParameter="{Binding}"
                                    NumberOfTapsRequired="1"/>
                        </Grid.GestureRecognizers>

                        <Label Text="{Binding Name}"
                            Padding="10, 15, 0, 0"
                            FontAttributes="Bold"
                            VerticalOptions="End"
                            VerticalTextAlignment="End"
                            Grid.Row="0"
```

```
                        Grid.Column="0"/>
                    <Label Text="{Binding ReasonToFollow}"
                        ToolTipProperties.Text="{Binding ReasonToFollow}"
                        LineBreakMode="TailTruncation"
                        VerticalOptions="End"
                        VerticalTextAlignment="Center"
                        MaxLines="4"
                        Margin="0, 0, 5, 0"
                        Grid.Row="0"
                        Grid.Column="1"/>

                    <BoxView Grid.Row="1"
                         Style="{StaticResource DefaultHorizonalSeparator}"
                         Grid.Column="0"
                         Grid.ColumnSpan="2"/>
                </Grid>

            </DataTemplate>

        </dxcv:DXCollectionView.ItemTemplate>

    </dxcv:DXCollectionView>

</VerticalStackLayout>

<VerticalStackLayout HorizontalOptions="Center"
                     VerticalOptions="Center"
                    Spacing="10">
    <skia:SKLottieView IsVisible="{Binding IsBusy}"

                RepeatCount="-1"
                Source="loading.json"
                HorizontalOptions="Center"
                WidthRequest="130"
                HeightRequest="130"/>
    <Label Text="{x:Static loc:LocStrings.IfYouDontWantToWaitYouCanChangeOtherFields}"
                IsVisible="{Binding IsBusy}"
                HorizontalTextAlignment="Center"
                WidthRequest="270"
                HeightRequest="100"
                LineBreakMode="TailTruncation"
                MaxLines="4"/>
</VerticalStackLayout>


    </Grid>

  </dxco:BottomSheet>
 </VerticalStackLayout>
</dxco:TabViewItem>
<dxco:TabViewItem HeaderText="{x:Static loc:LocStrings.HowToKeep}"
            HeaderIconPosition="Right"
            HeaderWidth="160">
 <VerticalStackLayout Padding="10">
    <dxe:MultilineEdit Text="{Binding ReasonToFollow.Value}"
                x:Name="ME_ReasonToFollow"
                LabelText="{x:Static loc:LocStrings.ReasonToFollow}"
                HasError="{Binding ReasonToFollow.IsValid, Converter={StaticResource IsFalse}}"
                IsErrorIconVisible="False"
                StartIcon="approved"
                EndIcon="info"
                EndIconCommand="{Binding ShowSnackbarForReasonToFollowCommand}"
                EndIconCommandParameter="{x:Reference ME_ReasonToFollow}"
                ClearIconVisibility="Never" />
    <dxe:TextEdit Text="{Binding Habit.Question}"
```

```
                    x:Name="TE_Habit"
                    LabelText="{x:Static loc:LocStrings.Question}"
                    StartIcon="question"
                    EndIcon="info"
                    EndIconCommand="{Binding ShowSnackbarForQuestionCommand}"
                    EndIconCommandParameter="{x:Reference TE_Habit}"/>
            <dxe:MultilineEdit Text="{Binding Habit.Description}"
                       LabelText="{x:Static loc:LocStrings.NotesOrHowToKeepHabit}"
                       StartIcon="notes" />
        </VerticalStackLayout>

      </dxco:TabViewItem>
   </dxco:TabView>
</v:ContentPageBase>
```

Файл: HelperView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<v:ContentPageBase xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:loc="clr-namespace:SET.MAUI.Resources.AppStrings"
             xmlns:dxc="clr-namespace:DevExpress.Maui.Controls;assembly=DevExpress.Maui.Controls"
             xmlns:v="clr-namespace:SET.MAUI.Views"
             xmlns:viewmodels="clr-namespace:SET.MAUI.ViewModels"
             xmlns:collectionview="clr-
namespace:DevExpress.Maui.CollectionView;assembly=DevExpress.Maui.CollectionView"
             xmlns:skia="clr-namespace:SkiaSharp.Extended.UI.Controls;assembly=SkiaSharp.Extended.UI"
             xmlns:editors="clr-namespace:DevExpress.Maui.Editors;assembly=DevExpress.Maui.Editors"
             xmlns:maui="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"
             xmlns:templates="clr-namespace:SET.MAUI.Views.Templates"
             x:Class="SET.MAUI.Views.HelperView"
             x:DataType="viewmodels:HelperViewModel"
             Shell.NavBarIsVisible="False">
   <v:ContentPageBase.Resources>
      <Color x:Key="NavigationBarColorLight">White</Color>
      <Color x:Key="NavigationBarColorDark">#111111</Color>
      <Color x:Key="SurfaceColorLight">#FAFAFA</Color>
      <Color x:Key="SurfaceColorDark">#1D1E24</Color>
      <DataTemplate x:Key="HelperMessageTemplate">
         <templates:HelperMessageTemplate />
      </DataTemplate>
      <DataTemplate x:Key="UserMessageTemplate">
         <templates:UserMessageTemplate />
      </DataTemplate>

      <templates:MessageDataTemplateSelector x:Key="MessageDataTemplateSelector"
                                 HelperMessageTemplate="{StaticResource HelperMessageTemplate}"
                                 UserMessageTemplate="{StaticResource UserMessageTemplate}"/>
   </v:ContentPageBase.Resources>

   <Grid RowDefinitions="58, *, 16, Auto, 16"
        BackgroundColor="{StaticResource Primary}">
      <HorizontalStackLayout HorizontalOptions="Start"
                    Grid.Row="0"
                    Padding="16, 0"
                    BackgroundColor="{AppThemeBinding Light={StaticResource Primary},
                              Dark={StaticResource NavigationBarColorDark}}"
                    Spacing="10"
                    VerticalOptions="CenterAndExpand">
         <dxc:DXImage Source="helper"
                 TintColor="{StaticResource DarkIconColor}"
                 WidthRequest="35"
```

```
                HeightRequest="35" />
        <Label Text="{Binding Title}"
            TextColor="{StaticResource DarkIconColor}"
            VerticalOptions="Center"
            VerticalTextAlignment="Center"
            FontSize="20"
            FontFamily="MonaSansBold"
            FontAttributes="Bold" />
    </HorizontalStackLayout>

    <Border Grid.Row="1"
        Padding="0, 10, 0, 30"
        BackgroundColor="{AppThemeBinding Light={StaticResource SurfaceColorLight},
                                Dark={StaticResource SurfaceColorDark}}"
        StrokeShape="RoundRectangle 0, 0, 36, 36"
        StrokeThickness="0">
        <collectionview:DXCollectionView x:Name="DXC_DisplayMessages"
                                ItemsSource="{Binding DisplayMessages}"
                                ItemTemplate="{StaticResource MessageDataTemplateSelector}"
                                ItemSpacing="20"
                                Orientation="Vertical"
                                IsScrollBarVisible="True"
                                IsPullToRefreshEnabled="False"
                                IsLoadMoreEnabled="False"/>
    </Border>

    <Grid ColumnDefinitions="15*, 70*, 15*"
        RowDefinitions="20*, 40*, Auto, 40*"
        IsVisible="{Binding IsAnimationVisible}"
        Grid.Row="1">
        <skia:SKLottieView IsAnimationEnabled="{Binding IsAnimationVisible}"
                    RepeatCount="-1"
                    Source="{AppThemeBinding Light=emptychat_light.json, Dark=emptychat_dark.json}"
                    Grid.Row="1"
                    Grid.Column="1" />
        <Label Text="{x:Static loc:LocStrings.SelfDevelopmentAssistantShortDescription}"
            FontFamily="MonaSansMedium"
            FontSize="14"
            TextColor="{AppThemeBinding Light={StaticResource LightNormalText}, Dark={StaticResource
DarkNormalText}}"
            HorizontalTextAlignment="Center"
            Padding="15, 0, 15, 0"
            Grid.Row="2"
            Grid.ColumnSpan="3"/>
    </Grid>

    <Grid ColumnDefinitions="*, Auto"
        ColumnSpacing="3"
        Padding="16, 0"
        Grid.Row="3">
        <Border BackgroundColor="{AppThemeBinding Light={StaticResource LightPrimary}, Dark={StaticResource
DarkPrimary}}"
            HeightRequest="48"
            StrokeShape="RoundRectangle 36,36,36,36"
            Grid.Column="0">
            <Grid ColumnDefinitions="*, Auto">
                <editors:TextEdit PlaceholderText="{x:Static loc:LocStrings.EnterText}"
                            x:Name="TE_Prompt"
                            Completed="TE_Prompt_Completed"
                            Text="{Binding Prompt}"
                            TextFontFamily="MonaSansMedium"
                            TextFontSize="14"
                            BorderThickness="0"
                            HeightRequest="48"
                            FocusedBorderThickness="0"
```

```
                        TextColor="{AppThemeBinding Light={StaticResource LightNormalText},
Dark={StaticResource DarkNormalText}}"
                        PlaceholderColor="{AppThemeBinding Light={StaticResource SecondaryLight},
Dark={StaticResource SecondaryDark}}"
                        BackgroundColor="{AppThemeBinding Light={StaticResource LightPrimary},
Dark={StaticResource DarkPrimary}}"
                        TranslationY="4"
                        ClearIconVisibility="Never"
                        Margin="16, 0, 16, 5"
                        RotationX="20"/>

            <skia:SKLottieView IsVisible="{Binding IsBusy}"
                        RepeatCount="-1"
                        Source="loading.json"
                        HorizontalOptions="Center"
                        WidthRequest="48"
                        HeightRequest="48"
                        Grid.Column="1"/>
          </Grid>
        </Border>

        <dxc:SimpleButton Icon="send_message"
                    x:Name="SB_AskQuestion"
                    HeightRequest="48"
                    WidthRequest="48"
                    IsVisible="{Binding IsBusy, Converter={StaticResource IsFalse}}"
                    Clicked="SB_AskQuestion_Clicked"
                    Style="{StaticResource PrimaryButton}"
                    BackgroundColor="{StaticResource LightPrimary}"
                    IconColor="{AppThemeBinding Light={StaticResource SecondaryLight}, Dark={StaticResource
SecondaryDark}}"
                    Grid.Column="1" />

        <dxc:DXImage Source="stop_video"
                HeightRequest="35"
                WidthRequest="35"
                Margin="7"
                VerticalOptions="CenterAndExpand"
                IsVisible="{Binding IsBusy}"
                TintColor="{AppThemeBinding Light={StaticResource QuinaryLight}, Dark={StaticResource
QuinaryDark}}"
                Grid.Column="1">
          <dxc:DXImage.GestureRecognizers>
            <TapGestureRecognizer Command="{Binding CancelAnswerGenerationCommand}"/>
          </dxc:DXImage.GestureRecognizers>
        </dxc:DXImage>
      </Grid>
    </Grid>
</v:ContentPageBase>
```

Файл: LoginView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<v:ContentPageBase xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            xmlns:dxe="clr-namespace:DevExpress.Maui.Editors;assembly=DevExpress.Maui.Editors"
            xmlns:dxco="clr-namespace:DevExpress.Maui.Controls;assembly=DevExpress.Maui.Controls"
            xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"
            xmlns:loc="clr-namespace:SET.MAUI.Resources.AppStrings"
            xmlns:vm="clr-namespace:SET.MAUI.ViewModels"
            xmlns:v="clr-namespace:SET.MAUI.Views"
            xmlns:templates="clr-namespace:SET.MAUI.Views.Templates"
            x:Class="SET.MAUI.Views.LoginView"
```

```
            x:DataType="vm:LoginViewModel"
            Shell.FlyoutBehavior="Disabled"
            Shell.NavBarIsVisible="False"
            Shell.TabBarIsVisible="False"
            Title="{x:Static loc:LocStrings.Login}">
    <Grid>
        <VerticalStackLayout Padding="15"
                             Spacing="20"
                             VerticalOptions="CenterAndExpand">
            <Image Source="logolargesize"
                   WidthRequest="200"
                   HeightRequest="200"/>
            <Label Text="{Binding AppName, Converter={StaticResource ToUpperCase}}"
                TextColor="OrangeRed"
                HorizontalOptions="Center"
                FontSize="Title"
                FontFamily="Roboto"
                FontAttributes="Bold"/>
            <dxe:TextEdit Text="{Binding Email.Value}"
                          LabelText="{x:Static loc:LocStrings.Email}"
                          AutofillContentType="EmailAddress"
                          HelpText="{x:Static loc:LocStrings.Correct}"
                          StartIcon="email"
                          Keyboard="Email"
                          ClearIconVisibility="Never"
                          HasError="{Binding Email.IsValid,
                                     Converter={StaticResource IsFalse}}"
                          ErrorText="{Binding Email.Errors,
                                      Converter={StaticResource FirstValidationError}}"
                          TextChangedCommand="{Binding ValidateEmailCommand}" />


            <dxe:PasswordEdit Text="{Binding Password.Value}"
                              x:Name="PasswordEntry"
                              LabelText="{x:Static loc:LocStrings.Password}"
                              StartIcon="password"
                              AutofillContentType="Password"
                              HelpText="{x:Static loc:LocStrings.Correct}"
                              HasError="{Binding Password.IsValid, Converter={StaticResource IsFalse}}"
                              ErrorText="{Binding Password.Errors,
                                          Converter={StaticResource FirstValidationError}}"
                              TextChangedCommand="{Binding ValidatePasswordCommand}" />

            <dxco:SimpleButton Text="{x:Static loc:LocStrings.Login}"
                               Command="{Binding LoginCommand}"
                               Style="{StaticResource PrimaryButton}" />

            <dxco:SimpleButton Text="{x:Static loc:LocStrings.SignUp}"
                               Command="{Binding SignupCommand}"
                               BackgroundColor="{AppThemeBinding Light={StaticResource DarkPrimary},
Dark={StaticResource Primary}}"
                               TextColor="{AppThemeBinding Light={StaticResource Primary}, Dark={StaticResource
DarkPrimary}}" />
        </VerticalStackLayout>

        <ActivityIndicator Color="{StaticResource Primary}"
                           IsRunning="{Binding IsBusy}"
                           IsVisible="{Binding IsBusy}"
                           VerticalOptions="Center"
                           HorizontalOptions="Center"
                           WidthRequest="100" />
    </Grid>
</v:ContentPageBase>
```

Файл: MauiNavigationService.cs

```
using Microsoft.Extensions.Configuration;

using SET.MAUI.ViewModels;
using SET.MAUI.Views;

using System.Globalization;
using System.Reflection;
using System.Web;

namespace SET.Core.Services;

public class MauiNavigationService : INavigationService
{
   private readonly IConfiguration m_config;
   private readonly ISettingsService m_settingsService;

   public MauiNavigationService(IConfiguration config, IUrlBuilder urlBuilder, ISettingsService settingsService)
   {
      m_config = config;
      UrlBuilder = urlBuilder;
      m_settingsService = settingsService;
   }

   public bool IsLoggedIn => !string.IsNullOrWhiteSpace( m_settingsService.AuthAccessToken );

   public IUrlBuilder UrlBuilder { get; }

   public Task GoToInitialViewAsync()
   {
      return IsLoggedIn ? NavigateToMainAsync<ProgressOfHabitsViewModel>() :
NavigateToAsync<LoginViewModel>(isAbsoluteRoute: true);
   }

   public async Task NavigateToMainAsync<TViewModel>() where TViewModel : BaseViewModel
   {
      await InternalNavigateToAsync(typeof(TViewModel), routeParameters: null, isMainRoute: true, isAbsoluteRoute:
false);
   }

   public Task NavigateToAsync<TViewModel>() where TViewModel : BaseViewModel
   {
      return NavigateToAsync<TViewModel>( isAbsoluteRoute: false );
   }

   public async Task NavigateToAsync<TViewModel>( bool isAbsoluteRoute ) where TViewModel : BaseViewModel
   {
      await InternalNavigateToAsync(typeof(TViewModel), routeParameters: null, isMainRoute: false, isAbsoluteRoute
);
   }

   public Task NavigateToAsync<TViewModel>( IDictionary<string, object> routeParameters ) where TViewModel :
BaseViewModel
   {
      return NavigateToAsync<TViewModel>( isAbsoluteRoute: false, routeParameters );
   }

   public Task NavigateToAsync<TViewModel>( bool isAbsoluteRoute, IDictionary<string, object> routeParameters )
where TViewModel : BaseViewModel
   {
      return InternalNavigateToAsync( typeof( TViewModel ), routeParameters, isMainRoute: false, isAbsoluteRoute );
   }
```

```
    public Task NavigateToAsync<TViewModel>( Guid? id ) where TViewModel : BaseViewModel
    {
        return NavigateToAsync<TViewModel>( isAbsoluteRoute: false, id );
    }

    public Task NavigateToAsync<TViewModel>( bool isAbsoluteRoute, Guid? id ) where TViewModel :
BaseViewModel
    {
        Dictionary<string, object> parameters = null;
        if( id != null )
        {
            parameters = new();
            parameters.Add( key: "Id", value: id );
        }

        return InternalNavigateToAsync( typeof( TViewModel ), parameters, isMainRoute: false, isAbsoluteRoute );
    }

    public Task GoBackAsync()
    {
        return Shell.Current.GoToAsync(state: "..", animate: true);
    }

    private static Task InternalNavigateToAsync( Type viewModelType, IDictionary<string, object> routeParameters,
bool isMainRoute, bool isAbsoluteRoute )
    {
        string route = viewModelType.Name.
            Replace( oldValue: "ViewModel", newValue: "" ).
            ToLower( CultureInfo.GetCultureInfo( name: "en" ) );

        string absolutePrefix;
        if (isMainRoute)
        {
            absolutePrefix = "//main/";
        }
        else
        {
            absolutePrefix = isAbsoluteRoute ? "///" : "";
        }
        route = $"{absolutePrefix}{route}";

        ShellNavigationState shellNavigation = new( route );

        bool doAnimation = true;
        Task navigateTask = routeParameters?.Count >= 1 ?
            Shell.Current.GoToAsync( shellNavigation, doAnimation, routeParameters ) :
            Shell.Current.GoToAsync( shellNavigation, doAnimation );
        return navigateTask;
    }
}
```

Файл: ProfileView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<views:ContentPageBase x:Class="SET.MAUI.Views.ProfileView"
                       xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
                       xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
                       xmlns:dxco="clr-namespace:DevExpress.Maui.Controls;assembly=DevExpress.Maui.Controls"
                       xmlns:dxe="clr-namespace:DevExpress.Maui.Editors;assembly=DevExpress.Maui.Editors"
                       xmlns:dxcv="clr-
namespace:DevExpress.Maui.CollectionView;assembly=DevExpress.Maui.CollectionView"
                       xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"
                       xmlns:loc="clr-namespace:SET.MAUI.Resources.AppStrings"
                       xmlns:views="clr-namespace:SET.MAUI.Views"
```

```
            xmlns:vm="clr-namespace:SET.MAUI.ViewModels"
            xmlns:models="clr-namespace:SET.Core.Models;assembly=SET.Core"
            xmlns:templates="clr-namespace:SET.MAUI.Views.Templates"
            x:DataType="vm:ProfileViewModel"
            Title="{x:Static loc:LocStrings.Profile}">
<views:ContentPageBase.ToolbarItems>
   <ToolbarItem IconImageSource="settings"
          Command="{Binding GoToSettingsCommand}" />
</views:ContentPageBase.ToolbarItems>

<VerticalStackLayout Padding="10, 0, 10, 0">

   <Grid ColumnSpacing="10">
      <Grid.ColumnDefinitions>
         <ColumnDefinition Width="50" />
         <ColumnDefinition />
      </Grid.ColumnDefinitions>

      <Grid Margin="0, 0, 0, 0"
          Padding="0"
          RowSpacing="0"
          ColumnSpacing="0"
          Grid.Column="0">
         <Frame BackgroundColor="{StaticResource LightPrimary}"
              Margin="0"
              Padding="0"
              WidthRequest="48"
              HeightRequest="48"
              VerticalOptions="Center"
              HorizontalOptions="Center"
              HasShadow="False"
              CornerRadius="24">
            <Frame.IsClippedToBounds>
               <OnPlatform x:TypeArguments="x:Boolean">
                  <On Platform="iOS">true</On>
                  <On Platform="Android">false</On>
               </OnPlatform>
            </Frame.IsClippedToBounds>

            <Image Source="user" />
         </Frame>

         <Ellipse Fill="Transparent"
                Stroke="White"
                StrokeThickness="2"
                WidthRequest="50"
                HeightRequest="50"
                HorizontalOptions="Center"
                VerticalOptions="Center" />
      </Grid>

      <dxe:TextEdit Text="{Binding UserName.Value}"
                LabelText="{x:Static loc:LocStrings.UserName}"
                ClearIconVisibility="Never"
                IsReadOnly="True"
                EndIcon="check"
                EndIconColor="{StaticResource Primary}"
                IsEndIconVisible="{Binding UserName.IsValid}"
                ErrorColor="{StaticResource Error}"
                TextChangedCommand="{Binding ValidateUserNameCommand}"
                HasError="{Binding UserName.IsValid,
                              Converter={StaticResource IsFalse}}"
                ErrorText="{Binding UserName.Errors,
                               Converter={StaticResource FirstValidationError}}"
                Margin="0, 2"
```

```
                Grid.Column="1">
            <dxe:TextEdit.GestureRecognizers>
                <TapGestureRecognizer x:Name="TGR_Name"
                                      Tapped="TGR_Name_Tapped"/>
            </dxe:TextEdit.GestureRecognizers>
        </dxe:TextEdit>
    </Grid>

    <dxe:MultilineEdit Text="{Binding MainSlogan}"
                       LabelText="{x:Static loc:LocStrings.MainSlogan}"
                       ClearIconVisibility="Never"
                       IsReadOnly="True"
                       EndIcon="check"
                       EndIconColor="{StaticResource Primary}"
                       Margin="0, 2">
        <dxe:MultilineEdit.GestureRecognizers>
            <TapGestureRecognizer x:Name="TGR_MainSlogan"
                                  Tapped="TGR_MainSlogan_Tapped"/>
        </dxe:MultilineEdit.GestureRecognizers>
    </dxe:MultilineEdit>

    <dxe:MultilineEdit Text="{Binding Mission}"
                       LabelText="{x:Static loc:LocStrings.Mission}"
                       ClearIconVisibility="Never"
                       IsReadOnly="True"
                       EndIcon="check"
                       EndIconColor="{StaticResource Primary}"
                       Margin="0, 2">
        <dxe:MultilineEdit.GestureRecognizers>
            <TapGestureRecognizer x:Name="TGR_Mission"
                                  Tapped="TGR_Mission_Tapped"/>
        </dxe:MultilineEdit.GestureRecognizers>
    </dxe:MultilineEdit>

    <dxco:DXPopup x:Name="DXP_Prompt"
                  AllowAnimation="True"
                  AllowScrim="True"
                  AllowShadow="True"
                  CloseOnScrimTap="True"
                  MinimumWidthRequest="300">
        <VerticalStackLayout>
            <Label x:Name="L_Prompt"
                   HorizontalOptions="Center"
                   VerticalOptions="Center"
                   Margin="0, 10, 0, 0"
                   FontSize="Subtitle"
                   FontAttributes="Bold"/>
            <dxe:MultilineEdit x:Name="ME_PromptResult"
                               ClearIconVisibility="Never"
                               SizeChanged="ME_PromptResult_SizeChanged"
                               Margin="10"
                               TextFontSize="Medium"/>
            <dxco:SimpleButton x:Name="SB_Save"
                               Style="{StaticResource PrimaryButton}"
                               Text="{Binding LocSave}"
                               HorizontalOptions="End"
                               FontSize="Medium"
                               Margin="0, 5, 5, 5"
                               Clicked="SB_Save_Clicked"/>
        </VerticalStackLayout>
    </dxco:DXPopup>

    <templates:DefaultActivityIndicator />
</VerticalStackLayout>
```

</views:ContentPageBase>

Файл: ProgressOfHabitService.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace SET.Core.Services;

public class ProgressOfHabitService : BaseRemoteService, IProgressOfHabitService
{
   public ProgressOfHabitService( IServiceProvider serviceProvider )
      : base( serviceProvider )
   {
      //do nothing
   }

   public async Task UpdateAsync( ProgressOfHabit progressOfHabit )
   {
      #region Check parameter
      if (progressOfHabit.Id == default)
      {
         throw new ArgumentException( message: $"{nameof( progressOfHabit )}.{nameof( progressOfHabit.Id )} is
default" );
      }

      if(progressOfHabit.Habit == null)
      {
         throw new ArgumentException( message: $"{nameof( progressOfHabit )}.{nameof( progressOfHabit.Habit )} is
null" );
      }

      if(progressOfHabit.Habit.Frequency == null)
      {
         throw new ArgumentException( message: $"{nameof( progressOfHabit )}.{nameof( progressOfHabit.Habit
)}.{nameof( progressOfHabit.Habit.Frequency )} is null" );
      }
      #endregion

      UserHabit habit = progressOfHabit.Habit;

      if (habit.Frequency.IntervalType == IntervalType.Year)
      {
         var zeroTime = TimeOnly.FromTimeSpan( TimeSpan.Zero );
         DateTime firstDateOfYear = new( progressOfHabit.Date.Year, month: 1, day: 1 );
         DateTime earliestDateTime = habit.Progresses.Select( p => p.Date ).Min().ToDateTime( zeroTime );
         DateTime largestDateTime = habit.Progresses.Select( p => p.Date ).Max().ToDateTime( zeroTime );

         //get diff of lowest and largest progress.Date
         int diffOfEarliestAndLargest = (int)(largestDateTime - earliestDateTime).Days;
         //get diff of value.Date and current year
         int diffOfDateAndYearStart = (int)(progressOfHabit.Date.ToDateTime( zeroTime ) - firstDateOfYear).Days;

         if (diffOfEarliestAndLargest < diffOfDateAndYearStart)
         {
            if (progressOfHabit.IsCompleted)
            {
               habit.CountOfFollowedPerSpecificInterval++;
            }
            else
            {
```

```
                habit.CountOfFollowedPerSpecificInterval--;
            }
        }
    }

    string isCompletedAsStr = progressOfHabit.IsCompleted ? "check" : "uncheck";
    double beforeProgress = habit.PercentageAchieved;

    string url = $"{UrlBuilder.ProgressOfHabit}/{progressOfHabit.Id}";
    UpdateProgressDto dto = new()
    {
        Id = progressOfHabit.Id,
        PercentageAchieved = habit.PercentageAchieved,
        PreviousPercentageAchieved = habit.PreviousPercentageAchieved,
        Date = progressOfHabit.Date,
        IsCompleted = progressOfHabit.IsCompleted,
        HabitId = habit.Id,
        FollowedHabitCount = habit.FollowedCount
    };
    ResponseOfUpdateProgressPost response = await RequestProvider.PostAsync<UpdateProgressDto,
ResponseOfUpdateProgressPost>(
        url,
        dto,
        SettingsService.AuthAccessToken
    );
    habit.PreviousPercentageAchieved = response.PreviousPercentageAchieved;
    habit.PercentageAchieved = response.PercentageAchived;

    LoggingService.LogInfo( $"Before progress of habit: {beforeProgress};{Environment.NewLine}" +
        $"After {isCompletedAsStr}: {habit.PercentageAchieved}." );
}

public double ComputeScore( double frequency, double previousScore, double checkmarkValue, int complexity )
{
    double coefOfComplexity = complexity switch
    {
        1 => 5.6,
        2 => 3.33,
        3 => 2.3,
        4 => 1.7,
        5 => 1.525,
        6 => 1.4,
        7 => 1.0,
        8 => 0.77,
        9 => 0.524,
        10 => 0.392,
        _ => throw new ArgumentException( message: $"Complexity {complexity} of habit is not supported",
        paramName: nameof( complexity ) )
    };

    double exponentialSmoothingFactor = 0.5;
    double scalingFactor = 13.0;
    double decayFactor = Math.Pow( exponentialSmoothingFactor, Math.Sqrt( frequency ) * coefOfComplexity /
scalingFactor );

    double score = previousScore * decayFactor;
    score += checkmarkValue * (1 - decayFactor);
    return score;
}

public int ConvertScoreToPercentage( double score )
{
    return (int)Math.Round( score * 100.0, MidpointRounding.ToEven );
}
```

}

Файл: ProgressOfHabitsView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<v:ContentPageBase xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
          xmlns:dxcv="clr-namespace:DevExpress.Maui.CollectionView;assembly=DevExpress.Maui.CollectionView"
          xmlns:dxg="clr-namespace:DevExpress.Maui.DataGrid;assembly=DevExpress.Maui.DataGrid"
             xmlns:dxco="clr-namespace:DevExpress.Maui.Controls;assembly=DevExpress.Maui.Controls"
             xmlns:dx="clr-namespace:DevExpress.Maui.Core;assembly=DevExpress.Maui.Core"
             xmlns:skia="clr-namespace:SkiaSharp.Extended.UI.Controls;assembly=SkiaSharp.Extended.UI"
          xmlns:vm="clr-namespace:SET.MAUI.ViewModels"
          xmlns:v="clr-namespace:SET.MAUI.Views"
          xmlns:m="clr-namespace:SET.Core.Models;assembly=SET.Core"
             xmlns:loc="clr-namespace:SET.MAUI.Resources.AppStrings"
          xmlns:controls="clr-namespace:SET.MAUI.Controls"
          IconImageSource="fire_third.svg"
          xmlns:ios="clr-
namespace:Microsoft.Maui.Controls.PlatformConfiguration.iOSSpecific;assembly=Microsoft.Maui.Controls"
          x:DataType="vm:ProgressOfHabitsViewModel"
          x:Class="SET.MAUI.Views.ProgressOfHabitsView"
          Shell.NavBarIsVisible="True">
   <Shell.TitleView>
      <Grid VerticalOptions="CenterAndExpand"
                     ColumnDefinitions="Auto, *, Auto, 10, Auto">
         <Label Text="{x:Static loc:LocStrings.AtomicHabits}"
              VerticalOptions="Center"
              VerticalTextAlignment="Center"
              HorizontalOptions="Start"
                 TextColor="{StaticResource TitleTextColor}"
              FontSize="Subtitle"
              FontAttributes="Bold"
              HeightRequest="50"
              Grid.Column="0" />
         <dxco:DXImage Source="plus"
                   IsVisible="{Binding SelectedHabit, Converter={StaticResource IsNull}}"
                     VerticalOptions="Center"
                     HorizontalOptions="End"
                      TintColor="{StaticResource TitleTextColor}"
                     Margin="65, 0, 0, 0"
                     WidthRequest="40"
                     HeightRequest="40"
                   Grid.Column="4">
            <dxco:DXImage.GestureRecognizers>
               <TapGestureRecognizer NumberOfTapsRequired="1"
                                  Command="{Binding AddHabitCommand}"/>
            </dxco:DXImage.GestureRecognizers>
         </dxco:DXImage>
         <dxco:DXImage Source="edit"
                   TintColor="{StaticResource TitleTextColor}"
                   Margin="35, 0, 0, 0"
                   HorizontalOptions="End"
                   WidthRequest="40"
                   VerticalOptions="Center"
                   IsVisible="{Binding SelectedHabit, Converter={StaticResource IsNotNull}}"
                   HeightRequest="30"
                   Grid.Column="2">
            <dxco:DXImage.GestureRecognizers>
               <TapGestureRecognizer NumberOfTapsRequired="1"
                                  Command="{Binding EditHabitCommand}"
                                  CommandParameter="{Binding SelectedHabit}"/>
            </dxco:DXImage.GestureRecognizers>
         </dxco:DXImage>
```

```
        <dxco:DXImage Source="delete"
                      TintColor="{StaticResource TitleTextColor}"
                      Margin="3, 0, 0, 0"
                      HorizontalOptions="End"
                      WidthRequest="40"
                      VerticalOptions="Center"
                      IsVisible="{Binding SelectedHabit, Converter={StaticResource IsNotNull}}"
                      HeightRequest="30"
                      Grid.Column="4">
          <dxco:DXImage.GestureRecognizers>
            <TapGestureRecognizer NumberOfTapsRequired="1"
                                  Command="{Binding DeleteHabitCommand}"
                                  CommandParameter="{Binding SelectedHabit}"/>
          </dxco:DXImage.GestureRecognizers>
        </dxco:DXImage>
      </Grid>
    </Shell.TitleView>

    <Grid BackgroundColor="{StaticResource Background}">
      <dxg:DataGridView ItemsSource="{Binding UserHabits}"
                        AllowDragDropRows="True"
                        CompleteRowDragDrop="DGV_Habits_CompleteRowDragDrop"
                    x:Name="DGV_Habits"
                    LongPress="DGV_Habits_LongPress"
                    AllowInitiallySelectedRow="False"
                        BackgroundColor="White"
                        AllowVirtualHorizontalScrolling="True"
                    VerticalLineThickness="0"
                        FullSwipeMode="Start"
                    AllowLiveDataShaping="True">
        <dxg:DataGridView.ColumnHeaderAppearance>
          <dxg:ColumnHeaderAppearance BackgroundColor="WhiteSmoke"/>
        </dxg:DataGridView.ColumnHeaderAppearance>
        <dxg:DataGridView.Columns>
          <dxg:TextColumn FieldName="Name"/>
        </dxg:DataGridView.Columns>

        <dxg:DataGridView.CellAppearance>
          <dxg:CellAppearance SelectionColor="LightGray"/>
        </dxg:DataGridView.CellAppearance>
      </dxg:DataGridView>

      <dx:DXStackLayout Orientation="Vertical"
                        ItemSpacing="10"
                        HorizontalOptions="CenterAndExpand"
                        VerticalOptions="Start"
                        IsVisible="{Binding UserHabits.Count, Converter={StaticResource IsEmptyCollection}}">
        <skia:SKLottieView IsAnimationEnabled="{Binding UserHabits.Count, Converter={StaticResource
IsEmptyCollection}}"
                           RepeatCount="-1"
                           WidthRequest="300"
                           HeightRequest="250"
                           Margin="0, 150, 0, 0"
                           Source="checks.json" />
        <Label Text="{x:Static loc:LocStrings.LoadingContent}"
               IsVisible="{Binding IsBusy}"
               FontFamily="MonaSansMedium"
               FontSize="14"
               TextColor="{AppThemeBinding Light={StaticResource LightNormalText}, Dark={StaticResource
DarkNormalText}}"
               HorizontalTextAlignment="Center"/>
        <Label Text="{x:Static loc:LocStrings.HabitCollectionIsEmptyDescription}"
               IsVisible="{Binding IsBusy, Converter={StaticResource IsFalse}}"
               FontFamily="MonaSansMedium"
               FontSize="14"
```

```
            TextColor="{AppThemeBinding Light={StaticResource LightNormalText}, Dark={StaticResource
DarkNormalText}}"
            HorizontalTextAlignment="Center"/>
    </dx:DXStackLayout>
  </Grid>
</v:ContentPageBase>
```

Файл: SettingsView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<v:ContentPageBase xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="SET.MAUI.Views.SettingsView"
            xmlns:loc="clr-namespace:SET.MAUI.Resources.AppStrings"
            xmlns:dxco="clr-namespace:DevExpress.Maui.Controls;assembly=DevExpress.Maui.Controls"
         xmlns:dxg="clr-namespace:DevExpress.Maui.DataGrid;assembly=DevExpress.Maui.DataGrid"
            xmlns:dxe="clr-namespace:DevExpress.Maui.Editors;assembly=DevExpress.Maui.Editors"
            xmlns:dxcv="clr-
namespace:DevExpress.Maui.CollectionView;assembly=DevExpress.Maui.CollectionView"
            xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"
            xmlns:templates="clr-namespace:SET.MAUI.Views.Templates"
            xmlns:v="clr-namespace:SET.MAUI.Views"
            xmlns:vm="clr-namespace:SET.MAUI.ViewModels"
            xmlns:m="clr-namespace:SET.Core.Models;assembly=SET.Core"
            x:DataType="vm:SettingsViewModel"
            Title="{x:Static loc:LocStrings.Settings}">
  <Grid>
    <VerticalStackLayout Padding="15"
                 Spacing="20"
                 VerticalOptions="CenterAndExpand">
      <Image Source="logolargesize"
             WidthRequest="200"
             HeightRequest="200"/>
      <Label Text="{Binding AppName, Converter={StaticResource ToUpperCase}}"
          TextColor="OrangeRed"
          HorizontalOptions="Center"
          FontSize="Title"
          FontFamily="Roboto"
          FontAttributes="Bold"/>
      <dxco:SimpleButton Command="{Binding LogoutCommand}"
                 Text="{x:Static loc:LocStrings.Logout}"
                 Style="{StaticResource PrimaryButton}" />
      <dxco:SimpleButton Command="{Binding DeleteAccountCommand}"
                 Text="{x:Static loc:LocStrings.DeleteAccount}"
                 BackgroundColor="{AppThemeBinding Light={StaticResource DarkPrimary}, Dark={StaticResource
Primary}}"
                 TextColor="{AppThemeBinding Light={StaticResource Primary}, Dark={StaticResource
DarkPrimary}}"/>
    </VerticalStackLayout>

    <templates:DefaultActivityIndicator />
  </Grid>
</v:ContentPageBase>
```

Файл: SignupView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<v:ContentPageBase xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="SET.MAUI.Views.SignupView"
            xmlns:loc="clr-namespace:SET.MAUI.Resources.AppStrings"
            xmlns:dxco="clr-namespace:DevExpress.Maui.Controls;assembly=DevExpress.Maui.Controls"
```

```
            xmlns:dxe="clr-namespace:DevExpress.Maui.Editors;assembly=DevExpress.Maui.Editors"
            xmlns:skia="clr-namespace:SkiaSharp.Extended.UI.Controls;assembly=SkiaSharp.Extended.UI"
            xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"
            xmlns:vm="clr-namespace:SET.MAUI.ViewModels"
            xmlns:v="clr-namespace:SET.MAUI.Views"
            x:DataType="vm:SignupViewModel"
            Shell.FlyoutBehavior="Disabled"
            Shell.NavBarIsVisible="True"
            Shell.TabBarIsVisible="False"
            Title="{x:Static loc:LocStrings.SignUp}">
<ScrollView>
  <Grid>
    <VerticalStackLayout Padding="15, 15, 15, 15"
                         VerticalOptions="StartAndExpand"
                         Spacing="10">
      <skia:SKLottieView IsAnimationEnabled="True"
                         RepeatCount="-1"
                         WidthRequest="500"
                         HeightRequest="200"
                         Source="one_check.json" />

      <dxe:TextEdit LabelText="{x:Static loc:LocStrings.UserName}"
                    Text="{Binding Name.Value}"
                    StartIcon="user"
                    HasError="{Binding Name.IsValid,
                                       Converter={StaticResource IsFalse}}"
                    ErrorText="{Binding Name.Errors,
                                        Converter={StaticResource FirstValidationError}}"
                    TextChangedCommand="{Binding ValidateNameCommand}"
                    AutofillContentType="Username"
                    ClearIconVisibility="Never" />
      <dxe:TextEdit LabelText="{x:Static loc:LocStrings.Email}"
                    StartIcon="email"
                    AutofillContentType="EmailAddress"
                    Text="{Binding Email.Value}"
                    Keyboard="Email"
                    HasError="{Binding Email.IsValid,
                                       Converter={StaticResource IsFalse}}"
                    ErrorText="{Binding Email.Errors,
                                        Converter={StaticResource FirstValidationError}}"
                    TextChangedCommand="{Binding ValidateEmailCommand}"
                    ClearIconVisibility="Never"
                    x:Name="EmailTextEdit" />
      <dxe:PasswordEdit LabelText="{x:Static loc:LocStrings.Password}"
                        Text="{Binding Password.Value}"
                        StartIcon="password"
                        AutofillContentType="Password"
                        HasError="{Binding Password.IsValid,
                                           Converter={StaticResource IsFalse}}"
                        ErrorText="{Binding Password.Errors,
                                            Converter={StaticResource FirstValidationError}}"
                        TextChangedCommand="{Binding ValidatePasswordCommand}"
                        ClearIconVisibility="Never"
                        x:Name="PasswordTextEdit" />

      <dxe:ChoiceChipGroup IsMultiline="False"
                           x:Name="CCG_Genders"
                           SelectionChanged="CCG_Genders_SelectionChanged"
                           HorizontalOptions="Center">
        <dxe:Chip Text="{x:Static loc:LocStrings.Man}" />
        <dxe:Chip Text="{x:Static loc:LocStrings.Woman}" />
        <dxe:Chip Text="{x:Static loc:LocStrings.OtherSex}" />
      </dxe:ChoiceChipGroup>

      <dxco:SimpleButton FontAttributes="Bold"
```

```
                    Text="{x:Static loc:LocStrings.SignUp}"
                    Command="{Binding SignupCommand}"
                    Style="{StaticResource PrimaryButton}" />
        </VerticalStackLayout>

        <ActivityIndicator Color="{StaticResource Primary}"
                    IsRunning="{Binding IsBusy}"
                    IsVisible="{Binding IsBusy}"
                    VerticalOptions="Center"
                    HorizontalOptions="Center"
                    WidthRequest="100" />
    </Grid>
  </ScrollView>
</v:ContentPageBase>
```

Файл: SignupView.xaml.cs

```
namespace SET.MAUI.Views;

public partial class SignupView : ContentPageBase
{
  public SignupView( SignupViewModel viewModel )
  {
    BindingContext = viewModel;
    ViewModel = viewModel;
    InitializeComponent();

    CCG_Genders.SelectChipAtIndex( index: 0 );
  }

  public SignupViewModel ViewModel { get; }

  private void CCG_Genders_SelectionChanged( object sender, EventArgs e )
  {
    if (CCG_Genders.SelectedChip != null)
    {
      Gender gender = (Gender)CCG_Genders.SelectedIndex;

      ViewModel.SetGenderCommand.Execute( gender.ToString() );
    }
  }
}
```

Файл: BaseTemplateView.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<v:ContentPageBase xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="SET.MAUI.Views.BaseTemplateView"
            xmlns:loc="clr-namespace:SET.MAUI.Resources.AppStrings"
            xmlns:dxco="clr-namespace:DevExpress.Maui.Controls;assembly=DevExpress.Maui.Controls"
            xmlns:dxe="clr-namespace:DevExpress.Maui.Editors;assembly=DevExpress.Maui.Editors"
            xmlns:dxcv="clr-
namespace:DevExpress.Maui.CollectionView;assembly=DevExpress.Maui.CollectionView"
            xmlns:skia="clr-namespace:SkiaSharp.Extended.UI.Controls;assembly=SkiaSharp.Extended.UI"
            xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"
            xmlns:v="clr-namespace:SET.MAUI.Views"
            xmlns:vm="clr-namespace:SET.MAUI.ViewModels"
            xmlns:m="clr-namespace:SET.Core.Models;assembly=SET.Core"
            xmlns:templates="clr-namespace:SET.MAUI.Views.Templates"
            xmlns:controls="clr-namespace:SET.MAUI.Controls"
            x:DataType="vm:MainViewModel"
            Title="{x:Static loc:LocStrings.Profile}">
  <VerticalStackLayout>
```

```
    </VerticalStackLayout>
</v:ContentPageBase>
```

Файл: HabitWithProgressTemplateSelector.cs

```
using DevExpress.Maui.DataGrid;
using DevExpress.Maui.Editors;

using System;
using System.Collections.Concurrent;

namespace SET.MAUI.Views.Templates;

public class HabitWithProgressTemplateSelector : DataTemplateSelector
{
    private readonly int m_columnIndex;
    private readonly ProgressOfHabitsViewModel m_progressOfHabitsViewModel;
    private readonly DataTemplate m_dataTemplate;

    public HabitWithProgressTemplateSelector(ProgressOfHabitsViewModel viewModel, int columnIndex)
    {
        m_progressOfHabitsViewModel = viewModel;
        m_columnIndex = columnIndex;
        m_dataTemplate = new DataTemplate( loadTemplate: () =>
        {
            CheckEdit checkEdit = new()
            {
                Margin = new Thickness( 15, 0, 0, 0 )
            };

            checkEdit.SetBinding( CheckEdit.IsCheckedProperty, binding: new Binding( path:
$"Item.Progresses[{m_columnIndex}].IsCompleted" ) );
            checkEdit.CheckedCheckBoxImage = ImageSource.FromFile( "logo.png" );

            Color primaryColor = (Color)Application.Current.Resources["Primary"];
            checkEdit.CheckedCheckBoxColor = primaryColor;

            IValueConverter uncheckedProgressToImgConverter = new UncheckedProgressToImgConverter(
viewModel.ServiceOfHabit );
            checkEdit.Bind(
                targetProperty: CheckEdit.UncheckedCheckBoxImageProperty,
                path: $"Item.Progresses[{m_columnIndex}]",
                converter: uncheckedProgressToImgConverter
            );
            checkEdit.BindTapGesture( commandPath: "ChangeValueOfProgressOfHabitCommand", commandSource:
m_progressOfHabitsViewModel, parameterPath: $"Item.Progresses[{m_columnIndex}]", numberOfTapsRequired: 1 );

            return checkEdit;
        } );
    }

    protected override DataTemplate OnSelectTemplate( object item, BindableObject container )
    {
        return m_dataTemplate;
    }
}
```

Файл: HelperMessageTemplate.xaml

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xaml-comp compile="true" ?>
<Grid xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
```

```
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="SET.MAUI.Views.Templates.HelperMessageTemplate"
    xmlns:msgs="clr-namespace:SET.MAUI.Messages"
    xmlns:dxe="clr-namespace:DevExpress.Maui.Editors;assembly=DevExpress.Maui.Editors"
    xmlns:dx="clr-namespace:DevExpress.Maui.Core;assembly=DevExpress.Maui.Core"
    x:DataType="msgs:DisplayMessage"
    ColumnDefinitions="35, 10, Auto, *"
    Padding="16, 0, 0, 0"
    x:Name="G_HelperMessageTemplate">

  <Grid Grid.Column="0"
      x:Name="G_HelperIcon"
      WidthRequest="35"
      HeightRequest="35"
      VerticalOptions="End">
    <BoxView Color="{AppThemeBinding Light={StaticResource Primary}, Dark={StaticResource DarkPrimary}}"
          CornerRadius="5, 5, 5, 0" />
    <dx:DXImage Source="ai"
            TintColor="White"
            WidthRequest="22"
            HeightRequest="22"/>
  </Grid>

  <Grid Grid.Column="2"
      MaximumWidthRequest="260"
      x:Name="G_Answer">
    <BoxView CornerRadius="16, 16, 0, 16"
          Color="{AppThemeBinding Light={StaticResource TertiaryLight}, Dark={StaticResource
TertiaryDark}}"/>
    <dxe:MultilineEdit BackgroundColor="{AppThemeBinding Light={StaticResource TertiaryLight},
Dark={StaticResource TertiaryDark}}"
                TextFontSize="18"
                TextColor="{AppThemeBinding Light={StaticResource LightNormalText}, Dark={StaticResource
DarkNormalText}}"
                x:Name="ME_Answer"
                Text="{Binding Text}"
                TextFontFamily="MonaSansMedium"
                BorderThickness="0"
                ClearIconVisibility="Never"
                IsReadOnly="True"
                BoxPadding="5"
                Margin="10, 5"
                VerticalOptions="Start"
                BoxMinHeight="10"
                MinimumHeightRequest="18"
                TextChanged="ME_Answer_TextChanged"/>
  </Grid>
</Grid>
```

Файл: HelperMessageTemplate.xaml.cs

```
using Microsoft.Maui.Handlers;

namespace SET.MAUI.Views.Templates;

public partial class HelperMessageTemplate : Grid
{
  private Grid m_parentGrid;
  private HelperViewModel m_viewModel;

  public HelperMessageTemplate()
  {
```

```
        InitializeComponent();
    }

    void ME_Answer_TextChanged( object sender, EventArgs e )
    {
        if (ME_Answer.Height != -1 && G_HelperMessageTemplate.Height < ME_Answer.Height + 10)
        {
            G_HelperMessageTemplate.HeightRequest = ME_Answer.Height + 10;
        }
    }
}
```

Файл: UserMessageTemplate.xaml

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xaml-comp compile="true" ?>
<Grid xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
      xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
      xmlns:msgs="clr-namespace:SET.MAUI.Messages"
      xmlns:dxco="clr-namespace:DevExpress.Maui.Controls;assembly=DevExpress.Maui.Controls"
      xmlns:dxe="clr-namespace:DevExpress.Maui.Editors;assembly=DevExpress.Maui.Editors"
      x:Class="SET.MAUI.Views.Templates.UserMessageTemplate"
      x:DataType="msgs:DisplayMessage"
      x:Name="G_UserMessageTemplate"
      Padding="0, 0, 16, 0"
      ColumnDefinitions="*, Auto, 10, Auto">

    <Grid Grid.Column="1"
          MaximumWidthRequest="260"
          x:Name="G_Question">
        <BoxView CornerRadius="16, 16, 16, 0"
                 Color="{AppThemeBinding Light={StaticResource LightPrimary}, Dark={StaticResource
LightPrimary}}"/>
        <dxe:MultilineEdit BackgroundColor="{AppThemeBinding Light={StaticResource LightPrimary},
Dark={StaticResource LightPrimary}}"
                           TextFontSize="18"
                           TextColor="{AppThemeBinding Light={StaticResource LightNormalText}, Dark={StaticResource
DarkNormalText}}"
                           x:Name="ME_Question"
                           Text="{Binding Text}"
                           TextFontFamily="MonaSansMedium"
                           BorderThickness="0"
                           ClearIconVisibility="Never"
                           IsReadOnly="True"
                           BoxPadding="5"
                           Margin="10, 5"
                           VerticalOptions="Start"
                           BoxMinHeight="10"
                           MinimumHeightRequest="18"
                           TextVerticalAlignment="Center"/>
    </Grid>

    <Grid Grid.Column="3"
          x:Name="G_UserIcon"
          WidthRequest="35"
          HeightRequest="35"
          VerticalOptions="End">
        <BoxView Color="{StaticResource LightPrimary}"
                 CornerRadius="5, 5, 0, 5" />
        <dxco:DXImage Source="user"
                      WidthRequest="22"
                      HeightRequest="22"/>
    </Grid>
</Grid>
```