

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГТУ»)

Факультет информационных технологий и компьютерной безопасности
(факультет)

Кафедра систем управления и информационных технологий в строительстве

КУРСОВОЙ ПРОЕКТ

по дисциплине Инструменты и методы построения пользовательского интерфейса
тема Разработка программного продукта «Электронная книга кулинарных рецептов» с использованием технологии MAUI

Расчетно-пояснительная записка

Разработал студент

Д.В. Тюленев
Подпись, дата Инициалы, фамилия

Руководитель

Н.В. Акамсина
Подпись, дата Инициалы, фамилия

Члены комиссии

Подпись, дата Инициалы, фамилия

Подпись, дата Инициалы, фамилия

Нормоконтролер

Н.В. Акамсина
Подпись, дата Инициалы, фамилия

Защищена 27.12.23
дата

Оценка отлично

2023

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГТУ»)

Кафедра Систем автоматизированного проектирования и информационных систем

ЗАДАНИЕ
на курсовой проект

по дисциплине Инструменты и методы построения пользовательского интерфейса

тема Разработка программного продукта «Электронная книга кулинарных рецептов» с использованием технологии MAUI

Студент группы БОИС-211

Тюленев Данил Вячеславович

Фамилия, имя, отчество

Технические условия процессор Inter Core I5 3.45 ГГц, операционная система Windows 11, ОЗУ 16384 МБ.

Содержание и объем проекта (графические работы, расчеты и прочее):
анализ предметной области (6 страниц); моделирование системы (5 страниц);
разработка приложения информационной системы (16 страниц); 17 рисунков, 1
приложение.

Сроки выполнения этапов анализ предметной области (10.09.2023 – 29.09.2023);
моделирование информационной подсистемы (29.09.2023-15.10.2023); разработка
информационной подсистемы (11.10.2023- 29.11.2023); оформление расчетно-
пояснительной записки (30.11.2023-21.12.2023)

Срок защиты курсового проекта 27.12.23


Руководитель



Подпись, дата

Н.В. Акамсина
Инициалы, фамилия

Задание принял студент

08.09.23 

Подпись, дата

Д.В. Тюленев
Инициалы, фамилия

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
1 Анализ предметной области	5
2 Моделирование программного продукта «Кулинарные рецепты»	10
2.1 Разработка модели IDEF0	10
2.2 Диаграмма последовательности	12
2.3 Проектирование базы данных	14
3 Разработка программного продукта	16
3.1 Аппаратное и программное обеспечение необходимое для реализации программного продукта	16
3.2 Реализация программного продукта	18
3.3 Пример работы программного продукта	23
ЗАКЛЮЧЕНИЕ	32
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ	33
ПРИЛОЖЕНИЕ А	35

ВВЕДЕНИЕ

В современном мире развитие мобильных технологий играет ключевую роль в удовлетворении повседневных потребностей пользователей. Одной из наиболее популярных областей приложений является кулинария, где мобильные приложения для рецептов и приготовления блюд становятся незаменимыми помощниками в кулинарном процессе. В данном контексте разработка инновационного и удобного в использовании приложения для кулинарных рецептов представляет собой актуальную задачу.

Целью данного курсового проекта является разработка приложения «Электронная книга кулинарных рецептов», с использованием платформы MAUI (Multi-platform App UI). Платформа MAUI предоставляет возможность создания кроссплатформенных мобильных приложений с общим кодом для различных операционных систем. Выбор MAUI обусловлен стремлением к максимальной эффективности и экономии ресурсов при разработке приложения.

В данном введении будут рассмотрены актуальность темы, обзор существующих решений в данной области, а также основные задачи и цели курсового проекта. Разработка приложения для кулинарных рецептов на платформе MAUI позволит создать универсальный инструмент для гурманов, обеспечивая удобство использования и доступность на различных устройствах.

Основными задачами проекта являются:

1. Изучение и анализ предметной области и аналогов программного продукта;
2. Моделирование информационной системы с помощью унифицированного языка моделирования UML;
3. Проектирование и разработка базы данных;
4. Разработка программы с использованием технологии MAUI.

1 Анализ предметной области

Современные тенденции в области кулинарии свидетельствуют о растущем интересе пользователей к готовке в домашних условиях. С развитием социальных сетей и популяризацией кулинарных шоу, пользователи становятся более активными в поиске новых рецептов и технологий приготовления. Программные приложения в этой области должны учитывать данную динамику и предоставлять удобные инструменты для удовлетворения растущих потребностей аудитории [1,2].

Пользователи ищут удобные инструменты для поиска новых рецептов, основанных на доступных ингредиентах. Анализ требований пользователей в этой сфере выявляет необходимость эффективной системы фильтрации и поиска. Визуальный аспект играет важную роль в выборе блюда. Пользователи ожидают наличие качественных фотографий, видеоинструкций и возможности поделиться своими кулинарными шедеврами [3, 4].

В ходе анализа предметной области определены основные функциональные требования к приложению, также проведён анализ существующих аналогов. В результате выполнения данного курсового проекта будет разработано приложение «кулинарные рецепты», способное значительно помочь пользователям с поиском кулинарных рецептов [5, 6].

Предметная область кулинарных рецептов включает в себя следующие особенности:

1. Поиск кулинарных рецептов: основная задача программного продукта - предоставить пользователям системы удобный поиск кулинарных рецептов по их предпочтениям, с учётом выбранной категории и фильтров.

2. Сохранение понравившихся рецептов: предоставить пользователю функциональность сохранения рецептов в своей библиотеки для будущего повторного просмотра [7].

3. Возможность оценить рецепт: предоставить пользователю возможность написать рецензию на выбранный кулинарных рецепт для формирования его

общего рейтинга, а также обеспечить удобным функционалом просмотра чужих комментариев.

4. Администрирование системы: приложение должно иметь роль автора, пользователи данной роли имеют возможность добавлять, удалять и изменять кулинарные рецепты, которые доступны в системе для обычных пользователей.

Исследование существующих приложений для кулинарных рецептов позволяет выделить успешные решения, а также выявить пробелы, которые можно заполнить новым приложением. Анализ конкурентов также помогает определить уникальные возможности, которые могут выделить приложение на рынке [8].

При анализе существующих аналогов приложений с возможностью просмотра кулинарных рецептов, можно обратить внимание на следующие продукты:

1. Cookpad — это социальная платформа для обмена рецептами, где пользователи могут создавать свои кулинарные записи, делиться фотографиями блюд и общаться в сообществах. Приложение предоставляет обширную коллекцию рецептов различных блюд.

Интерфейс программы представлен на рисунке 1.1.

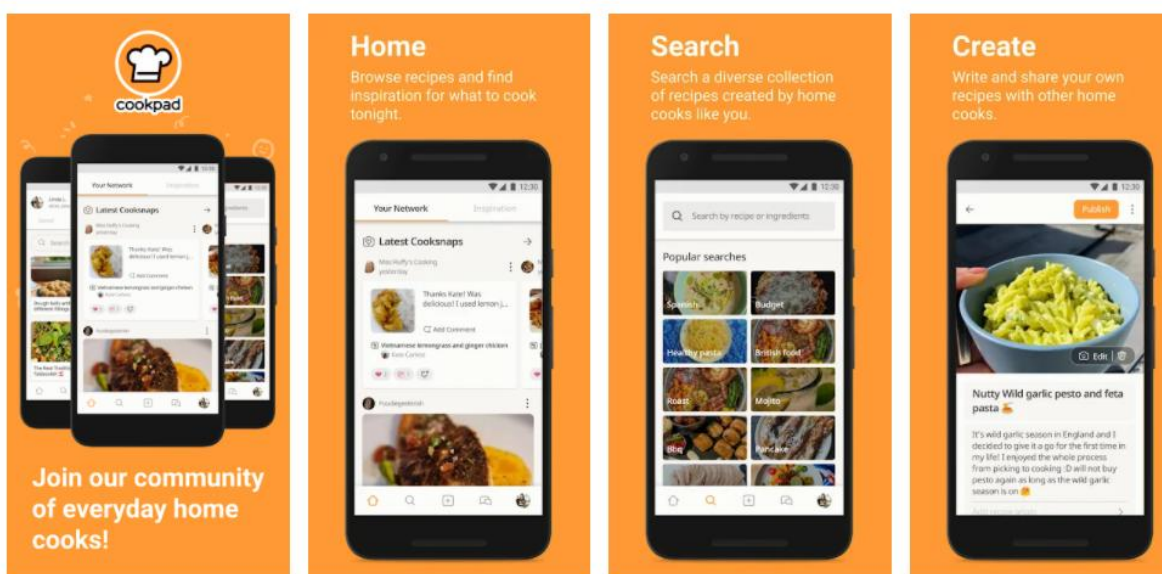


Рисунок 1.1 – Интерфейс CookPad

Проанализировав аналог Cookpad, можно выделить следующие особенности и функциональные возможности программы:

- Социальное взаимодействие: Cookpad предоставляет платформу для обмена рецептами и опытом приготовления блюд. Пользователи могут создавать свои рецепты, делиться ими с другими и взаимодействовать в сообществах;
- Широкий выбор рецептов: приложение содержит обширную базу данных с рецептами различных блюд, что позволяет пользователям находить и экспериментировать с разнообразными кулинарными идеями;
- Пользовательская активность: пользователи могут комментировать и оценивать рецепты, а также обмениваться советами и вопросами. Это создает активное сообщество, способствующее обмену опытом;
- Возможность создания профиля: пользователи могут создавать свои кулинарные профили, сохранять любимые рецепты и следить за активностью других участников;
- Функция списков покупок: Cookpad обеспечивает функционал списков покупок, что облегчает организацию и подготовку кулинарных покупок.

Cookpad является популярным выбором для тех, кто ищет социальное взаимодействие в области кулинарии и готов поделиться своим опытом, но стоит учитывать и ограничения, связанные с общим характером контента.

2. Yummly — это онлайн-платформа и мобильное приложение, ориентированное на готовку и кулинарные рецепты. Основной целью Yummly является предоставление персонализированных рецептов и кулинарных рекомендаций для пользователей.

Интерфейс программы представлен на рисунке 1.2.

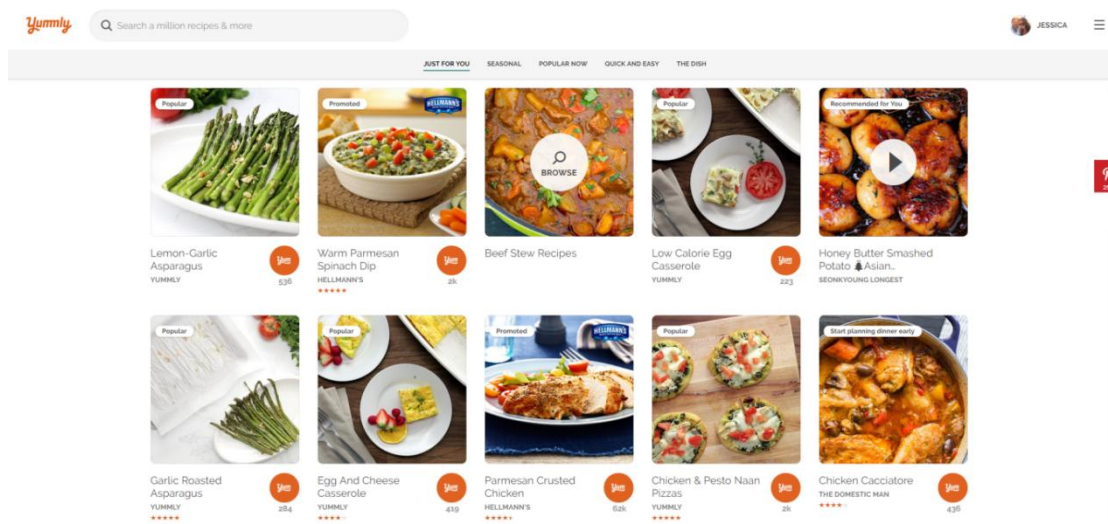


Рисунок 1.2 – Интерфейс Yummly

Вот несколько ключевых особенностей Yummly:

- Персонализированные рецепты: Yummly использует технологии искусственного интеллекта и алгоритмы рекомендаций для адаптации контента под индивидуальные предпочтения и диетические ограничения пользователя;
- Расширенный поиск и фильтрация: Пользователи могут использовать продвинутый поиск и фильтры для нахождения рецептов в соответствии с различными критериями, такими как тип блюда, ингредиенты, диетические предпочтения и т.д.;
- Интерактивные функции: Помимо рецептов, приложение предоставляет видеоинструкции, возможность создавать списки покупок, а также функцию "Кулинарный помощник" для удобного приготовления блюд;
- Планирование приемов пищи: Yummly предоставляет возможность планирования приемов пищи, составления ежедневных меню и учета калорийности блюд;
- Интерактивное сообщество: Пользователи могут обмениваться своим опытом, комментировать и оценивать рецепты. Существует возможность создания кулинарных сообществ и подписки на публикации других пользователей;
- Совместимость с умными устройствами: Yummly интегрируется с умными устройствами в кухне, позволяя управлять процессом готовки с

использованием голосовых команд или управлять рецептами с помощью смарт-девайсов.

Yummlу предоставляет удобный и инновационный способ находить, организовывать и готовить разнообразные блюда в соответствии с индивидуальными предпочтениями пользователей.

Эти приложения предоставляют различные функции, начиная от социальных элементов и искусственного интеллекта до видеорецептов и персонализированных рекомендаций, позволяя пользователям находить приложение, наилучшим образом соответствующее их потребностям и предпочтениям [9].

Благодаря анализу существующих аналогов стало понятно, каким требованиям должен соответствовать разрабатываемый продукт, а именно:

- Удобный поиск кулинарных рецептов;
- Просмотр комментариев пользователей;
- Легкий и интуитивно понятный пользовательский интерфейс;
- Безопасность данных.

Анализ аналогов, описанных выше, позволил определить основные проблемы, возникаемые в предметной области и подобрать возможные программные средства их решения [10].

2 Моделирование программного продукта «Кулинарные рецепты»

2.1 Разработка модели IDEF0

Для проектирования информационной системы будем использовать методологии языка UML [12].

При работе с системой автор будет иметь возможность решать следующие задачи: добавление и удаление кулинарных рецептов, редактирование кулинарных рецептов.

При работе с системой пользователь будет иметь возможность решать следующие задачи:

- Поиск и просмотр доступных рецептов;
- Комментирование и выставление оценок рецептам;
- Добавление рецептов в закладки для будущего повторного просмотра;
- Добавление друзей для возможности делиться рецептами.

На рисунке 2.1 представлена контекстная диаграмма стандарта IDEF0. Данная модель описывает организацию работы разрабатываемой системы [13].

Механизмы, использующие функциональность системы – пользователи и авторы рецептов. Средства управления – защита конфиденциальной информации и правила пользования системы. На вход подаётся информация о пользователе системы и данные для поиска кулинарного рецепта. Результат работы системы представляет собой документацию – найденный кулинарный рецепт, список комментариев других пользователей.

На основании выделенных компонентов диаграммы была составлена контекстная диаграмма, представленная на рисунке 2.1.

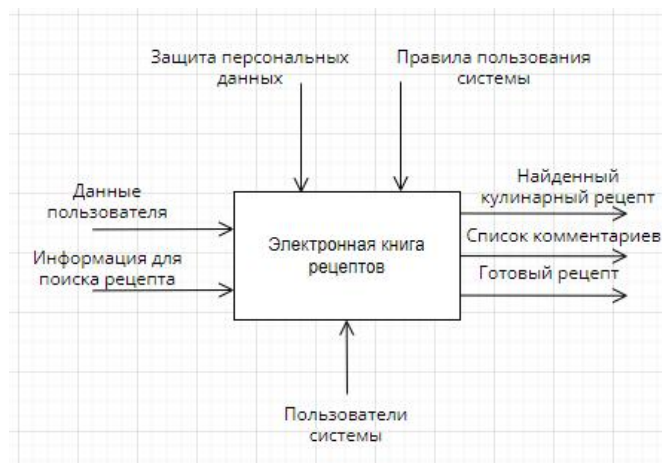


Рисунок 2.1 – Контекстная диаграмма

Диаграмма первого уровня – декомпозированная диаграмма, на которой крупно показаны основные процессы предприятия, обеспечивающие ее профильную деятельность.

Так как декомпозиция – это разложение сложного объекта, на составные части и элементы, то для осуществления декомпозиции необходимо выделить основные элементы рассматриваемой области [14].

1. Авторизация пользователя. Данный процесс характеризуется созданием нового аккаунта пользователя.

2. Просмотр и выбор кулинарного рецепта – следующий процесс, выполняющийся после регистрации пользователя в системе. На данном этапе происходит поиск рецепта в соответствии с установленными пользователем фильтрами.

3. Формирование комментария – на данном уровне приложения пользователю предоставляется возможность поставить оценку рецепту и оставить текстовый комментарий, который будет виден другим посетителям сервиса.

4. Сохранение рецепта. Данный процесс представляет собой добавление рецепта в закладки, чтобы предоставить пользователю легко повторно вернуться к найденному рецепту.

5. Получение списка созданных рецептов. На данном этапе проверяется роль пользователя для доступа к функции создания рецепта; если доступ разрешён, то автору предоставляется список созданных им кулинарных рецептов.

6. Изменение рецепта – следующий процесс, выполняющийся после получения списка созданных рецептов. Этот процесс характеризуется функционалом изменения созданного рецепта в соответствии с требованиями пользователя.

Диаграмма первого уровня контекстной диаграммы представлена на рисунке 2.2.

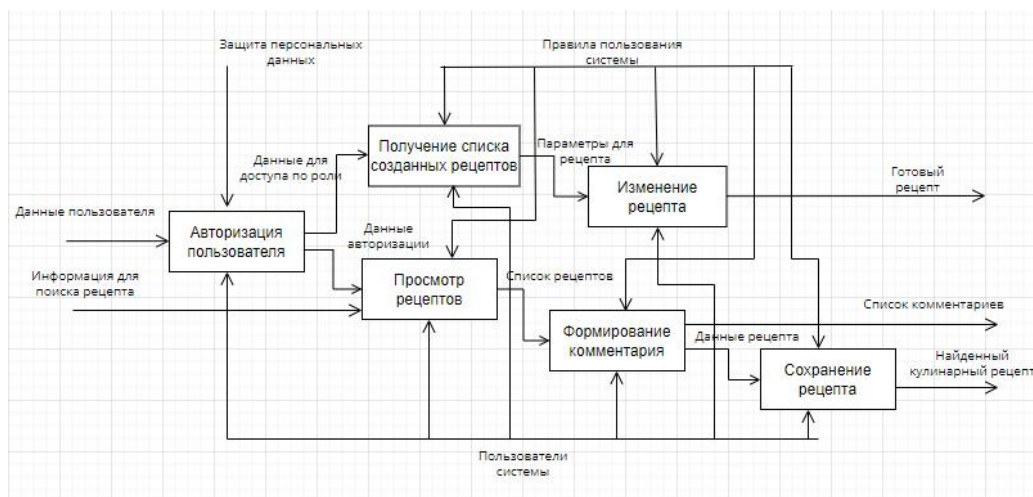


Рисунок 2.2 – Диаграмма декомпозиции IDEF0

2.2 Диаграмма последовательности

На диаграмме последовательности для системы «Электронная книга кулинарных рецептов» необходимо отразить следующие взаимодействия пользователей.

Для составления диаграммы последовательности необходимо выделить компоненты, которые будут отображены на диаграмме:

- Пользователь;
- Рецепты;
- Комментарии;
- Заметки.

Каждый объект имеет свою временную линию, изображаемую пунктиром под объектом. Сценарий действий включает в себя:

1. Пользователь авторизуется в системе, для дальнейшей работы.
2. Пользователь просматривает доступные кулинарные рецепты.
3. Пользователь оставляет комментарий и ставит оценку выбранному рецепту.
4. Пользователь просматривает комментарии, которые оставили другие пользователи системы.
5. Пользователь добавляет понравившийся кулинарный рецепт в заметки, сохраняет для повторного просмотра.
6. Пользователь получает список сохранённых рецептов.
7. Производится проверка доступа к роли «Автора».
8. Автор получает список созданных им кулинарных рецептов.
9. Автор выбирает или создаёт новый кулинарный рецепт и заполняет параметры: описание, название, категория и т.д.
10. Автор сохраняет и получает данные об отредактированном им кулинарном рецепте.

Диаграмма последовательности представлена на рисунке 2.3.

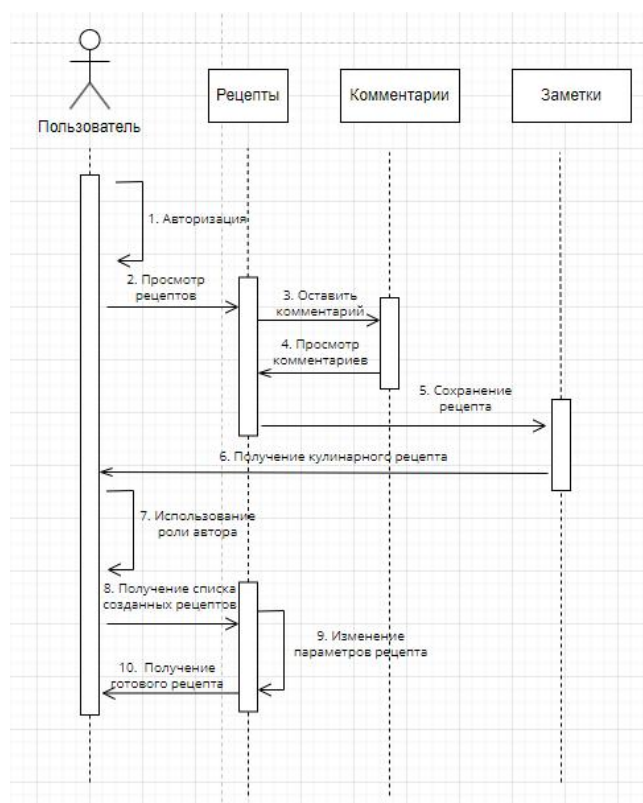


Рисунок 2.3 – Диаграмма последовательности

2.3 Проектирование базы данных

В процессе работы программного обеспечения происходит неоднократное обращение к базе данных для получения, обновления и внесения новых данных.

Для понимания взаимодействия данных, спроектируем логическую и физическую модели данных. Логическая модель представлена на рисунке 2.4.

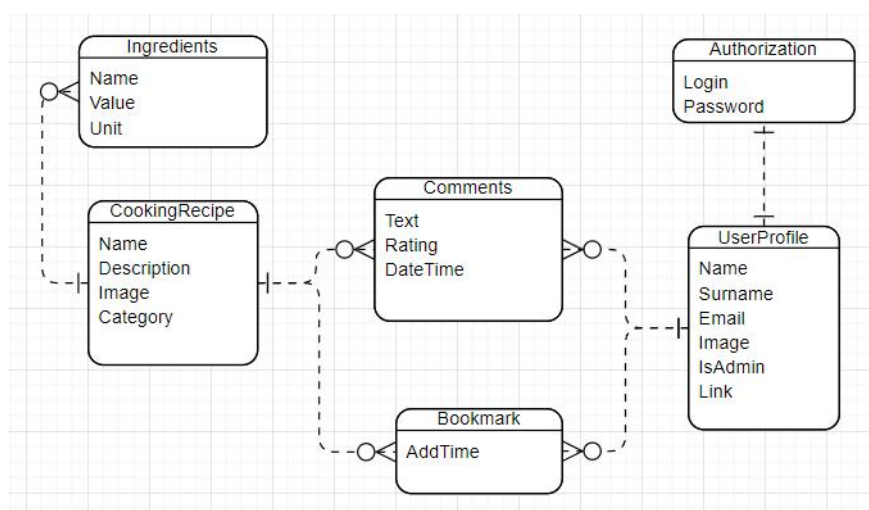


Рисунок 2.4 – Логическая модель базы данных

В реализованной базе данных представлены 6 сущностей:

- «UserProfile» – сущность пользователя системы;
- «Authorization» – сущность, используемая в процессе авторизации;
- «Comments» – сущность комментария пользователя на рецепте;
- «Bookmark» – сущность заметки кулинарного рецепта;
- «CookingRecipe» – сущность кулинарного рецепта;
- «Ingredients» – сущность, описывающая ингредиенты рецепта;

Физическая модель представлена на рисунке 2.5.

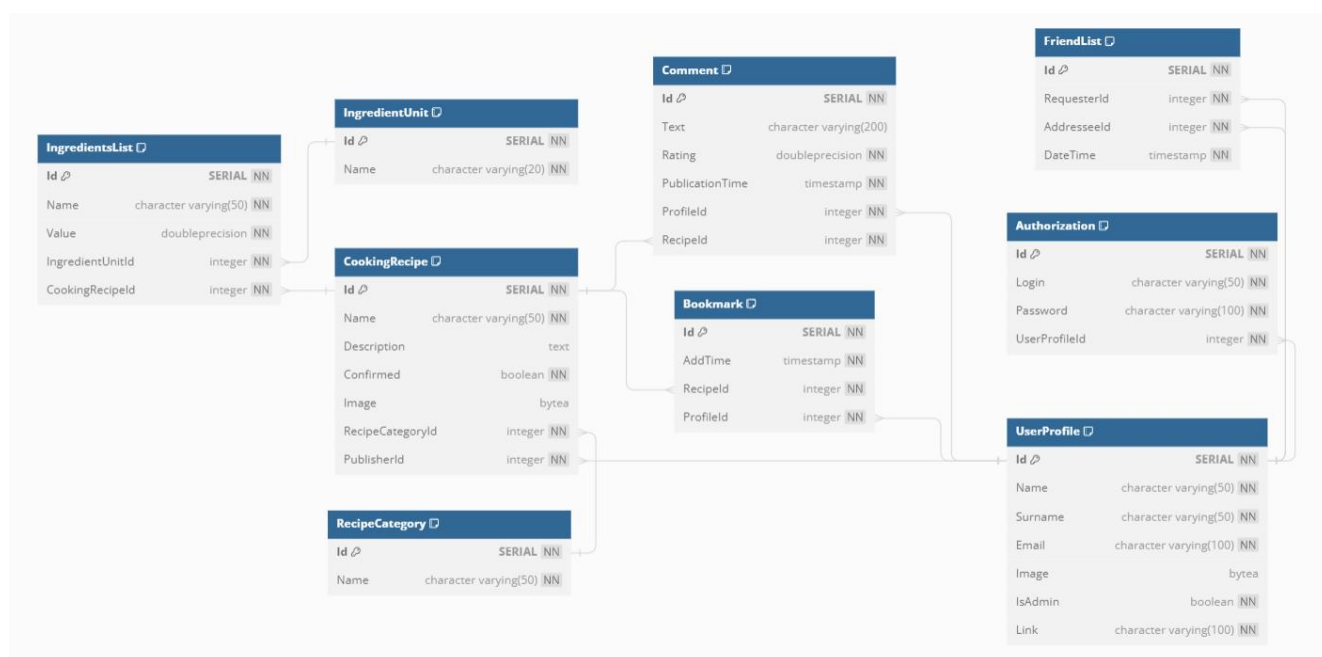


Рисунок 2.5 – Физическая модель базы данных

В результате проектирования получили логическую и физическую модель базы данных для системы «Электронная книга кулинарных рецептов». Эти модели предоставляют базовый функционал для управления пользователями, рецептами, ингредиентами и отзывами.

3 Разработка программного продукта

3.1 Аппаратное и программное обеспечение необходимое для реализации программного продукта

Программное обеспечение будет разработано с целью предоставления пользователям возможности просмотра кулинарных рецептов [15].

Информационная подсистема будет иметь следующие примерные технические требования для сервера баз данных и устройств пользователя.

Технические характеристики серверов:

- процессор: Intel® Xeon® E5450 2,60 ГГц;
- объем оперативной памяти: 32 Гб;
- жесткие диски: Общий объем памяти 1ТБ;
- сетевой адаптер: 250 Мбит/с;
- ОС: Windows 2016 Server.

Технические характеристики устройства пользователя:

- тактовая частота процессора – 1500 МГц;
- объем оперативной памяти – 2048 Мб;
- количество свободного места на диске – 512 Мб;
- ОС: Android версии 13.0 и выше с API 30.0

Для разработки продукта использовалось следующее программное обеспечение:

Microsoft Visual Studio 2022 — это современная интегрированная среда разработки (Integrated Development Environment, IDE), предлагающая набор инструментов для разработки программного обеспечения для Windows, Android, iOS [16].

.NET Multi-platform App UI или сокращенно MAUI представляет кроссплатформенный фреймворк от компании Microsoft для создания нативных мобильных и десктопных приложений с использованием языка программирования C# и языка разметки XAML. С помощью .NET MAUI можно

разрабатывать приложения под такие операционные системы как Android, iOS, macOS и Windows, используя при этом один и тот же код [17].

PostgreSQL – является одной из наиболее популярных систем управления реляционными базами данных. PostgreSQL использует реляционную модель. Реляционная модель предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта [18].

ASP.NET – это фреймворк для создания веб-приложений, разработанный компанией Microsoft. Он представляет собой часть технологии .NET и является одним из популярных инструментов для веб-разработки на платформе Windows. ASP.NET позволяет использовать различные языки программирования, такие как C#, Visual Basic.NET, JScript, и другие. Однако, C# является наиболее распространенным и рекомендуемым языком.

ASP.NET предоставляет модель программирования, основанную на событиях и объектно-ориентированной методологии. В основе лежит концепция кода на стороне сервера, что обеспечивает удобство разработки и обработку событий на сервере.

ASP.NET предоставляет широкий набор библиотек и компонентов для обработки различных аспектов веб-разработки, таких как работа с базами данных, аутентификация и авторизация, управление состоянием и многое другое. ASP.NET предоставляет возможности для создания веб-служб с использованием стандартов, таких как SOAP (Simple Object Access Protocol) и REST (Representational State Transfer). ASP.NET предоставляет разработчикам мощный инструментарий для построения современных и эффективных веб-приложений на платформе Windows [19].

Docker - это платформа для разработки, доставки и запуска приложений в контейнерах. Контейнер - это легковесный, автономный и исполняемый пакет, который включает в себя всё необходимое для запуска программы, включая код, среду выполнения, системные инструменты, библиотеки и зависимости. Docker

обеспечивает стандартизацию и упрощение процесса упаковки, доставки и выполнения приложений.

3.2 Реализация программного продукта

Разработка трёхуровневого приложения представляет собой структурный подход к организации архитектуры приложения, включающий три основных уровня: представление (presentation layer), бизнес-логика (business logic layer), и уровень данных (data layer). Этот подход позволяет достичь модульности, удобства сопровождения и повторного использования кода [20].

Трёхуровневая система представлена на рисунке 3.1.

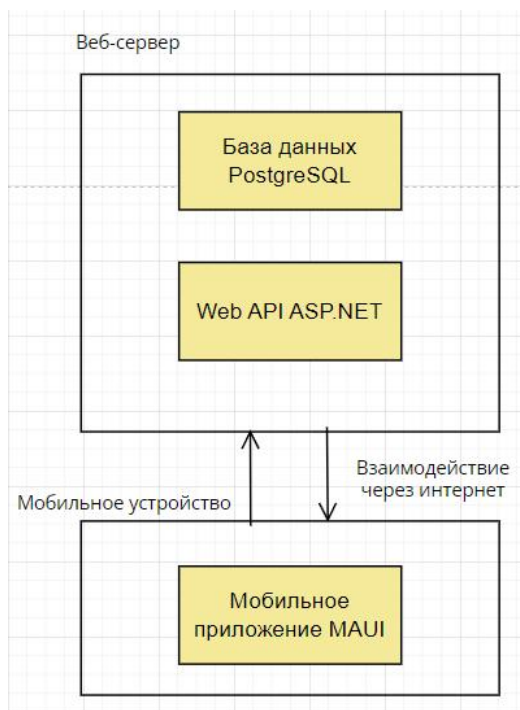


Рисунок 3.1 – Трёхуровневая система

Представление (Presentation Layer): этот уровень отвечает за отображение данных пользователю и взаимодействие с ним. Включает в себя пользовательский интерфейс (UI), который может быть веб-страницей, мобильным приложением, десктопным интерфейсом и т.д, который ответственен за сбор пользовательского ввода, отображение данных и передачу запросов на бизнес-логический уровень.

Бизнес-логика (Business Logic Layer): этот уровень содержит бизнес-правила и логику приложения. Здесь обрабатываются запросы от представления, выполняются бизнес-процессы, проводится валидация данных и принимаются решения в соответствии с правилами предметной области. Бизнес-логика отделена от представления и уровня данных, что обеспечивает независимость и удобство тестирования.

Уровень данных (Data Layer): этот уровень занимается доступом к данным и управлением хранилищем данных. Включает в себя работу с базой данных, файловой системой или другими источниками данных. Здесь реализуется взаимодействие с базой данных, выполнение запросов, а также маппинг объектов приложения на структуры данных в хранилище.

Применение трёхуровневой архитектуры является распространенным подходом при создании масштабируемых и легко сопровождаемых приложений.

Опишем основной функционал классов и методов, реализованных в информационной подсистеме. Разделение на классы необходимо для более удобной модульной разработки программного обеспечения.

Описание классов представлено в таблице 1.

Таблица 1 – Описание классов программы

Название	Описание
AuthorizationController	Класс-контроллер веб-службы, хранящий в себе функционал авторизации пользователя, а именно: входа в аккаунт и регистрации.
BookmarksController	Класс-контроллер веб-службы реализующий функционал добавления заметок пользователя для сохранения кулинарных рецептов.
FriendsListController	Класс-контроллер веб-службы реализующий функционал друзей: добавление, удаление и просмотр списка.
ProfileController	Класс-контроллер веб-службы, хранящий в себе функционал, связанный с пользователем системы: просмотр и редактирование профиля, смена пароля и т.д.
CommentsController	Класс-контроллер веб-службы, хранящий в себе функционал, добавления рейтинга кулинарных рецептов, а также комментариев пользователей: добавление и просмотр.

Название	Описание
CookingRecipeController	Класс-контроллер веб-службы реализующий функционал добавления, редактирования и просмотр списка кулинарных рецептов.
ApiAccessMiddleware	Класс промежуточного ПО, который выполняет проверку доступа к веб-службе, с использованием специального сгенерированного ключа.
CookingRecipeDbContext	Класс, который используется для доступа к базе данных, взаимодействию со всеми хранящимися таблицами.
CookingRecipe	Класс-модель, описывающий данные, которые хранятся в таблице кулинарными рецептами.
UserProfile	Класс-модель, описывающий данные, которые хранятся в таблице пользователей системы.
IngredientsList	Класс-модель, описывающий данные, которые хранятся в таблице с ингредиентами кулинарных рецептов.
ApiServiceCommunication	Класс определяющий сервис для взаимодействия с веб сервисами, расположенными на другом хосте, при помощи протокола HTTP.
NavigationService	Класс определяющий сервис для навигации между страницами с использованием стека навигации MAUI.
UserAuthorization	Класс определяющий сервис для взаимодействия с веб сервисом авторизации пользователя системы внутри приложения.
UserProfile	Класс определяющий сервис для взаимодействия с веб сервисом просмотра и редактирования информации о пользователях системы.
CookingRecipes	Класс-сервис, который отвечает за взаимодействие с веб сервисом кулинарных рецептов, для выполнения функций добавления, удаления и редактирования, а также поиска и просмотра списка рецептов и заметок.

Описание основных методов, реализованных в программе, представлено в таблице 2. Таблица содержит описание ключевых методов, реализованных в программе, исключая вспомогательные, а также методы, которые являются автоматически сгенерированными средой разработки. Например, обработчики событий или конструкторы окон, страниц.

Таблица 2 – Описание методов

Класс	Название метода	Описание
AuthorizationController	Task<IActionResult> LoginHandler([FromQuery] LoginRequestModel request)	Функция для авторизации пользователя в систему. Если данные корректны, возвращает токен пользователя, роль, а также время последней авторизации.
	Task<IActionResult> RegistrationHandler([FromBody] RegistrationRequestModel request)	Функция для регистрации нового пользователя системы, принимает контактные данные и конфигурацию профиля. Если данные корректны, создаётся профиль и возвращается токен и роль пользователя.
ProfileController	Task<IActionResult> EditProfileByTokenHandler([FromBody] EditProfileRequestModel request)	Функция редактирования профиля пользователя, в качестве входных данных принимает новые контактные данные пользователя.
	Task<IActionResult> GetProfileInfoByTokenHandler()	Функция для получения данных пользователя.
	Task<IActionResult> GetProfilesListHandler([FromQuery] GetProfilesListRequestModel request)	Функция для получения списка пользователей системы, в качестве входного параметра принимает настройки фильтрации.

Продолжение таблицы 2

Класс	Название метода	Описание
	Task<IActionResult> ChangePasswordByTokenHandler([FromBody] ChangePasswordRequestModel request)	Функция для изменения пароль пользователя, в качестве входного параметра принимает пару: старый и новый пароль.
CookingRecipeController	Task<IActionResult> AddRecipeByTokenHandler([FromBody] AddRecipeRequestModel request)	Функция для добавления кулинарного рецепта, принимает информацию о рецепте: категория, название, изображение и т.д.
	Task<IActionResult> DeleteRecipeHandler([FromQuery] DeleteRecipeRequestModel request)	Функция для удаления кулинарного рецепта по идентификатору.
	Task<IActionResult> EditRecipeHandler([FromBody] EditRecipeRequestModel request)	Функция для редактирования кулинарного рецепта.
	Task<IActionResult> GetRecipeHandler([FromQuery] GetRecipeRequestModel request)	Функция для получения данных о рецепте, на вход получает идентификатор рецепта.
	Task<IActionResult> GetRecipesListHandler([FromQuery] GetRecipesListRequestModel request)	Функция для получения списка кулинарных рецептов с использованием фильтра поиска.
CommentsList	Task<string> AddComment(RequestInfo<AddCom mentRequestModel> requestModel)	Функция для добавления отзыва о кулинарном рецепте.
	Task<string> DeleteComment(RequestInfo<Delete CommentRequestModel> Model)	Функция для удаления комментария и оценки кулинарного рецепта.

Класс	Название метода	Описание
	Task<GetCommentsResponseModel> GetCommentsList(RequestInfo<GetRecipeCommentsRequestModel> model)	Функция для получения списка комментариев пользователей по идентификатору кулинарного рецепта.

3.3 Пример работы программного продукта

Программный продукт встречает пользователя окном авторизации, где предлагается войти в профиль или зарегистрировать новый аккаунт (рисунок 3.2).

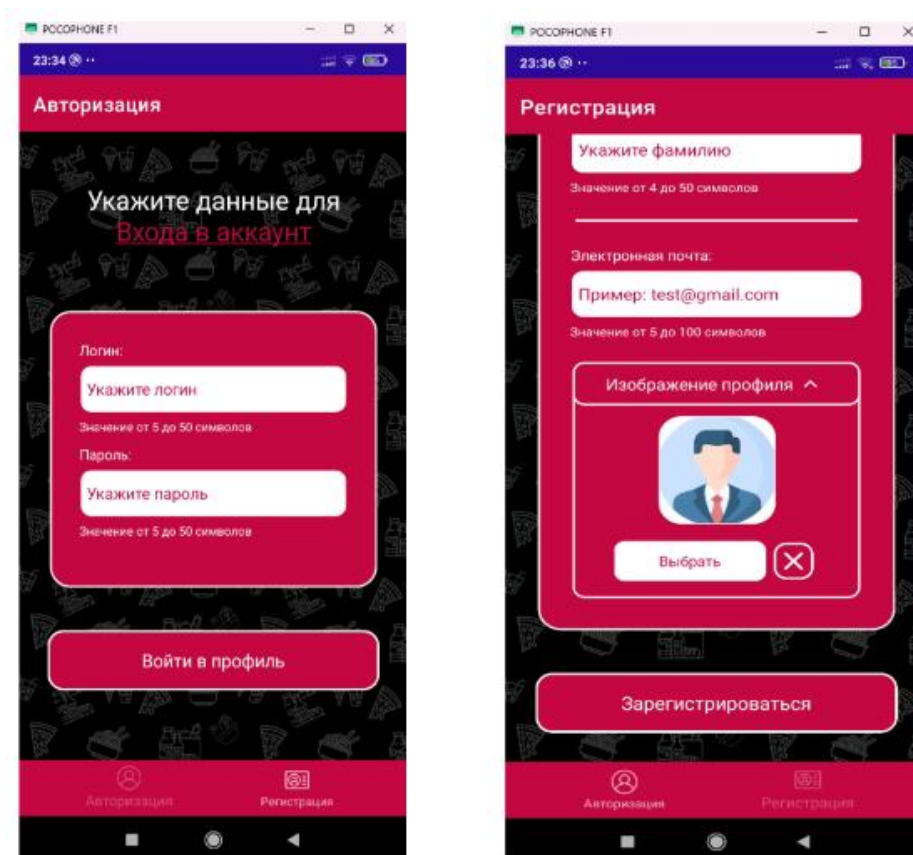


Рисунок 3.2 – Страницы авторизации

После авторизации в системе пользователь перенаправляется на страницу просмотра кулинарных рецептов, на которой человек может с использованием фильтров и функции сортировки подобрать себе необходимую запись.

Работа страницы просмотра списка рецептов показана на рисунке 3.3.



Рисунок 3.3 – Страница просмотра рецептов

Функция поиска необходимого рецепта с применением фильтров и параметров сортировки продемонстрирована на рисунке 3.4.



Рисунок 3.4 – Страница просмотра рецептов

После чего кликом по элементу списка, можно перейти на страницу просмотра конкретного кулинарного рецепта, на котором можно посмотреть описание, список требуемых ингредиентов, данные об авторе, а также открывается возможность просмотра комментариев и сохранения рецепта в заметки профиля пользователя (рисунок 3.5).

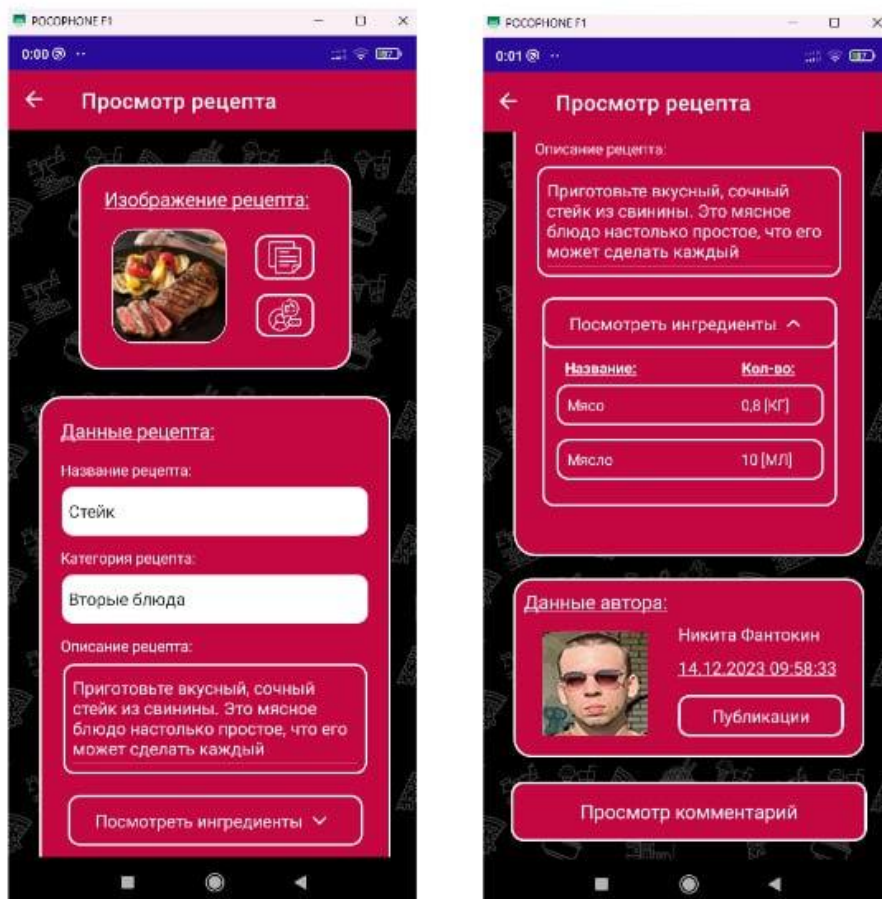


Рисунок 3.5 – Страница просмотра кулинарного рецепта

Если у пользователя появляется необходимость для сохранения рецепта в закладки для просмотра в более позднее время, то после нажатия на кнопку возле изображения рецепта появится сообщение и рецепт будет сохранен (рисунок 3.6).

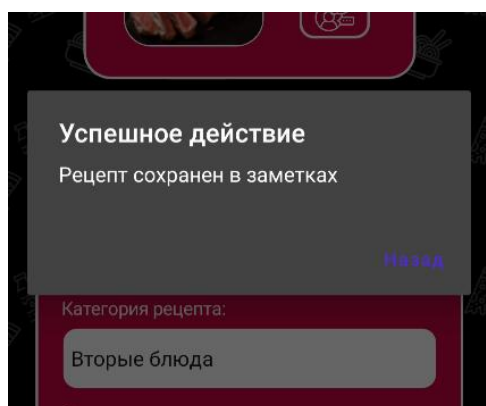


Рисунок 3.6 – Процесс сохранения рецепта в закладки

Теперь нажмём на кнопку «Просмотр комментариев», после чего пользователю будет доступна возможность оставить комментарий и оценку о кулинарном рецепте, а также просмотреть комментарии и оценки других пользователей (рисунок 3.7).

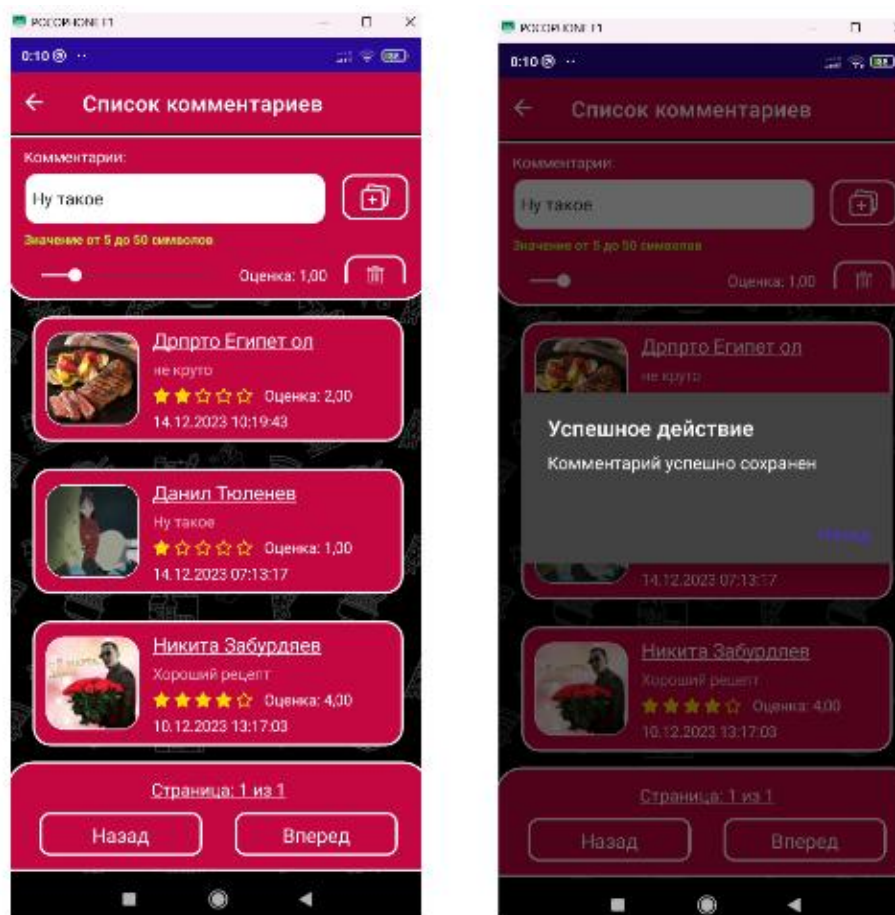


Рисунок 3.7 – Страница просмотра комментариев рецепта

Также со страницы просмотра рецепта можно просмотреть данные об авторе рецепта. При нажатии на кнопку «Публикации» на карточке автора, пользователь будет перенаправлен на страницу просмотра списка рецептов, которые были загружены данным автором, что продемонстрировано на рисунке 3.8.



Рисунок 3.8 – Страница автора рецепта

На главной странице можно перейти на вкладку с сохранёнными заметками, на которой отображается список добавленных пользователем кулинарных рецептов (рисунок 3.9). Также можно открыть меню навигации для перехода к другим разделам приложения: «Рецепты» (используется для поиска и сохранения рецептов), «Профиль» (используется для редактирования профиля пользователя, а также для просмотра списка друзей), «Редакция» (доступен только пользователям с ролью автора, и применяется для создания, удаления и редактирования рецептов) (рисунок 3.10).



Рисунок 3.9 – Страница списка заметок

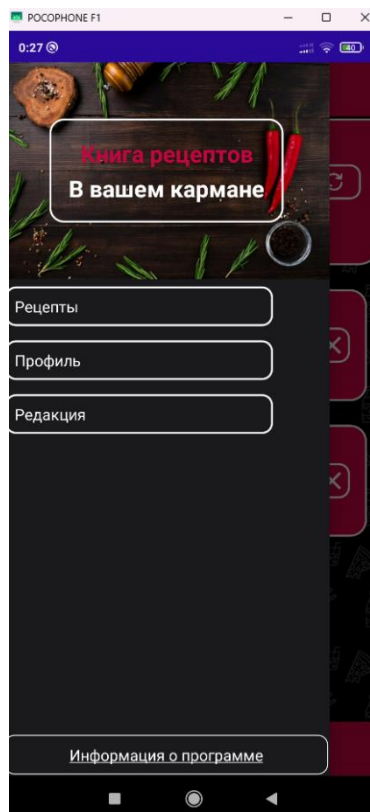


Рисунок 3.10 – Меню навигации

При переходе к разделу «Профиль» пользователю откроется страница с данными о его аккаунте. На данной странице можно поменять изображение профиля, настроить контактные данные, изменить пароль, или вовсе удалить профиль из системы приложения (рисунок 3.11).

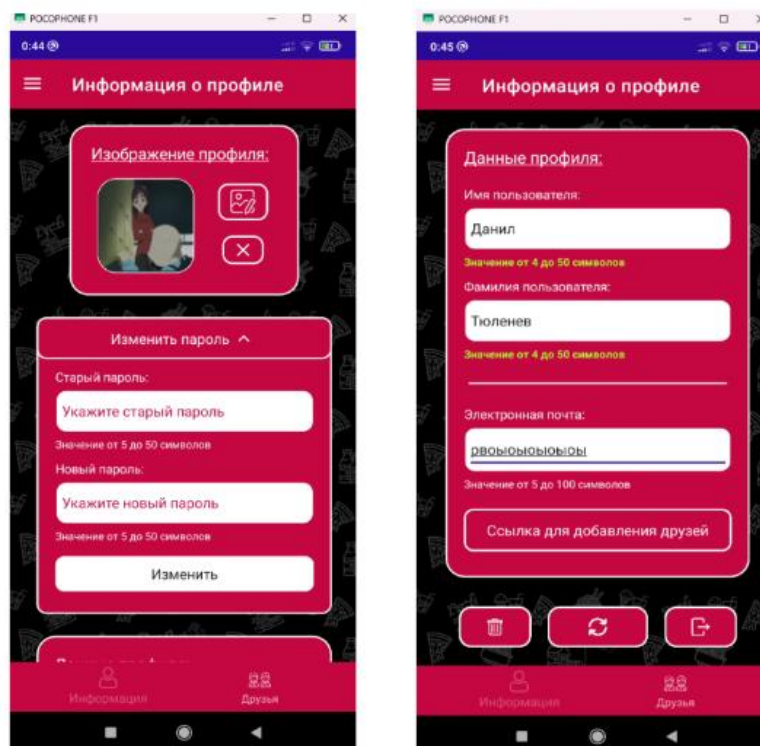


Рисунок 3.11 – Страница профиля пользователя

При редактировании данных через текстовые поля, также производится валидация данных, и при некорректном вводе отобразится сообщение об ошибке. Также через эту страницу можно поделиться ссылкой на профиль, для добавления в друзья или выйти из аккаунта.

Раздел «Редакция» доступен только пользователям с ролью «Автора», при переходе на него открывается страница, на которой открывается список с добавленными пользователем рецептами. У автора имеется возможность просмотра рецептов, добавление, удаление и редактирование уже существующих (рисунок 3.12).

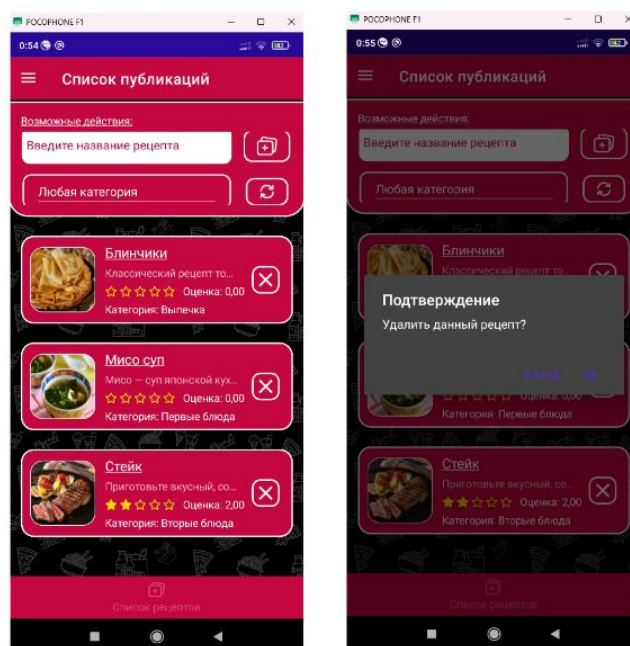


Рисунок 3.12 – Страница профиля пользователя

При выборе элемента списка, открывается страница для настройки рецепта: изменение категории, изображения, названия и описания, а также редактирования списка ингредиентов (рисунок 3.13).

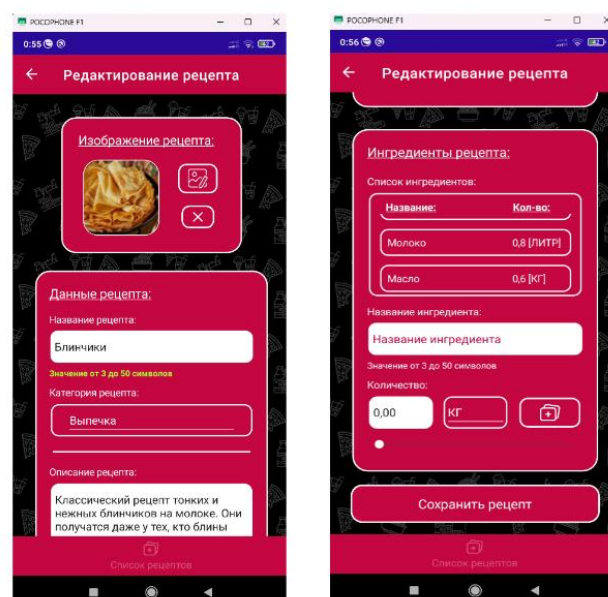


Рисунок 3.13 – Страница редактирования рецепта

Весь основной функционал программного продукта был рассмотрен, что позволяет утверждать о достижении всех поставленных целей.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта были изучены и применены основные инструменты и методы построения пользовательского интерфейса с использованием технологии MAUI. В результате был разработан программный продукт, функционал которого соответствует требованиям и потребностям людей, которые занимаются поиском кулинарных рецептов. Полученный программный продукт обеспечивает удобство и эффективность.

В процессе выполнения курсового проекта были пройдены такие этапы как:

- анализ предметной области и аналогов;
- моделирование информационной системы с помощью унифицированного языка моделирования UML;
- спроектирована и разработана база данных;
- разработано программное обеспечение.

Благодаря анализу и моделированию, разработка программного обеспечения проходила планомерно, по подготовленному алгоритму действий. Благодаря этому все задачи, поставленные при курсовом проектировании были выполнены.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Шилдт Герберт C# 4.0 полное руководство: Переведено с английского – М.: ООО “И.Д. Вильямс”, 2011
2. Савельев А. О. Разработка приложений для мобильных устройств на платформе Windows Mobile: Электронная книга / А.О. Савельев, Д.В. Рудаков — 2-е изд. — М.: ИНТУИТ, 2016. — 215 с.
3. Киммел, Пол UML. Основы визуального анализа и проектирования / Пол Киммел. - М.: НТ Пресс, 2008. - 272 с.
4. Зайцев М.Г. Объектно-ориентированный анализ и программирование /М.Г. Зайцев - учебное пособие - Н:НГТУ, 2017. - 84 с
5. Самохвалов Э.Н. Введение в проектирование и разработку приложений на языке программирования C# / Э.Н. Самохвалов, Г.И. Ревунков, Ю.Е. Гапанюк. - учебное пособие. - М.:МГТУ 2018. - 244с.
6. Калянова Г.Н. Структурные модели бизнеса: DFD-технологии/ Г.Н. Калянова -М.: Финансы и статистика, 2009. - 256 с.
7. Розенберг Д. Применение объектного моделирования с использованием UML и анализ прецедентов / Д. Розенберг, К. Скотт Пер. с англ. - М.: ДМК Пресс. - 160 с
8. Вендров А.М. Объектно-ориентированный анализ и проектирование с использованием языка UML / А.М. Вендров, В.В. Малышко – М.:АСТ, 2016 - 139с. Чарльз Петцольд. Программирование для Microsoft Windows на C#. В 2-х томах. Том 1./ Пер. с англ. – М.: Издательско-торговый дом “Русская редакция”, 2002. – 576 с.
9. Рассел, Джесси Интернет-магазин / Джесси Рассел. - М.: VSD, 2012. - 489 с.
10. Шилдт Герберт C# 4.0 полное руководство: Переведено с английского – М.: ООО “И.Д. Вильямс”, 2011

11. Курипта О.В. Основы программирования и алгоритмизации: практикум / О.В.Курипта, О.В. Минакова, Д.К. Проскурин; Воронежский ГАСУ. – Воронеж, 2015. – 132 с.
12. Соловьев, Д. Интернет-магазин без правил / Д. Соловьев, А. Писарев. - М.: Питер, 2013. - 907 с.
13. Фельдман, Я. А. Создаем информационные системы (+ CD-ROM) / Я.А. Фельдман. - М.: Солон-Пресс, 2007. - 120 с.
14. Рассел, Джесси Интернет-магазин / Джесси Рассел. - М.: VSD, 2012. - 489 с.
15. Салбер, Алена. Как открыть Интернет-магазин / Алена Салбер. - М.: Омега-Л, 2016. - 320 с.
16. Смит Дж. П. C50 Entity Framework Core в действии / пер. с англ. Д. А. Беликова. – М.: ДМК Пресс, 2022. – 690 с.: ил.
17. Документация по языку C# – Электрон. дан. – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/csharp>
18. Документация по технологии MAUI – Электрон. дан. – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/maui/what-is-maui?view=net-maui-8.0>
19. Документация по MAUI – Электрон. дан. – Режим доступа: <https://metanit.com/sharp/maui/1.1.php>
20. Центр документации Entity Framework – Электрон. дан. – Режим доступа: <https://learn.microsoft.com/ru-ru/ef/>

ПРИЛОЖЕНИЕ А

Файл «MauiProgram.cs»

```
using CommunityToolkit.Maui;
using MauiLabs.View.Commons.ViewModels;
using MauiLabs.View.Pages;
using MauiLabs.View.Services;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Options;
using System.Reflection;
using System.Resources;

namespace MauiLabs.View
{
    public static class MauiProgram : object
    {
        public static MauiApp CreateMauiApp()
        {
            var assembly = Assembly.GetExecutingAssembly();
            var builder = MauiApp.CreateBuilder();
            builder
                .UseMauiApp<CookingRecipeApp>()
                .UseMauiCommunityToolkit()
                .ConfigureFonts(fonts =>
                {
                    fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");
                    fonts.AddFont("OpenSans-Semibold.ttf", "OpenSansSemibold");
                });
            using (var stream =
assembly.GetManifestResourceStream("MauiLabs.View.appsettings.json"))
            {
                var config = new ConfigurationBuilder().AddJsonStream(stream);
                builder.Configuration.AddConfiguration(config.Build());
            }

            builder.Logging.AddDebug();
            builder.Services.AddViewServices(builder.Configuration).Wait();

            builder.Services.AddViewPages(builder.Configuration).Wait();
            builder.Services.AddViewModels(builder.Configuration).Wait();
            return builder.Build();
        }
    }
}
```

Файл «CookingRecipeShell.xaml.cs»

```
using MauiLabs.View.Pages;
using MauiLabs.View.Pages.ProfilePages;
using MauiLabs.View.Pages.RecipePages;
using System.Net;
using System.Runtime.CompilerServices;

namespace MauiLabs.View
{
    public partial class CookingRecipeShell : Shell
    {
        public static readonly BindableProperty MyTabIsEnabledProperty =
BindableProperty.Create(
            "MyTabIsEnabled", typeof(bool),
            typeof(CookingRecipeShell), true, propertyChanged: TabIsEnabledChangedHandler);

        public static readonly BindableProperty IsAdminProperty = BindableProperty.Create(
```

```

        "isAdmin", typeof(bool),
        typeof(CookingRecipeShell), true, propertyChanged: TabIsEnabledChangedHandler);

    public bool MyTabIsEnabled
    {
        get => (bool)this.GetValue(MyTabIsEnabledProperty); set =>
this.SetValue(MyTabIsEnabledProperty, value);
    }
    public bool IsAdmin { get => (bool)this.GetValue(IsAdminProperty); set =>
this.SetValue(IsAdminProperty, value); }
    protected static async void TabIsEnabledChangedHandler(BindableObject @object,
object oldValue, object newValue)
    {
        await Console.Out.WriteLineAsync($"TabIsEnabled Value: {newValue}");
    }
    public static void SetMyTabIsEnabled(bool enabled = true)
    {
        if (Shell.Current is CookingRecipeShell recipeShell)
        {
            recipeShell.Dispatcher.Dispatch(() => recipeShell.MyTabIsEnabled = enabled);
        }
    }
    public static void SetIsAdmin(bool value = false)
    {
        if (Shell.Current is CookingRecipeShell recipeShell)
        {
            recipeShell.Dispatcher.Dispatch(() => recipeShell.IsAdmin = value);
        }
    }
    public static void SetTabBarVisibility(Page page, bool value)
    {
        if (Shell.Current.CurrentItem.Items.Count > 1)
        {
            Shell.SetTabBarIsVisible(page, value);
            var currentSection = Shell.Current.CurrentItem.CurrentItem;

            Shell.Current.CurrentItem.CurrentItem = null;
            Shell.Current.CurrentItem.CurrentItem = currentSection;
        }
    }
    public CookingRecipeShell() : base()
    {
        this.InitializeComponent();

        var aboutLinkTap = new TapGestureRecognizer() { NumberOfTapsRequired = 1 };
        aboutLinkTap.Tapped += this.AboutLinkTapHandler;

        this>AboutLabel.GestureRecognizers.Add(aboutLinkTap);
    }
    protected virtual async void AboutLinkTapHandler(object sender, TappedEventArgs
args)
    {
        var pageUri = new Uri(@"https://github.com/mo0nchild/cs-maui-labs");

        try { await Browser.Default.OpenAsync(pageUri,
BrowserLaunchMode.SystemPreferred); }
        catch (Exception errorInfo)
        {
            await this.DisplayAlert("Произошла ошибка", errorInfo.Message, "Назад");
        }
    }
    protected override void OnAppearing() => base.OnAppearing();
    protected override void OnDisappearing() => base.OnDisappearing();
}
}

```

Файл «CookingRecipeShell.xaml»

```
<?xml version="1.0" encoding="UTF-8" ?>
<Shell
  x:Class="MauiLabs.View.CookingRecipeShell"
  xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
  xmlns:recipePages="clr-namespace:MauiLabs.View.Pages.RecipePages"
  xmlns:profilePages="clr-namespace:MauiLabs.View.Pages.ProfilePages"
  Shell.BackgroundColor="{x:StaticResource FirstColor}" Shell.TabBarIsVisible="True"
  FlyoutBackgroundColor="{x:StaticResource SecondColor}" x:Name="this">
  <Shell.Resources>
    <ResourceDictionary>
      <Style x:Key="FlyoutHeaderLabel" TargetType="Label">
        <Setter Property="TextColor" Value="{x:StaticResource FirstColor}"/>
        <Setter Property="HorizontalOptions" Value="Center"/>
        <Setter Property="VerticalOptions" Value="Center"/>
        <Setter Property="FontSize" Value="24"/>
        <Setter Property="FontAttributes" Value="Bold"/>
      </Style>
      <Style x:Key="FlyoutHeaderTextBorder" TargetType="Border">
        <Setter Property="Stroke" Value="White"/>
        <Setter Property="Padding" Value="20"/>
        <Setter Property="HorizontalOptions" Value="Center"/>
        <Setter Property="VerticalOptions" Value="Center"/>
        <Setter Property="BackgroundColor" Value="Transparent"/>
        <Setter Property="StrokeShape">
          <Setter.Value>
            <RoundRectangle CornerRadius="14" />
          </Setter.Value>
        </Setter>
        <Setter Property="StrokeThickness" Value="2"/>
      </Style>
    </ResourceDictionary>
  </Shell.Resources>

  <Shell.FlyoutHeader>
    <Grid ColumnDefinitions="*" RowDefinitions="*">
      <Image Aspect="AspectFill" Source="flyouthead.jpg" />
      <Border Style="{x:StaticResource FlyoutHeaderTextBorder}">
        <StackLayout Orientation="Vertical" Spacing="4">
          <Label Text="Книга рецептов" Style="{x:StaticResource
FlyoutHeaderLabel}" />
          <Label Text="В вашем кармане" TextColor="White" Style="{x:StaticResource
FlyoutHeaderLabel}" />
        </StackLayout>
      </Border>
    </Grid>
  </Shell.FlyoutHeader>

  <Shell.ItemTemplate>
    <DataTemplate>
      <FlexLayout Direction="Column" JustifyContent="Center" AlignItems="Stretch"
        Margin="0, 5">
        <Border BackgroundColor="Transparent" Stroke="White" StrokeThickness="2"
          Padding="10" WidthRequest="280">
          <Border.StrokeShape>
            <RoundRectangle CornerRadius="10" />
          </Border.StrokeShape>
          <Label Text="{Binding Title}" FontSize="16" TextColor="White"/>
        </Border>
      </FlexLayout>
    </DataTemplate>
  </Shell.ItemTemplate>

  <Shell.FlyoutFooter>
```

```

        <Grid ColumnDefinitions="*" RowDefinitions="*">
            <Border Stroke="White" StrokeThickness="1" Padding="10"
BackgroundColor="Transparent">
                <Border.StrokeShape>
                    <RoundRectangle CornerRadius="10" />
                </Border.StrokeShape>
                <Label x:Name="AboutLabel" Text="Информация о программе"
TextDecorations="Underline" FontSize="16"
                TextColor="White" HorizontalOptions="Center" VerticalOptions="Center"/>
            </Border>
        </Grid>
    </Shell.FlyoutFooter>

    <TabBar Route="authorization" Shell.FlyoutItemIsVisible="False" >
        <Tab Route="loginTab" Title="Авторизация" Icon="login.png">
            <ShellContent Route="loginPage" ContentTemplate="{DataTemplate
profilePages:AuthorizationPage}"/>
        </Tab>
        <Tab Route="registrationTab" Title="Регистрация" Icon="registration.png">
            <ShellContent Route="registrationPage" ContentTemplate="{DataTemplate
profilePages:RegistrationPage}"/>
        </Tab>
    </TabBar>
    <FlyoutItem Route="recipes" Title="Рецепты" >
        <Tab Route="listTab" Title="Список рецептов" Icon="cooking.png">
            <ShellContent Route="listPage" ContentTemplate="{DataTemplate
recipePages:RecipesListPage}"/>
        </Tab>
        <Tab Route="bookmarksTab" Title="Список заметок" Icon="recipelist.png">
            <ShellContent Route="bookmarksPage" ContentTemplate="{DataTemplate
profilePages:BookmarksListPage}"/>
        </Tab>
    </FlyoutItem>
    <FlyoutItem Route="profile" Title="Профиль">
        <Tab Route="infoTab" Title="Информация" Icon="user.png">
            <ShellContent Route="infoPage" ContentTemplate="{DataTemplate
profilePages:ProfileInfoPage}" />
        </Tab>
        <Tab Route="friendsTab" Title="Друзья" Icon="friend.png">
            <ShellContent Route="friendsPage" ContentTemplate="{DataTemplate
profilePages:FriendsListPage}" />
        </Tab>
    </FlyoutItem>
    <FlyoutItem Route="publisher" Title="Редакция" IsVisible="{Binding IsAdmin,
Source={x:Reference this}}">
        <Tab Route="creatingTab" Title="Список рецептов" Icon="addicon.png">
            <ShellContent Route="creatingPage" ContentTemplate="{DataTemplate
recipePages:PublishedListPage}"/>
        </Tab>
    </FlyoutItem>
</Shell>

```

Файл «CookingRecipeApp.xaml»

```

<?xml version = "1.0" encoding = "UTF-8" ?>
<Application xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
xmlns:local="clr-namespace:MauiLabs.View"
x:Class="MauiLabs.View.CookingRecipeApp">
    <Application.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <!--<ResourceDictionary Source="Resources/Styles/Colors.xaml" />
                <ResourceDictionary Source="Resources/Styles/Styles.xaml" />-->
            </ResourceDictionary.MergedDictionaries>
            <ResourceDictionary>
                <Color x:Key="FirstColor">#C3073F</Color>
            </ResourceDictionary>
        </ResourceDictionary>
    </Application.Resources>

```

```

        <Color x:Key="SecondColor">#1A1A1D</Color>
    </ResourceDictionary>
</ResourceDictionary.MergedDictionaries>
</ResourceDictionary>
</Application.Resources>
</Application>

```

Файл «CookingRecipeApp.xaml.cs»

```

using MauiLabs.View.Pages.ProfilePages;
using MauiLabs.View.Pages.RecipePages;

namespace MauiLabs.View
{
    public partial class CookingRecipeApp : Application
    {
        public CookingRecipeApp(IServiceProvider serviceProvider) : base()
        {
            this.InitializeComponent();
            this.MainPage = new CookingRecipeShell();
        }
        protected override Window CreateWindow(IActivationState activationState)
        {
            var applicationWindow = base.CreateWindow(activationState);
            if (DeviceInfo.Platform == DevicePlatform.WinUI)
            {
                (applicationWindow.MaximumWidth, applicationWindow.MaximumHeight) = (500,
750);
                (applicationWindow.Width, applicationWindow.Height) = (500, 750);
            }
            return applicationWindow;
        }
    }
}

```

Файл «DependencyInjection.cs»

```

using MauiLabs.View.Services.ConfigureOptions;
using MauiLabs.View.Services.Implements;
using MauiLabs.View.Services.Interfaces;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Options;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MauiLabs.View.Services
{
    using WebApiOptions = ConfigureWebApi.WebApiOptions;
    public static class DependencyInjection : object
    {
        public static Task<IServiceCollection> AddViewServices(this IServiceCollection
collection, IConfiguration configuration)
        {
            collection.ConfigureOptions<ConfigureWebApi>();
            var clientOptions =
collection.BuildServiceProvider().GetService<IOptions<WebApiOptions>>().Value;

            collection.AddHttpClient(clientOptions.ApiClient, options =>
            {
                options.DefaultRequestHeaders.Add("ApiKey", clientOptions.ApiKey);
                options.Timeout = TimeSpan.FromMilliseconds(16000);
                options.BaseAddress = new Uri(clientOptions.BaseUrl);
            }

```

```

    });
    collection.AddTransient<IApiServiceCommunication, ApiServiceCommunication>();
    collection.AddTransient<INavigationService, NavigationService>();

    collection.AddTransient<IUserAuthorization, UserAuthorization>();
    collection.AddTransient<IUserProfile, UserProfile>();
    collection.AddTransient<IFriendsList, FriendsList>();

    collection.AddTransient<ICookingRecipes, CookingRecipes>();
    collection.AddTransient<ICommentsList, CommentsList>();
    collection.AddTransient<IBookmarksList, BookmarksList>();
    return Task.FromResult(collection);
}
}
}

```

Файл «ApiServiceCommunication.cs»

```

using MauiLabs.View.Services.ApiModels.Commons;
using MauiLabs.View.Services.Commons;
using MauiLabs.View.Services.ConfigureOptions;
using MauiLabs.View.Services.Interfaces;
using Microsoft.Extensions.Options;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Net.Http.Json;
using System.Text;
using System.Text.Json;
using System.Text.Json.Serialization;
using System.Threading.Tasks;

namespace MauiLabs.View.Services.Implements
{
    using WebApiOptions = ConfigureWebApi.WebApiOptions;
    public partial class ApiServiceCommunication : IApiServiceCommunication
    {
        protected internal readonly IHttpClientFactory httpClientFactory = default!;
        protected internal readonly WebApiOptions webApiOptions = default!;

        public virtual Uri BaseAddress { get; protected set; } = default!;

        private readonly JsonSerializerOptions jsonOptions = new
        JsonSerializerOptions(JsonSerializerDefaults.Web);
        public ApiServiceCommunication(IHttpClientFactory httpFactory,
        IOptions<WebApiOptions> options) : base()
        {
            (this.httpClientFactory, this.webApiOptions) = (httpFactory, options.Value);

            using var httpClient =
        this.httpClientFactory.CreateClient(this.webApiOptions.ApiClient);
            this.BaseAddress = httpClient.BaseAddress;
        }
        public virtual void Dispose() { }
        public virtual async Task<TResponse>
        SendRequestAsync<TResponse>(HttpRequestMessage requestMessage,
        CancellationToken cancelToken, Func<HttpContent, Task<TResponse>>
        contentTransform)
        {
            using var httpClient =
        this.httpClientFactory.CreateClient(this.webApiOptions.ApiClient);
            var responseTask = httpClient.SendAsync(requestMessage,
        HttpCompletionOption.ResponseContentRead, cancelToken);

```



```

        if (await responseTask.WaitUntil(cancelToken)) throw new
TaskCanceledException();

        using var response = await responseTask;
        if (response.StatusCode != HttpStatusCode.OK)
        {
            if (response.StatusCode != HttpStatusCode.BadRequest) throw new
Exception("Не удается подключиться к серверу");

            var responseContent = await response.Content.ReadAsStringAsync();
            var errorMessage =
JsonConvert.DeserializeObject<ProblemDetails>(responseContent);

            throw new ViewServiceException(errorMessage?.Detail ?? errorMessage?.Title,
response.StatusCode);
        }
        return await contentTransform.Invoke(response.Content);
    }

    public virtual async Task<string> AddDataToServer<TRequest>(string requestPath,
RequestInfo<TRequest> model)
    {
        where TRequest : class
        {
            using var request = model.CreateRequestMessage(HttpMethod.Post,
$"{{this.BaseAddress}}{requestPath}");
            return await this.SendRequestAsync(request, model.CancelToken, async (content)
=> await content.ReadAsStringAsync());
        }

        public virtual async Task<string> DeleteDataFromServer<TRequest>(string
requestPath, RequestInfo<TRequest> model)
        {
            where TRequest : class
            {
                using var request = model.CreateRequestMessage(HttpMethod.Delete,
$"{{this.BaseAddress}}{requestPath}");
                return await this.SendRequestAsync(request, model.CancelToken, async (content)
=> await content.ReadAsStringAsync());
            }

            public virtual async Task<TResponse> GetDataFromServer<TRequest, TResponse>(string
requestPath, RequestInfo<TRequest> model)
            {
                where TRequest : class
                {
                    using var request = model.CreateRequestMessage(HttpMethod.Get,
$"{{this.BaseAddress}}{requestPath}");
                    return await this.SendRequestAsync(request, model.CancelToken, async (content)
=>
                    {
                        return JsonConvert.DeserializeObject<TResponse>(await
content.ReadAsStringAsync());
                    });
                }

                public virtual async Task<string> UpdateDataToServer<TRequest>(string requestPath,
RequestInfo<TRequest> model,
bool fullUpdate = true) where TRequest : class
                {
                    using var request = model.CreateRequestMessage(fullUpdate ? HttpMethod.Put :
HttpMethod.Patch,
$"{{this.BaseAddress}}{requestPath}");
                    return await this.SendRequestAsync(request, model.CancelToken, async (content)
=> await content.ReadAsStringAsync());
                }
            }

            public static class TaskCancellationExtension : object
            {

```

```

        public static async Task<bool> WaitUntil(this Task thisTask, CancellationToken
token)
        {
            while (!thisTask.IsCompleted && !token.IsCancellationRequested)
            {
                await Task.Delay(100);
            }
            return thisTask.Status != TaskStatus.RanToCompletion &&
token.IsCancellationRequested;
        }
    }
}

```

Файл «BookmarksList.cs»

```

using MauiLabs.View.Services.ApiModels.ProfileModels.Bookmarks.Requests;
using MauiLabs.View.Services.ApiModels.ProfileModels.Bookmarks.Responses;
using MauiLabs.View.Services.Commons;
using MauiLabs.View.Services.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MauiLabs.View.Services.Implements
{
    public partial class BookmarksList : IBookmarksList
    {
        protected internal readonly IApiServiceCommunication apiService = default!;
        public BookmarksList(IApiServiceCommunication apiService) : base() =>
this.apiService = apiService;

        public virtual async Task<string> AddBookmark(RequestInfo<AddBookmarkRequestModel>
requestModel)
        {
            var requestPath = string.Format("cookingrecipes/bookmarks/addbytoken");
            return await
this.apiService.AddDataToServer<AddBookmarkRequestModel>(requestPath, requestModel);
        }
        public virtual async Task<string>
DeleteBookmark(RequestInfo<DeleteBookmarkRequestModel> requestModel)
        {
            var requestPath = string.Format("cookingrecipes/bookmarks/deletebytoken");
            return await
this.apiService.DeleteDataFromServer<DeleteBookmarkRequestModel>(requestPath,
requestModel);
        }
        public virtual async Task<GetBookmarksResponseModel>
GetBookmarksById(RequestInfo<GetBookmarksByIdRequestModel> model)
        {
            var requestPath = string.Format("cookingrecipes/bookmarks/getlist");
            return await this.apiService.GetDataFromServer<GetBookmarksByIdRequestModel,
GetBookmarksResponseModel>(requestPath, model);
        }
        public virtual async Task<GetBookmarksResponseModel>
GetBookmarks(RequestInfo<GetBookmarksRequestModel> model)
        {
            var requestPath = string.Format("cookingrecipes/bookmarks/getlistbytoken");
            return await this.apiService.GetDataFromServer<GetBookmarksRequestModel,
GetBookmarksResponseModel>(requestPath, model);
        }
    }
}

```

Файл «CommentsList.cs»

```

using MauiLabs.View.Services.ApiModels.RecipeModels.Comments.Requests;
using MauiLabs.View.Services.ApiModels.RecipeModels.Comments.Responses;
using MauiLabs.View.Services.Commons;
using MauiLabs.View.Services.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MauiLabs.View.Services.Implements
{
    public partial class CommentsList : ICommentsList
    {
        protected internal readonly IApiServiceCommunication apiService = default!;
        public CommentsList(IApiServiceCommunication apiService) : base() =>
        this.apiService = apiService;

        public virtual async Task<string> AddComment(RequestInfo<AddCommentRequestModel>
requestModel)
        {
            var requestPath = string.Format("cookingrecipes/comments/addbytoken");
            return await
this.apiService.AddDataToServer<AddCommentRequestModel>(requestPath, requestModel);
        }
        public virtual async Task<string>
DeleteComment(RequestInfo<DeleteCommentRequestModel> requestModel)
        {
            var requestPath = string.Format("cookingrecipes/comments/deletebytoken");
            return await
this.apiService.DeleteDataFromServer<DeleteCommentRequestModel>(requestPath,
requestModel);
        }
        public virtual async Task<string> EditComment(RequestInfo<EditCommentRequestModel>
requestModel)
        {
            var requestPath = string.Format("cookingrecipes/comments/editbytoken");
            return await
this.apiService.UpdateDataToServer<EditCommentRequestModel>(requestPath, requestModel);
        }

        public virtual async Task<GetCommentsResponseModel>
GetCommentsList(RequestInfo<GetRecipeCommentsRequestModel> model)
        {
            var requestPath = string.Format("cookingrecipes/comments/getlist/byrecipe");
            return await this.apiService.GetDataFromServer<GetRecipeCommentsRequestModel,
GetCommentsResponseModel>(requestPath, model);
        }
        public virtual async Task<GetCommentResponseModel>
GetCommentInfo(RequestInfo<GetCommentRequestModel> model)
        {
            var requestPath = string.Format("cookingrecipes/comments/getbytoken");
            return await this.apiService.GetDataFromServer<GetCommentRequestModel,
GetCommentResponseModel>(requestPath, model);
        }
    }
}

```

Файл «CookingRecipes.cs»

```

using MauiLabs.View.Services.ApiModels.Commons;
using MauiLabs.View.Services.ApiModels.ProfileModels.Authorization.Responses;
using MauiLabs.View.Services.ApiModels.RecipeModels.CookingRecipe.Requests;
using MauiLabs.View.Services.ApiModels.RecipeModels.CookingRecipe.Responses;
using MauiLabs.View.Services.ApiModels.RecipeModels.IngredientUnits.Responses;

```

```

using MauiLabs.View.Services.ApiModels.RecipeModels.RecipeCategory.Responses;
using MauiLabs.View.Services.Commons;
using MauiLabs.View.Services.ConfigureOptions;
using MauiLabs.View.Services.Interfaces;
using Microsoft.Extensions.Options;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http.Json;
using System.Text;
using System.Text.Json;
using System.Threading.Tasks;
using System.Web;

namespace MauiLabs.View.Services.Implements
{
    using WebApiOptions = ConfigureWebApi.WebApiOptions;
    public partial class CookingRecipes : ICookingRecipes
    {
        protected internal readonly IApiServiceCommunication apiService = default!;
        public CookingRecipes(IApiServiceCommunication apiService) : base() =>
        this.apiService = apiService;

        public virtual async Task<GetRecipesListResponseModel>
        GetRecipesList(RequestInfo<GetRecipesListRequestModel> model)
        {
            var requestPath = string.Format("cookingrecipes/recipes/getlist");
            return await this.apiService.GetDataFromServer<GetRecipesListRequestModel,
            GetRecipesListResponseModel>(requestPath, model);
        }
        public virtual async Task<GetRecipeResponseModel>
        GetRecipeInfo(RequestInfo<GetRecipeRequestModel> model)
        {
            var requestPath = string.Format("cookingrecipes/recipes/get");
            return await this.apiService.GetDataFromServer<GetRecipeRequestModel,
            GetRecipeResponseModel>(requestPath, model);
        }
        public virtual async Task<string> AddRecipeInfo(RequestInfo<AddRecipeRequestModel>
        requestModel)
        {
            var requestPath = string.Format("cookingrecipes/recipes/addbytoken");
            return await
            this.apiService.AddDataToServer<AddRecipeRequestModel>(requestPath, requestModel);
        }
        public virtual async Task<string>
        DeleteRecipeInfo(RequestInfo<DeleteRecipeRequestModel> requestModel)
        {
            var requestPath = string.Format("cookingrecipes/recipes/delete");
            return await
            this.apiService.DeleteDataFromServer<DeleteRecipeRequestModel>(requestPath, requestModel);
        }
        public virtual async Task<string>
        EditRecipeInfo(RequestInfo<EditRecipeRequestModel> requestModel)
        {
            var requestPath = string.Format("cookingrecipes/recipes/edit");
            return await
            this.apiService.UpdateDataToServer<EditRecipeRequestModel>(requestPath, requestModel);
        }
        public virtual async Task<GetRecipesListResponseModel>
        GetPublishedListById(RequestInfo<GetPublisherRecipesListByIdRequestModel> model)
        {
            var requestPath = string.Format("cookingrecipes/recipes/getpublisherlist");
            return await this.apiService

```

```

        .GetDataFromServer<GetPublisherRecipesListByIdRequestModel,
        GetRecipesListResponseModel>(requestPath, model);
    }
    public virtual async Task<GetRecipesListResponseModel>
    GetPublishedList(RequestInfo<GetPublisherRecipesListRequestModel> model)
    {
        var requestPath =
        string.Format("cookingrecipes/recipes/getpublisherlist/bytoken");
        return await
        this.apiService.GetDataFromServer<GetPublisherRecipesListRequestModel,
        GetRecipesListResponseModel>(requestPath, model);
    }
    public virtual async Task<GetRecipeCategoriesListResponseModel>
    GetCategoriesList(string token, CancellationToken cancelToken)
    {
        var requestPath = string.Format("cookingrecipes/category/getlist");
        using var request = new HttpRequestMessage(HttpMethod.Get, requestPath)
        {
            Headers = { { "Authorization", string.Format("Bearer {0}", token) } },
        };
        return await this.apiService.SendRequestAsync(request, cancelToken, async
content =>
        {
            return
            JsonConvert.DeserializeObject<GetRecipeCategoriesListResponseModel>(await
content.ReadAsStringAsync());
        });
    }
    public virtual async Task<GetIngredientUnitsResponseModel> GetUnitsList(string
token, CancellationToken cancelToken)
    {
        var requestPath = string.Format("cookingrecipes/units/getlist");
        using var request = new HttpRequestMessage(HttpMethod.Get, requestPath)
        {
            Headers = { { "Authorization", string.Format("Bearer {0}", token) } },
        };
        return await this.apiService.SendRequestAsync(request, cancelToken, async
content =>
        {
            return JsonConvert.DeserializeObject<GetIngredientUnitsResponseModel>(await
content.ReadAsStringAsync());
        });
    }
}

```

Файл «NavigationService.cs»

```

using MauiLabs.View.Services.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MauiLabs.View.Services.Implements
{
    using Parameters = IDictionary<string, object>;
    public partial class NavigationService : INavigationService
    {
        protected internal readonly IServiceProvider serviceProvider = default!;
        public NavigationService(IServiceProvider serviceProvider) : base()
        {
            this.serviceProvider = serviceProvider;
        }
    }
}

```

```

        protected virtual TPage GeneratePage<TPage>() =>
ActivatorUtilities.CreateInstance<TPage>(this.serviceProvider);
        public virtual async Task NavigateToPage<TPage>(Shell root, Parameters parameters
= null) where TPage : Page
        {
            var pageInstance = this.serviceProvider.GetService<TPage>() ??
this.GeneratePage<TPage>();
            if (pageInstance is INavigationService.IQueryableNavigation queryAttributable
&& parameters != null)
            {
                queryAttributable.SetNavigationQuery(parameters);
            }
            pageInstance.Unloaded += (_, _) =>
CookingRecipeShell.SetTabBarVisibility(pageInstance, true);
            pageInstance.Loaded += (_, _) =>
CookingRecipeShell.SetTabBarVisibility(pageInstance, false);

            await root.Navigation.PushAsync(pageInstance);
        }
    }
}

```

Файл «UserAuthorization.cs»

```

using MauiLabs.View.Services.ConfigureOptions;
using MauiLabs.View.Services.Interfaces;
using Microsoft.Extensions.Options;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http.Json;
using System.Text;
using System.Text.Json;
using System.Threading.Tasks;
using MauiLabs.View.Services.ApiModels.Commons;
using MauiLabs.View.Services.ApiModels.ProfileModels.Authorization.Responses;
using MauiLabs.View.Services.ApiModels.ProfileModels.Authorization.Requests;
using MauiLabs.View.Services.Commons;

namespace MauiLabs.View.Services.Implements
{
    #nullable enable
    using WebApiOptions = ConfigureWebApi.WebApiOptions;
    public partial class UserAuthorization : IUserAuthorization
    {
        protected internal readonly IHttpClientFactory httpClientFactory = default!;
        protected internal readonly WebApiOptions webApiOptions = default!;

        private readonly JsonSerializerOptions jsonOptions = new
JsonSerializerOptions(JsonSerializerDefaults.Web);
        public UserAuthorization(IHttpClientFactory httpFactory, IOptions<WebApiOptions>
options) : base()
        {
            this.httpClientFactory = httpFactory;
            this.webApiOptions = options.Value;
        }
        public virtual async Task<LoginResponseModel?> AuthorizeUser(LoginRequestModel
model, CancellationToken token)
        {
            using (var httpClient =
this.httpClientFactory.CreateClient(this.webApiOptions.ApiClient))
            {
                var requestPath =
string.Format("cookingrecipes/auth/login?Login={0}&Password={1}", model.Login,
model.Password);
            }
        }
    }
}

```

```

        using var response = await httpClient.GetAsync(requestPath, token);
        if (response.StatusCode != HttpStatusCode.OK)
        {
            var errorMessage = await
response.Content.ReadFromJsonAsync<ProblemDetails>(jsonOptions);
            throw new ViewServiceException(errorMessage?.Detail ??
errorMessage?.Title);
        }
        return await
response.Content.ReadFromJsonAsync<LoginResponseModel>(jsonOptions);
    }

    public virtual async Task<LoginResponseModel?>
RegistrationUser(RegistrationRequestModel model, Cancellation token)
    {
        using (var httpClient =
this.httpClientFactory.CreateClient(this.webApiOptions.ApiClient))
        {
            using var response = await
httpClient.PostAsJsonAsync($"cookingrecipes/auth/registration", model, token);
            if (response.StatusCode != HttpStatusCode.OK)
            {
                if (token.IsCancellationRequested) return null;
                var errorMessage = await
response.Content.ReadFromJsonAsync<ProblemDetails>(jsonOptions);
                throw new ViewServiceException(errorMessage?.Detail ??
errorMessage?.Title);
            }
            return await
response.Content.ReadFromJsonAsync<LoginResponseModel>(jsonOptions);
        }
    }
}
#nullable disable
}

```

Файл «UserProfile.cs»

```

using MauiLabs.View.Services.ApiModels.Commons;
using MauiLabs.View.Services.ApiModels.ProfileModels.Profile.Requests;
using MauiLabs.View.Services.ApiModels.ProfileModels.Profile.Responses;
using MauiLabs.View.Services.Commons;
using MauiLabs.View.Services.ConfigureOptions;
using MauiLabs.View.Services.Interfaces;
using Microsoft.Extensions.Options;
using Microsoft.Maui.Controls.PlatformConfiguration;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http.Json;
using System.Text;
using System.Text.Json;
using System.Threading.Tasks;

namespace MauiLabs.View.Services.Implements
{
    using WebApiOptions = ConfigureWebApi.WebApiOptions;
    public partial class UserProfile : IUserProfile
    {
        protected internal readonly IApiServiceCommunication apiService = default!;
    }
}

```



```

    public UserProfile(IApiServiceCommunication apiService) : base() =>
    this.apiService = apiService;

    public virtual async Task<string>
    ChangeProfilePassword(RequestInfo<ChangePasswordRequestModel> requestModel)
    {
        var requestPath = string.Format("cookingrecipes/profile/editbytoken/password");
        return await
    this.apiService.UpdateDataToServer<ChangePasswordRequestModel>(requestPath, requestModel,
    false);
    }

    public virtual async Task<string> DeleteProfileInfo(string token,
    CancellationTokens cancelToken)
    {
        var requestPath = string.Format("cookingrecipes/profile/deletebytoken");
        using var request = new HttpRequestMessage(HttpMethod.Delete, requestPath)
        {
            Headers = { { "Authorization", string.Format("Bearer {0}", token) } },
        };
        return await this.apiService.SendRequestAsync<string>(request, cancelToken,
    async msg => await msg.ReadAsStringAsync());
    }

    public virtual async Task<string>
    EditProfileInfo(RequestInfo<EditProfileRequestModel> requestModel)
    {
        var requestPath = string.Format("cookingrecipes/profile/editbytoken");
        return await
    this.apiService.UpdateDataToServer<EditProfileRequestModel>(requestPath, requestModel);
    }

    public virtual async Task<GetProfileInfoResponseModel> GetProfileInfo(string token,
    int id, CancellationTokens cancelToken)
    {
        var requestPath = string.Format("cookingrecipes/profile/get?Id={0}", id);
        using var request = new HttpRequestMessage(HttpMethod.Get, requestPath)
        {
            Headers = { { "Authorization", string.Format("Bearer {0}", token) } },
        };
        return await this.apiService.SendRequestAsync(request, cancelToken, async
    content =>
    {
        return JsonConvert.DeserializeObject<GetProfileInfoResponseModel>(await
    content.ReadAsStringAsync());
    });
    }

    public virtual async Task<GetProfileInfoResponseModel>
    GetProfileInfoByToken(string token, CancellationTokens cancelToken)
    {
        var requestPath = string.Format("cookingrecipes/profile/getbytoken");
        using var request = new HttpRequestMessage(HttpMethod.Get, requestPath)
        {
            Headers = { { "Authorization", string.Format("Bearer {0}", token) } },
        };
        return await this.apiService.SendRequestAsync(request, cancelToken, async
    content =>
    {
        return JsonConvert.DeserializeObject<GetProfileInfoResponseModel>(await
    content.ReadAsStringAsync());
    });
    }

    public virtual async Task<GetProfilesListResponseModel>
    GetProfilesList(RequestInfo<GetProfilesListRequestModel> requestModel)
    {
        var requestPath = string.Format("cookingrecipes/profile/getlist");
    }

```



```

        return await this.apiService
            .GetDataFromServer<GetProfilesListRequestModel,
            GetProfilesListResponseModel>(requestPath, requestModel);
    }
}

```

Файл «ConfigureWebApi.cs»

```

using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Options;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MauiLabs.View.Services.ConfigureOptions
{
    public partial class ConfigureWebApi :
        IConfigurationNamedOptions<ConfigureWebApi.WebApiOptions>
    {
        protected internal readonly IConfiguration configuration = default!;
        public ConfigureWebApi(IConfiguration configuration): base() => this.configuration
        = configuration;

        public sealed class WebApiOptions : object
        {
            public required string BaseUrl { get; set; } = default!;
            public required string ApiKey { get; set; } = default!;
            public required string ApiClient { get; set; } = default!;
        }
        public virtual void Configure(string name, WebApiOptions options) =>
        this.Configure(options);

        public virtual void Configure(WebApiOptions options)
        {
            var webApiOptions =
            this.configuration.GetSection("WebApi").Get<WebApiOptions>();
            options.ApiClient = webApiOptions.ApiClient;

            (options.ApiKey, options.BaseUrl) = (webApiOptions.ApiKey,
            webApiOptions.BaseUrl);
        }
    }
}

```

Файл «RequestInfo.cs»

```

using System.Net.Http.Json;
using System.Web;

namespace MauiLabs.View.Services.Commons
{
    public partial class RequestInfo<TRequest> : object where TRequest : class
    {
        public required CancellationToken CancelToken { get; set; } = default!;
        public required string ProfileToken { get; set; } = default!;
        public required TRequest RequestModel { get; set; } = default!;

        protected virtual string GetQueryString(TRequest model)
        {
            var properties = from item in model.GetType().GetProperties()
                             where item.GetValue(model, null) != null
                             select item.Name + "=" +
            HttpUtility.UrlEncode(item.GetValue(model, null).ToString());
            return String.Join("&", properties.ToArray());
        }
    }
}

```

```

    }

    public HttpRequestMessage CreateRequestMessage(HttpMethod method, string path)
    {
        var bodyUsing = !(method == HttpMethod.Delete || method == HttpMethod.Get);
        var urlBuilder = new UriBuilder(new Uri(path));

        if (!bodyUsing) urlBuilder.Query = this.GetQueryString(this.RequestModel);
        var requestMessage = new HttpRequestMessage(method, urlBuilder.Uri)
        {
            Headers = { { "Authorization", $"Bearer {this.ProfileToken}" } },
            Content = bodyUsing ? JsonContent.Create(this.RequestModel) : null,
        };
        return requestMessage;
    }
    public override int GetHashCode() => this.RequestModel.GetHashCode();
    public override string ToString() => base.ToString();
}
}

```

Файл «UserManager.cs»

```

using MauiLabs.View.Services.ApiModels.ProfileModels.Authorization.Responses;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading.Tasks;

namespace MauiLabs.View.Services.Commons
{
    #nullable enable
    public static class UserManager : object
    {
        public static string AuthorizationRoute { get; private set; } = "//authorization";
        public static async Task AuthorizeUser(LoginResponseModel authorizationInfo)
        {
            await SecureStorage.Default.SetAsync("IsAdmin",
authorizationInfo.IsAdmin.ToString());
            await SecureStorage.Default.SetAsync("ProfileId",
authorizationInfo.ProfileId.ToString());

            await SecureStorage.Default.SetAsync("JwtToken", authorizationInfo.JwtToken);
            CookingRecipeShell.SetIsAdmin(authorizationInfo.IsAdmin);
        }
        public static async Task LogoutUser() => await Task.Run(async () =>
        {
            CookingRecipeShell.SetIsAdmin(false);
            SecureStorage.Default.RemoveAll();
            await Application.Current!.Dispatcher.DispatchAsync(async () =>
            {
                await Shell.Current.Navigation.PopToRootAsync();
                await Shell.Current.GoToAsync(UserManager.AuthorizationRoute);
            });
        });
        public static async Task<bool?> IsAdmin() => bool.Parse(await
SecureStorage.Default.GetAsync("IsAdmin"));
        public static async Task<int?> ProfileId() => int.Parse(await
SecureStorage.Default.GetAsync("ProfileId"));
        public static async Task<string> JwtToken() => await
SecureStorage.Default.GetAsync("JwtToken");

        private static async Task DisplayAlertAsync(string message)
        {
            if (Application.Current != null && Application.Current.MainPage != null)

```

```

    {
        MainThread.BeginInvokeOnMainThread(async () =>
        {
            await Application.Current.MainPage.DisplayAlert("Произошла ошибка",
message, "Назад");
        });
    }
    await Console.Out.WriteLineAsync(message);
}
public static async Task SendRequest(Func<string, Task> request, Action<Exception>?
cancelled = null)
{
    try { await request.Invoke(await UserManager.JwtToken()); }
    catch (ViewServiceException errorInfo) when (errorInfo.ExceptionType !=
HttpStatusCode.Unauthorized)
    {
        await UserManager.DisplayAlertAsync(errorInfo.Message);
    }
    catch (ViewServiceException errorInfo) when (errorInfo.ExceptionType ==
HttpStatusCode.Unauthorized)
    {
        await UserManager.DisplayAlertAsync("Пользователь не авторизирован");
        await LogoutUser();
    }
    catch (TaskCanceledException errorInfo) when (errorInfo.InnerException is
TimeoutException)
    {
        await UserManager.DisplayAlertAsync("Время подключения к серверу истекло");
        await LogoutUser();
    }
    catch (Exception errorInfo) { await CatchRequestError(errorInfo, cancelled); }
}
public static async Task CatchRequestError(Exception errorInfo, Action<Exception>?
cancelled = null)
{
    var errorType = errorInfo.GetType();
    /*
        Bad URL address (Domen):    [Desktop] = HttpRequestException, [Android] =
WebException;
        Device turn of WIFI:        [Desktop] = HttpRequestException, [Android] =
WebException,

        Bad URL address host:       404 Status code;
        Server OFF:                  503 Status code;

        Request cancelled:          [Desktop] = TaskCanceledException, [Android] =
WebException;
    */
    if (!(errorInfo is TaskCanceledException))
    {
        await UserManager.DisplayAlertAsync(errorInfo.Message);
        await LogoutUser();
    }
    else if (cancelled != null) cancelled.Invoke(errorInfo);
}
}
#nullable disable
}

```

Файл «ExpanderView.xaml»

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentView xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MauiLabs.View.Commons.ContentViews.ExpanderView"

```

```

        x:Name="this">
<ContentView.Resources>
    <ResourceDictionary>
        <Style x:Key="BorderStyle" TargetType="Border">
            <Setter Property="Stroke" Value="White"/>
            <Setter Property="StrokeShape">
                <Setter.Value>
                    <RoundRectangle CornerRadius="10"/>
                </Setter.Value>
            </Setter>
            <Setter Property="BackgroundColor" Value="{x:StaticResource FirstColor}"/>
            <Setter Property="StrokeThickness" Value="2"/>
        </Style>
        <Style x:Key="TextStyle" TargetType="Label">
            <Setter Property="FontSize" Value="16"/>
            <Setter Property="TextColor" Value="White"/>
        </Style>
        <Style x:Key="IconStyle" TargetType="Image">
            <Setter Property="WidthRequest" Value="12"/>
            <Setter Property="HeightRequest" Value="12"/>
            <Setter Property="Margin" Value="5, 4"/>
        </Style>
    </ResourceDictionary>
</ContentView.Resources>
<Grid RowSpacing="5" ColumnDefinitions="*" RowDefinitions="*"
HorizontalOptions="Fill">
    <Border Grid.Column="0" Grid.Row="0" x:Name="MainContent" HeightRequest="0"
        Style="{StaticResource BorderStyle}" Content="{x:Binding Item,
Source={x:Reference this}}"/>

    <Border Grid.Column="0" Grid.Row="0" x:Name="ExpandButton" VerticalOptions="Start"
        Style="{StaticResource BorderStyle}" HeightRequest="{x:Binding ButtonHeight,
Source={x:Reference this}}">

        <FlexLayout HorizontalOptions="Fill" Direction="Row" AlignItems="Center"
JustifyContent="Center" Padding="10">
            <Label Text="{x:Binding ButtonText, Source={x:Reference this}}"
Style="{x:StaticResource TextStyle}"/>
            <Image x:Name="ExpanderIcon" Style="{x:StaticResource IconStyle}"
                Source="{x:Binding ButtonIcon, Source={x:Reference this}}"/>
        </FlexLayout>
    </Border>
</Grid>
</ContentView>
using CommunityToolkit.Maui.Alerts;
using Microsoft.Maui;
using Microsoft.Maui.Controls;

namespace MauiLabs.View.Commons.ContentViews;

using MauiView = Microsoft.Maui.Controls.View;

public partial class ExpanderView : ContentView
{
    public static readonly BindableProperty ItemProperty = BindableProperty.Create(
        "Item", typeof(MauiView), typeof(ExpanderView), new StackLayout());

    public static readonly BindableProperty CanExpandProperty = BindableProperty.Create(
        "CanExpand", typeof(bool), typeof(ValidationEntryView), defaultValue: true);

    public static readonly BindableProperty ButtonTextProperty = BindableProperty.Create(
        "ButtonText", typeof(string),
        typeof(ExpanderView), defaultValueCreator: (@object) => "Раскрыть");

    public static readonly BindableProperty IsExpandedProperty = BindableProperty.Create(

```

```

        "IsExpanded", typeof(bool), typeof(ExpanderView), default!);

    public static readonly BindableProperty ButtonIconProperty = BindableProperty.Create(
        "ButtonIcon", typeof(ImageSource), typeof(ExpanderView), null);

    public static readonly BindableProperty ButtonHeightProperty =
        BindableProperty.Create(
            "ButtonHeight", typeof(double), typeof(ExpanderView), defaultValue: 48.0);

    public static readonly BindableProperty ExpanderHeightProperty =
        BindableProperty.Create(
            "ExpanderHeight", typeof(double), typeof(ExpanderView), default!);

    public MauiView Item { get => (MauiView)this.GetValue(ItemProperty); set =>
        this.SetValue(ItemProperty, value); }
    public string ButtonText { get => (string)this.GetValue(ButtonTextProperty); set =>
        this.SetValue(ButtonTextProperty, value); }
    public double ButtonHeight
    {
        get => (double)this.GetValue(ButtonHeightProperty); set =>
        this.SetValue(ButtonHeightProperty, value);
    }
    public ImageSource ButtonIcon
    {
        get => (ImageSource)this.GetValue(ButtonIconProperty); set =>
        this.SetValue(ButtonIconProperty, value);
    }
    public double ExpanderHeight { get => (double)GetValue(ExpanderHeightProperty); set
=> SetValue(ExpanderHeightProperty, value); }
    public bool CanExpand { get => (bool)GetValue(CanExpandProperty); set =>
        SetValue(CanExpandProperty, value); }
    public bool IsExpanded { get => (bool)GetValue(IsExpandedProperty); set =>
        SetValue(IsExpandedProperty, value); }

    public virtual int ExpandTapsRequired { get => 1; }
    public virtual bool IsPlaying { get; protected set; } = default!;

    protected Animation CompressAnimation { get => new(prop =>
        this.MainContent.HeightRequest = prop,
        this.ExpanderHeight, 0, easing: Easing.Linear); }
    protected Animation ExpandAnimation { get => new(prop =>
        this.MainContent.HeightRequest = prop,
        0, this.ExpanderHeight, easing: Easing.Linear); }
    public ExpanderView() : base()
    {
        this.InitializeComponent();
        var buttonTapRecognizer = new TapGestureRecognizer() { NumberOfTapsRequired =
        this.ExpandTapsRequired };
        buttonTapRecognizer.Tapped += this.ExpandButtonTapHandler;

        this.Dispatcher.Dispatch(() =>
        {
            this.MainContent.Padding = new Thickness(0, this.ExpandButton.Height, 0, 0);
            this.Item.Opacity = 0;
        });
        this.ExpandButton.GestureRecognizers.Add(buttonTapRecognizer);
    }
    protected virtual async void ExpandButtonTapHandler(object sender, TappedEventArgs
args)
    {
        var currentAnimation = this.IsExpanded ? this.CompressAnimation :
        this.ExpandAnimation;
        if (this.IsPlaying == true || !this.CanExpand) return; else this.IsPlaying = true;

        if (this.IsExpanded) await this.Item.FadeTo(0.0, 500, easing: Easing.SinInOut);
    }

```

```

        await Task.WhenAll(new Task[]
        {
            this.ExpanderIcon.RelRotateTo(180, length: 500, easing: Easing.Linear),
            this.Dispatcher.DispatchAsync(() => this.MainContent.Animate("Expand",
currentAnimation, length: 600)),
        });
        if (!this.IsExpanded) await this.Item.FadeTo(1.0, 500, easing: Easing.SinInOut);
        (this.IsPlaying, this.IsExpanded) = (default!, !this.IsExpanded);
    }
    public virtual void ResetExpander()
    {
        (this.ExpanderIcon.Rotation, this.MainContent.HeightRequest, this.Item.Opacity) =
(default, default, default);
        this.SetValue(IsExpandedProperty, false);
    }
    protected override void InvalidateLayout() => base.InvalidateLayout();
}

```

Файл «LoadingContentView.xaml»

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentView xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"
    x:Class="MauiLabs.View.Commons.ContentViews.LoadingContentView"
    Opacity="0" IsVisible="False">
    <ContentView.Resources>
        <ResourceDictionary>
            <toolkit:InvertedBoolConverter x:Key="InvertedBoolConverter" />
            <Color x:Key="LoadingBackground">#881A1A1D</Color>
            <Style x:Key="PanelStyle" TargetType="Border">
                <Setter Property="BackgroundColor" Value="{x:StaticResource SecondColor}"/>
                <Setter Property="StrokeShape">
                    <Setter.Value>
                        <RoundRectangle CornerRadius="20"/>
                    </Setter.Value>
                </Setter>
                <Setter Property="Stroke" Value="White"/>
                <Setter Property="StrokeThickness" Value="1"/>
                <Setter Property="Padding" Value="10"/>
            </Style>
            <Style x:Key="LoadingLabelStyle" TargetType="Label">
                <Setter Property="TextColor" Value="White"/>
                <Setter Property="FontSize" Value="16"/>
            </Style>
            <Style x:Key="CancelButtonStyle" TargetType="Button">
                <Setter Property="TextColor" Value="White"/>
                <Setter Property="FontSize" Value="16"/>
                <Setter Property="BorderColor" Value="White"/>
                <Setter Property="BorderWidth" Value="1"/>
                <Setter Property="CornerRadius" Value="10"/>
                <Setter Property="BackgroundColor" Value="{x:StaticResource FirstColor}"/>
            </Style>
        </ResourceDictionary>
    </ContentView.Resources>
    <FlexLayout BackgroundColor="{x:StaticResource LoadingBackground}"
AlignItems="Center"
        Direction="Column" JustifyContent="Center"
        IsEnabled="{Binding IsLoading, Converter={StaticResource
Key=InvertedBoolConverter}}">
        <Border x:Name="LoadingPanel" Style="{x:StaticResource PanelStyle}" Scale="0"
MaximumWidthRequest="400">
            <FlexLayout AlignItems="Center" Direction="Column" JustifyContent="Center"
MaximumHeightRequest="280">

```

```

        <ActivityIndicator IsRunning="{Binding IsLoading}" HeightRequest="20"
WidthRequest="20"
                Color="{x:StaticResource FirstColor}" Margin="0, 10"/>
        <Label Text="Подождите..." Style="{x:StaticResource LoadingLabelStyle}"/>
        <Button Text="Отмена" Style="{x:StaticResource CancelButtonStyle}"
                Command="{Binding CancelCommand}" Margin="0, 10" />
    </FlexLayout>
</Border>
</FlexLayout>
</ContentView>

using System.Windows.Input;

namespace MauiLabs.View.Commons.ContentViews;

public partial class LoadingContentView : ContentView
{
    public static readonly BindableProperty IsLoadingProperty = BindableProperty.Create(
        "IsLoading", typeof(bool),
        typeof(LoadingContentView), false, propertyChanged: LoadingPropertyChanged);

    public static readonly BindableProperty CancelCommandProperty =
BindableProperty.Create(
        "CancelCommand", typeof(ICommand), typeof(LoadingContentView));

    public bool IsLoading { get => (bool)this.GetValue(IsLoadingProperty); set =>
this.SetValue(IsLoadingProperty, value); }
    public ICommand CancelCommand
    {
        get => (ICommand)this.GetValue(CancelCommandProperty); set =>
this.SetValue(CancelCommandProperty, value);
    }
    public LoadingContentView() : base() => this.InitializeComponent();

    protected static async void LoadingPropertyChanged(BindableObject @object, object
oldValue, object newValue)
    {
        if(@object is LoadingContentView loadingAnimator)
        {
            if (loadingAnimator.IsLoading) { loadingAnimator.IsVisible = true; await
loadingAnimator.PlayAnimation(); }
            else { await loadingAnimator.StopAnimation(); loadingAnimator.IsVisible =
false; }
        }
        protected virtual async Task PlayAnimation() => await Task.WhenAll(new Task[]
        {
            this.LoadingPanel.ScaleTo(1.0, length: 800, easing: Easing.SinInOut),
            this.FadeTo(1.0, length: 800, easing: Easing.SinInOut),
        });
        protected virtual async Task StopAnimation() => await Task.WhenAll(new Task[]
        {
            this.LoadingPanel.ScaleTo(0.0, length: 800, easing: Easing.SinInOut),
            this.FadeTo(0.0, length: 800, easing: Easing.SinInOut),
        });
    }
}

```

Файл «ValidationEntryView.xaml»

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentView xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"
    x:Class="MauiLabs.View.Commons.ContentViews.ValidationEntryView"
    x:Name="this">

```



```

<ContentView.Resources>
    <Style x:Key="TextFieldStyle" TargetType="Entry">
        <Setter Property="PlaceholderColor" Value="{x:StaticResource
Key=FirstColor}" />
        <Setter Property="BackgroundColor" Value="White" />
        <Setter Property="FontSize" Value="16" />
        <Setter Property="TextColor" Value="{x:StaticResource Key=SecondColor}" />
    </Style>
    <Style x:Key="ErrorLabelStyle" TargetType="Label">
        <Setter Property="BackgroundColor" Value="Transparent" />
        <Setter Property="FontSize" Value="12" />
        <Style.Triggers>
            <DataTrigger TargetType="Label" Value="True"
                Binding="{x:Binding IsValidated, Source={x:Reference this}}">
                <Setter Property="FontAttributes" Value="Bold" />
                <Setter Property="TextColor" Value="GreenYellow" />
            </DataTrigger>
            <DataTrigger TargetType="Label" Value="False"
                Binding="{x:Binding IsValidated, Source={x:Reference this}}">
                <Setter Property="FontAttributes" Value="None" />
                <Setter Property="TextColor" Value="White" />
            </DataTrigger>
        </Style.Triggers>
    </Style>
    <Style x:Key="TextFieldLabelStyle" TargetType="Label">
        <Setter Property="BackgroundColor" Value="Transparent" />
        <Setter Property="TextColor" Value="White" />
        <Setter Property="FontSize" Value="14" />
    </Style>
</ContentView.Resources>
<VerticalStackLayout Spacing="5" BindingContext="{x:Reference this}">
    <Label Text="{x:Binding LabelText}" Style="{x:StaticResource TextFieldLabelStyle}"
/>
    <Border BackgroundColor="White" Padding="2">
        <Border.StrokeShape>
            <RoundRectangle CornerRadius="10" />
        </Border.StrokeShape>
        <Entry Placeholder="{x:Binding DefaultText}" Text="{x:Binding TextValue,
Mode=TwoWay}"
            IsEnabled="{x:Binding CanInput}" IsSpellCheckEnabled="False"
            IsReadOnly="{x:Binding IsReadOnly}"
            ClearButtonVisibility="WhileEditing" x:Name="TextField"
            IsPassword="{x:Binding IsHidden}"
            Style="{x:StaticResource Key=TextFieldStyle}" MaxLength="{x:Binding
MaxLength}">
            <Entry.Behaviors>
                <toolkit:TextValidationBehavior
                    RegexPattern="{x:Binding Regex}" Flags="ValidateOnValueChanged"
x:Name="ValidationError"
                    MinimumLength="{x:Binding MinLength}" MaximumLength="{x:Binding
MaxLength}" />
            </Entry.Behaviors>
        </Entry>
    </Border>
    <Label Text="{x:Binding ErrorText}" Style="{x:StaticResource ErrorLabelStyle}" />
</VerticalStackLayout>
</ContentView>
using CommunityToolkit.Maui.Behaviors;
using System.ComponentModel;
using System.Windows.Input;

namespace MauiLabs.View.Commons.ContentViews
{
    #nullable enable
    public partial class ValidationEntryView : ContentView, INotifyPropertyChanged

```



```

{
    public static readonly BindableProperty IsValidatedProperty =
BindableProperty.Create(
    nameof(IsValidated), typeof(bool), typeof(ValidationEntryView),
    defaultValue: false, defaultBindingMode: BindingMode.OneWayToSource);

    public static readonly BindableProperty TextValueProperty =
BindableProperty.Create(
    nameof(TextValue), typeof(string), typeof(ValidationEntryView),
    defaultValue: string.Empty, defaultBindingMode: BindingMode.TwoWay);

    public static readonly BindableProperty ErrorTextProperty =
BindableProperty.Create(
    nameof(ErrorText), typeof(string), typeof(ValidationEntryView));

    public static readonly BindableProperty DefaultTextProperty =
BindableProperty.Create(
    nameof(DefaultText), typeof(string), typeof(ValidationEntryView));

    public static readonly BindableProperty LabelTextProperty =
BindableProperty.Create(
    nameof(LabelText), typeof(string), typeof(ValidationEntryView));

    public static readonly BindableProperty MaxLenghtProperty =
BindableProperty.Create(
    nameof(MaxLenght), typeof(int),
    typeof(ValidationEntryView), defaultValue: 50);

    public static readonly BindableProperty MinLenghtProperty =
BindableProperty.Create(
    nameof(MinLenght), typeof(int),
    typeof(ValidationEntryView), defaultValue: 5);

    public static readonly BindableProperty IsHiddenProperty =
BindableProperty.Create(
    nameof(IsHidden), typeof(bool), typeof(ValidationEntryView));

    public static readonly BindableProperty CanInputProperty =
BindableProperty.Create(
    nameof(CanInput), typeof(bool),
    typeof(ValidationEntryView), defaultValue: true);

    public static readonly BindableProperty IsReadOnlyProperty =
BindableProperty.Create(
    nameof(IsReadOnly), typeof(bool),
    typeof(ValidationEntryView), defaultValue: false);

    public static readonly BindableProperty RegexProperty = BindableProperty.Create(
    nameof(Regex), typeof(string),
    typeof(ValidationEntryView), defaultValue: string.Empty);

    public string TextValue { get => (string)GetValue(TextValueProperty); set =>
SetValue(TextValueProperty, value); }
    public bool IsHidden { get => (bool)GetValue(IsHiddenProperty); set =>
SetValue(IsHiddenProperty, value); }
    public bool CanInput { get => (bool)GetValue(CanInputProperty); set =>
SetValue(CanInputProperty, value); }
    public bool IsReadOnly { get => (bool)GetValue(IsReadOnlyProperty); set =>
SetValue(IsReadOnlyProperty, value); }

    public int MaxLenght { get => (int)GetValue(MaxLenghtProperty); set =>
SetValue(MaxLenghtProperty, value); }
    public int MinLenght { get => (int)GetValue(MinLenghtProperty); set =>
SetValue(MinLenghtProperty, value); }

```

```

        public string Regex { get => (string)GetValue(RegexProperty); set =>
SetValue(RegexProperty, value); }
        public bool IsValidated
        {
            get => (bool)this.GetValue(IsValidatedProperty); set =>
this.SetValue(IsValidatedProperty, value);
        }
        public string DefaultText
        {
            get => (string)this.GetValue(DefaultTextProperty); set =>
this.SetValue(DefaultTextProperty, value);
        }
        public string ErrorText { get => (string)GetValue(ErrorTextProperty); set =>
SetValue(ErrorTextProperty, value); }
        public string LabelText
        {
            get => (string)this.GetValue(LabelTextProperty); set =>
this.SetValue(LabelTextProperty, value);
        }
        public ValidationEntryView() : base()
        {
            this.InitializeComponent();
            var validationBinding = new Binding("IsValidated", source: this, mode:
BindingMode.OneWayToSource);
            var textValueBinding = new Binding("Text", source: this.TextField, mode:
BindingMode.TwoWay);

            this.ValidationErrors.SetBinding(ValidationBehavior.IsValidProperty,
validationBinding);
            this.ValidationErrors.IsValid = default!;

            this.SetBinding(ValidationEntryView.TextValueProperty, textValueBinding);
        }
        protected override void InvalidateLayout() => base.InvalidateLayout();
    }
#nullable disable
}

```

Файл «AuthorizationViewModel.cs»

```

using MauiLabs.View.Pages.RecipePages;
using MauiLabs.View.Services.ApiModels.ProfileModels.Authorization.Requests;
using MauiLabs.View.Services.ApiModels.ProfileModels.Authorization.Responses;
using MauiLabs.View.Services.Commons;
using MauiLabs.View.Services.Interfaces;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Diagnostics;
using System.Linq;
using System.Net;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Input;

namespace MauiLabs.View.Commons.ViewModels.ProfilesViewModels
{
    public partial class AuthorizationViewModel : INotifyPropertyChanged
    {
        protected internal readonly IUserAuthorization userAuthorization = default!;
        protected internal CancellationTokenSource cancellationSource = new();
        public ICommand AuthorizeCommand { get; protected set; } = default!;
        public ICommand CancelCommand { get; protected set; } = default!;
    }
}

```

```

public event PropertyChangedEventHandler PropertyChanged = default;
public AuthorizationViewModel(IUserAuthorization authorization) : base()
{
    this.userAuthorization = authorization;
    this.AuthorizeCommand = new Command(this.AuthorizeCommandHandler);
    this.CancelCommand = new Command(this.CancelCommandHandler);
}
protected virtual async void CancelCommandHandler(object sender) => await
Task.Run(() =>
{
    if (this.isLoading == false) return;
    this.cancellationSource.Cancel();

    this.cancellationSource = new CancellationTokenSource();
    this.IsLoading = default;
});
protected virtual async void AuthorizeCommandHandler(object sender)
{
    if (!this.IsLoginValid || !this.IsPasswordValid) return;
    this.IsLoading = true;
    LoginResponseModel requestResult = default!;
    try {
        requestResult = await
this.userAuthorization.AuthorizeUser(loginRequestModel, cancellationSource.Token);
        await UserManager.AuthorizeUser(requestResult);
        await Shell.Current.GoToAsync("//recipes", true);
    }
    catch (ViewServiceException errorInfo)
    {
        await Application.Current.MainPage.DisplayAlert("Произошла ошибка",
errorInfo.Message, "Назад");
    }
    catch (Exception errorInfo) { await
Console.Out.WriteLineAsync(errorInfo.Message); }
    this.IsLoading = false;
}
private protected bool isLoginValid = default!;
public bool IsLoginValid { get => this.isLoginValid; set { this.isLoginValid =
value; OnPropertyChanged(); } }

private protected bool isPasswordValid = default!;
public bool IsPasswordValid { get => this.isPasswordValid; set
{ this.isPasswordValid = value; OnPropertyChanged(); } }

private protected LoginRequestModel loginRequestModel = new();
public string UserLogin
{
    get => this.loginRequestModel.Login;
    set { this.loginRequestModel.Login = value; OnPropertyChanged(); }
}
public string UserPassword
{
    get => this.loginRequestModel.Password;
    set { this.loginRequestModel.Password = value; OnPropertyChanged(); }
}

private protected bool isLoading = default;
public bool IsLoading { get => this.isLoading; set { this.isLoading = value;
OnPropertyChanged(); } }

public virtual void OnPropertyChanged([CallerMemberName] string name = "")
{
    PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));
}
}

```

```
}
```

Файл «FriendsListViewModel.cs»

```
using MauiLabs.View.Services.ApiModels.Commons.ProfileModels;
using MauiLabs.View.Services.ApiModels.ProfileModels.FriendsList.Requests;
using MauiLabs.View.Services.Commons;
using MauiLabs.View.Services.Interfaces;
using SixLabors.ImageSharp.Advanced;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Linq;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Input;

namespace MauiLabs.View.Commons.ViewModels.ProfilesViewModels
{
    public partial class FriendsListViewModel : INotifyPropertyChanged
    {
        protected internal CancellationTokenSource cancellationSource = new();
        protected internal readonly IFriendsList friendsService = default!;

        public static readonly string DefaultProfileImage =
            $"MauiLabs.View.Resources.Images.Profile.defaultprofile.png";

        public ICommand GetFriendsListCommand { get; protected set; } = default!;
        public ICommand AddFriendCommand { get; protected set; } = default!;
        public ICommand DeleteFriendCommand { get; protected set; } = default!;
        public ICommand CancelCommand { get; protected set; } = default!;

        public event EventHandler<string> DisplayInfo = delegate { };
        public event EventHandler FriendsReload = delegate { };
        public event Func<string, Task<bool>> CheckConfirm = (_) => Task.FromResult(false);

        public event PropertyChangedEventHandler PropertyChanged;
        public FriendsListViewModel(IFriendsList friendsService) : base()
        {
            this.friendsService = friendsService;
            this.AddFriendCommand = new Command(() =>
            {
                if (this.ReferenceLink.Length >= 5) this.LaunchCancelableTask(() =>
                    this.AddFriendCommandHandler());
            });
            this.DeleteFriendCommand = new Command<int>(async (id) =>
            {
                if(await this.CheckConfirm.Invoke("Удалить данный профиль?"))
                {
                    this.LaunchCancelableTask(() => this.DeleteFriendCommandHandler(id));
                }
            });
            this.GetFriendsListCommand = new Command(() =>
            {
                this.LaunchCancelableTask(() => this.GetFriendsListCommandHandler());
            });
            this.CancelCommand = new Command(this.CancelCommandHandler);
        }
        protected virtual async void CancelCommandHandler(object sender) => await
            Task.Run(() =>
            {
                if (this.isLoading == false) return;
                this.cancellationSource.Cancel();
            });
    }
}
```

```

        this.cancellationSource = new CancellationTokenSource();
        (this.IsLoading, this.FriendsLoaded, this.AllCount) = (default, default, 0);
    });
    protected async void LaunchCancelableTask(Func<Task> cancelableTask) => await
Task.Run(async () =>
{
    this.IsLoading = true; await cancelableTask.Invoke();
    this.IsLoading = false;
});
    public virtual byte[] FileToByteArray(string filename)
    {
        using (var fileStream =
Assembly.GetExecutingAssembly().GetManifestResourceStream(filename))
        {
            using var binaryReader = new BinaryReader(fileStream);
            return binaryReader.ReadBytes((int)fileStream.Length);
        }
    }
    protected virtual async Task GetFriendsListCommandHandler() => await
UserManager.SendRequest(async (token) =>
{
    var requestResult = await this.friendsService.GetFriendsList(token,
this.cancellationSource.Token);
    foreach (var friendRecord in requestResult.Friends)
    {
        friendRecord.Profile.Image = friendRecord.Profile.Image.Length != 0
            ? friendRecord.Profile.Image : this.FileToByteArray(DefaultProfileImage);
    }
    this.FriendsList = new(requestResult.Friends);
    this.AllCount = requestResult.AllCount;

    this.FriendsReload.Invoke(this, new EventArgs());
    if (!this.FriendsLoaded) this.FriendsLoaded = true;
});
    protected virtual async Task AddFriendCommandHandler() => await
UserManager.SendRequest(async (token) =>
{
    await this.friendsService.AddFriend(new RequestInfo<AddFriendRequestModel>()
    {
        RequestModel = new AddFriendRequestModel() { ReferenceLink =
this.ReferenceLink },
        CancelToken = this.cancellationSource.Token, ProfileToken = token,
    });
    this.ReferenceLink = string.Empty;
    await this.GetFriendsListCommandHandler();
    this.DisplayInfo.Invoke(this, "Профиль друга добавлен");
});
    protected virtual async Task DeleteFriendCommandHandler(int id) => await
UserManager.SendRequest(async (token) =>
{
    await this.friendsService.DeleteFriend(new
RequestInfo<DeleteFriendRequestModel>()
    {
        RequestModel = new DeleteFriendRequestModel() { RecordId = id },
        CancelToken = this.cancellationSource.Token, ProfileToken = token,
    });
    await this.GetFriendsListCommandHandler();
    this.DisplayInfo.Invoke(this, "Профиль друга удален");
});
    public ObservableCollection<FriendInfoModel> FriendsList { get; protected set; } =
new();

    private protected string referenceLink = string.Empty;

```

```

        public string ReferenceLink { get => this.referenceLink; set { this.referenceLink
= value; OnPropertyChanged(); } }

        private protected int allCount = default!;
        public int AllCount { get => this.allCount; set { this.IsEmpty = (this.allCount =
value) <= 0; OnPropertyChanged(); } }

        private protected bool isEmpty = default!;
        public bool IsEmpty { get => this.isEmpty; set { this.isEmpty = value;
OnPropertyChanged(); } }

        private protected bool friendsLoaded = default;
        public bool FriendsLoaded { get => this.friendsLoaded; set { this.friendsLoaded =
value; OnPropertyChanged(); } }

        private protected bool isLoading = default;
        public bool IsLoading { get => this.isLoading; set { this.isLoading = value;
OnPropertyChanged(); } }

        public virtual void OnPropertyChanged([CallerMemberName] string name = "")
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));
        }
    }
}

```

Файл «ProfileInfoViewModel.cs»

```

using MauiLabs.View.Pages.ProfilePages;
using MauiLabs.View.Services.ApiModels.ProfileModels.Profile.Requests;
using MauiLabs.View.Services.ApiModels.ProfileModels.Profile.Responses;
using MauiLabs.View.Services.Commons;
using MauiLabs.View.Services.Interfaces;
using Microsoft.EntityFrameworkCore.Query.Internal;
using Microsoft.Maui.Layouts;
using SixLabors.ImageSharp.Formats.Png;
using SixLabors.ImageSharp.Processing;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Input;

namespace MauiLabs.View.Commons.ViewModels.ProfilesViewModels
{
    public partial class ProfileInfoViewModel : INotifyPropertyChanged
    {
        protected internal CancellationTokenSource cancellationSource = new();
        protected internal readonly IUserProfile profileService = default!;

        public static readonly string DefaultProfileImage =
$"MauiLabs.View.Resources.Images.Profile.defaultprofile.png";

        public ICommand GetProfileCommand { get; protected set; } = default!;
        public ICommand UpdateProfileCommand { get; protected set; } = default!;
        public ICommand DeleteProfileCommand { get; protected set; } = default!;

        public ICommand ChangePasswordCommand { get; protected set; } = default!;
        public ICommand CancelCommand { get; protected set; } = default!;

        public event EventHandler<string> DisplayAlert = delegate { };
        public event EventHandler<string> DisplayInfo = delegate { };
    }
}

```

```

public event Func<string, Task<bool>> CheckConfirm = (_) => Task.FromResult(false);

public event EventHandler<byte[]> ReloadImage = delegate { };
public event PropertyChangedEventHandler PropertyChanged;
public ProfileInfoViewModel(IUserProfile profileService) : base()
{
    this.profileService = profileService;
    this.GetProfileCommand = new Command(() => this.LaunchCancelableTask(() =>
this.GetProfileCommandHandler()));
    this.UpdateProfileCommand = new Command(() =>
    {
        if (!this.ProfileLoaded) { this.DisplayAlert.Invoke(this, "Необходимо
загрузить профиль"); return; }
        if (this.ValidationState.Where(item => !item.Value).Count() > 0)
        {
            this.DisplayAlert.Invoke(this, "Неверно заполнены поля"); return;
        }
        this.LaunchCancelableTask(() => this.UpdateProfileCommandHandler());
    });
    this.DeleteProfileCommand = new Command(async () =>
    {
        if (!this.ProfileLoaded) { this.DisplayAlert.Invoke(this, "Необходимо
загрузить профиль"); return; }
        if (await this.CheckConfirm.Invoke("Удалить данный профиль?"))
        {
            this.LaunchCancelableTask(() => this.DeleteProfileCommandHandler());
        }
    });
    this.ChangePasswordCommand = new Command(() => this.LaunchCancelableTask(() =>
this.ChangePasswordCommandHandler()));
    this.CancelCommand = new Command(this.CancelCommandHandler);
}
protected virtual async void CancelCommandHandler(object sender) => await
Task.Run(() =>
{
    if (this.isLoading == false) return;

    this.cancellationSource.Cancel();
    this.cancellationSource = new CancellationTokenseSource();

    (this.IsLoading, this.ProfileLoaded) = (default, default);
});
protected async void LaunchCancelableTask(Func<Task> cancelableTask) => await
Task.Run(async () =>
{
    this.IsLoading = true; await cancelableTask.Invoke();
    this.IsLoading = false;
});
public virtual byte[] FileToByteArray(string filename)
{
    using (var fileStream =
Assembly.GetExecutingAssembly().GetManifestResourceStream(filename))
    {
        using var binaryReader = new BinaryReader(fileStream);
        return binaryReader.ReadBytes((int)fileStream.Length);
    }
}
protected virtual async Task UpdateProfileCommandHandler() => await
UserManager.SendRequest(async (token) =>
{
    await this.profileService.EditProfileInfo(new
RequestInfo<EditProfileRequestModel>()
    {
        RequestModel = new EditProfileRequestModel()
    }
}

```



```

        Name = this.UserName, Surname = this.UserSurname, Email = this.UserEmail,
        Image = this.UserImage,
    },
    ProfileToken = token, CancelToken = this.cancellationSource.Token,
});
this.DisplayInfo.Invoke(this, "Данные успешно обновлены");
});
protected virtual async Task GetProfileCommandHandler() => await
userManager.SendRequest(async (token) =>
{
    var requestResult = await this.profileService.GetProfileInfoByToken(token,
this.cancellationSource.Token);
    if (requestResult.Image.Length == 0)
    {
        this.ReloadImage.Invoke(this, this.FileToByteArray(DefaultProfileImage));
        this.profileInfoModel.Image = null;
    }
    else this.ReloadImage.Invoke(this, this.profileInfoModel.Image =
requestResult.Image);

    (this.UserName, this.UserSurname) = (requestResult.Name,
requestResult.Surname);
    (this.UserEmail, this.ReferenceLink) = (requestResult.Email,
requestResult.ReferenceLink);

    if (!this.ProfileLoaded) this.ProfileLoaded = true;
    this.profileInfoModel.Id = requestResult.Id;
}, (errorInfo) =>
{
    this.UserName = this.UserSurname = this.UserEmail = "Не загружено";
    this.ReferenceLink = string.Empty;
    (this.profileInfoModel.Id, this.profileInfoModel.Image) = (-1, null);

    this.ReloadImage.Invoke(this, this.FileToByteArray(DefaultProfileImage));
    this.DisplayAlert.Invoke(this, "Данные не загружены, для редактирования
обновить");
});

protected virtual async Task DeleteProfileCommandHandler() => await
userManager.SendRequest(async (token) =>
{
    await this.profileService.DeleteProfileInfo(token,
this.cancellationSource.Token);
    this.DisplayInfo.Invoke(this, "Профиль успешно удалён");
    await userManager.LogoutUser();
});

protected virtual async Task ChangePasswordCommandHandler() => await
userManager.SendRequest(async (token) =>
{
    if (!this.ProfileLoaded) { this.DisplayAlert.Invoke(this, "Необходимо
загрузить профиль"); return; }
    if (this.PasswordValidationState.Where(item => !item.Value).Count() > 0)
    {
        this.DisplayAlert.Invoke(this, "Неверно заполнены поля"); return;
    }
    await this.profileService.ChangeProfilePassword(new
RequestInfo<ChangePasswordRequestModel>()
    {
        RequestModel = this.changePasswordModel,
        CancelToken = this.cancellationSource.Token, ProfileToken = token,
    });
    this.DisplayInfo.Invoke(this, "Пароль успешно обновлен");
    await userManager.LogoutUser();
});
});

```



```

private protected GetProfileInfoResponseModel profileInfoModel = new()
{
    Id = -1, Name = "Не загружено", Surname = "Не загружено",
    Email = "Не загружено", ReferenceLink = string.Empty,
};
public string UserName
{
    set { this.profileInfoModel.Name = value; this.OnPropertyChanged(); }
    get => this.profileInfoModel.Name;
}
public string UserSurname
{
    set { this.profileInfoModel.Surname = value; this.OnPropertyChanged(); }
    get => this.profileInfoModel.Surname;
}
public string UserEmail
{
    set { this.profileInfoModel.Email = value; this.OnPropertyChanged(); }
    get => this.profileInfoModel.Email;
}
public string ReferenceLink
{
    set { this.profileInfoModel.ReferenceLink = value; this.OnPropertyChanged(); }
    get => this.profileInfoModel.ReferenceLink;
}
public byte[] UserImage { get => this.profileInfoModel.Image; set =>
profileInfoModel.Image = value; }

private protected ChangePasswordRequestModel changePasswordModel = new()
{
    NewPassword = string.Empty, OldPassword = string.Empty
};
public string NewPassword
{
    set { this.changePasswordModel.NewPassword = value; this.OnPropertyChanged(); }
    get => this.changePasswordModel.NewPassword;
}
public string OldPassword
{
    set { this.changePasswordModel.OldPassword = value; this.OnPropertyChanged(); }
    get => this.changePasswordModel.OldPassword;
}

private protected bool profileLoaded = default;
public bool ProfileLoaded { get => this.profileLoaded; set { this.profileLoaded =
value; OnPropertyChanged(); } }

private protected bool isLoading = default;
public bool IsLoading { get => this.isLoading; set { this.isLoading = value;
OnPropertyChanged(); } }

public Dictionary<string, bool> ValidationState { get; set; } = new();
public Dictionary<string, bool> PasswordValidationState { get; set; } = new();
public virtual void OnPropertyChanged([CallerMemberName] string name = "")
{
    PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));
}
}
}

```

Файл «RegistrationViewModel.cs»

```

using MauiLabs.View.Services.ApiModels.ProfileModels.Authorization.Requests;
using MauiLabs.View.Services.ApiModels.ProfileModels.Authorization.Responses;
using MauiLabs.View.Services.Commons;

```

```

using MauiLabs.View.Services.Interfaces;
using SixLabors.ImageSharp.Formats.Png;
using SixLabors.ImageSharp.Processing;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Input;

namespace MauiLabs.View.Commons.ViewModels.ProfilesViewModels
{
    public partial class RegistrationViewModel : INotifyPropertyChanged
    {
        protected internal readonly IUserAuthorization userAuthorization = default!;
        protected internal CancellationTokenSource cancellationSource = new();
        public virtual double ImageSize { get => 112; }

        public static readonly string DefaultProfileImage =
            $"MauiLabs.View.Resources.Images.Profile.defaultprofile.png";

        public ICommand RegistrationCommand { get; protected set; } = default!;
        public ICommand ImagePickerCommand { get; protected set; } = default!;
        public ICommand ImageClearCommand { get; protected set; } = default!;
        public ICommand CancelCommand { get; protected set; } = default!;

        public event EventHandler<string> DisplayAlert = delegate { };
        public event PropertyChangedEventHandler PropertyChanged = default;
        public RegistrationViewModel(IUserAuthorization authorization) : base()
        {
            this.userAuthorization = authorization;
            this.RegistrationCommand = new Command(this.RegistrationCommandHandler);
            this.CancelCommand = new Command(this.CancelCommandHandler);

            this.ImagePickerCommand = new Command(this.ImagePickerCommandHandler);
            this.ImageClearCommand = new Command(async () =>
            {
                this.UserImage = null;
                this.PreviewImage = await this.FileToByteArray(DefaultProfileImage);
            });
            this.ImageClearCommand.Execute(this);
        }

        protected virtual async void CancelCommandHandler(object sender) => await
Task.Run(() =>
{
    if (this.isLoading == false) return;
    this.cancellationSource.Cancel();

    this.cancellationSource = new CancellationTokenSource();
    this.IsLoading = default;
});

        private Task<byte[]> FileToByteArray(string filename)
        {
            using (var fileStream =
Assembly.GetExecutingAssembly().GetManifestResourceStream(filename))
            {
                using var binaryReader = new BinaryReader(fileStream);
                return Task.FromResult(binaryReader.ReadBytes((int)fileStream.Length));
            }
        }
    }
}

```

```

    }
    protected virtual async void ImagePickerCommandHandler(object sender)
    {
        var fileFilter = (FileResult result, string extension) =>
        {
            return result.FileName.EndsWith(extension,
StringComparison.OrdinalIgnoreCase);
        };
        var pickerOption = new PickOptions() { FileTypes = FilePickerFileType.Images,
PickerTitle = "Выберите изображение" };
        try {
            var pickerResult = await FilePicker.Default.PickAsync(pickerOption);
            if (pickerResult != null && (fileFilter(pickerResult, "jpg") ||
fileFilter(pickerResult, "png")))
            {
                using var stream = await pickerResult.OpenReadAsync();
                using var image = SixLabors.ImageSharp.Image.Load(stream);
                image.Mutate(prop => prop.Resize((int)this.ImageSize,
(int)this.ImageSize, false));

                using var outputStream = new MemoryStream();
                image.Save(outputStream, new PngEncoder());
                this.UserImage = this.PreviewImage = outputStream.ToArray();
            }
        }
        catch (Exception errorInfo) { this.DisplayAlert.Invoke(this,
errorInfo.Message); }
    }

    protected virtual async void RegistrationCommandHandler(object sender)
    {
        if (this.ValidationState.Where(item => !item.Value).Count() > 0) return;
        this.IsLoading = true;
        LoginResponseModel requestResult = default!;
        try {
            requestResult = await
this.userAuthorization.RegistrationUser(registrationRequestModel,
cancellationSource.Token);
            await UserManager.AuthorizeUser(requestResult);
            await Shell.Current.GoToAsync("//recipes", true);
        }
        catch (ViewServiceException errorInfo)
        {
            await Application.Current.MainPage.DisplayAlert("Произошла ошибка",
errorInfo.Message, "Назад");
        }
        catch (Exception errorInfo) { await
Console.Out.WriteLineAsync(errorInfo.Message); }
        this.IsLoading = false;
    }
    private protected RegistrationRequestModel registrationRequestModel = new();
    public string UserName
    {
        set { this.registrationRequestModel.Name = value; this.OnPropertyChanged(); }
        get => this.registrationRequestModel.Name;
    }
    public string UserSurname
    {
        set { this.registrationRequestModel.Surname = value;
this.OnPropertyChanged(); }
        get => this.registrationRequestModel.Surname;
    }
    public string UserEmail
    {
        set { this.registrationRequestModel.Email = value; this.OnPropertyChanged(); }
    }

```

```

        get => this.registrationRequestModel.Email;
    }

    public string UserLogin
    {
        set { this.registrationRequestModel.Login = value; this.OnPropertyChanged(); }
        get => this.registrationRequestModel.Login;
    }
    public string UserPassword
    {
        set { this.registrationRequestModel.Password = value;
this.OnPropertyChanged(); }
        get => this.registrationRequestModel.Password;
    }
    public byte[] UserImage
    {
        set { this.PreviewImage = this.registrationRequestModel.Image = value;
this.OnPropertyChanged(); }
        get => this.registrationRequestModel.Image;
    }

    protected internal byte[] previewImage = new byte[0];
    public byte[] PreviewImage { get => this.previewImage; set { this.previewImage =
value; OnPropertyChanged(); } }

    private protected bool isLoading = default;
    public bool IsLoading { get => this.isLoading; set { this.isLoading = value;
OnPropertyChanged(); } }

    public Dictionary<string, bool> ValidationState { get; set; } = new();

    public virtual void OnPropertyChanged([CallerMemberName] string name = "")
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));
    }
}
}

```

Файл «PublishedListViewModel.cs»

```

using MauiLabs.View.Services.ApiModels.RecipeModels.CookingRecipe.Requests;
using MauiLabs.View.Services.ApiModels.RecipeModels.CookingRecipe.Responses;
using MauiLabs.View.Services.Commons;
using MauiLabs.View.Services.Interfaces;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Linq;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Input;

namespace MauiLabs.View.Commons.ViewModels.RecipesViewModels
{
    public partial class PublishedListViewModel : INotifyPropertyChanged
    {
        protected internal CancellationTokenSource cancellationSource = new();
        protected internal readonly ICookingRecipes cookingRecipes = default!;

        public static readonly string DefaultRecipeImage =
$"MauiLabs.View.Resources.Images.Recipe.defaultrecipe.jpg";
        public static readonly string DefaultCategory = "Любая категория";
    }
}

```

```

public ICommand GetPublishedListCommand { get; protected set; } = default!;
public ICommand DeletePublishedCommand { get; protected set; } = default!;
public ICommand CancelCommand { get; protected set; } = default!;

public event EventHandler<string> DisplayInfo = delegate { };
public event Func<string, Task<bool>> CheckConfirm = (_) => Task.FromResult(false);

public event EventHandler CategoriesReload = delegate { };
public event EventHandler RecipesReload = delegate { };

public event PropertyChangedEventHandler PropertyChanged;
public PublishedListViewModel(ICookingRecipes cookingRecipes) : base()
{
    this.cookingRecipes = cookingRecipes;
    this.GetPublishedListCommand = new Command(() =>
    {
        this.LaunchCancelableTask(() => this.GetPublishedListCommandHandler());
    });
    this.DeletePublishedCommand = new Command<int>(async (id) =>
    {
        if (await this.CheckConfirm.Invoke("Удалить данный рецепт?"))
        {
            this.LaunchCancelableTask(() => this.DeletePublishedCommandHandler(id));
        }
    });
    this.CancelCommand = new Command(this.CancelCommandHandler);
}
protected virtual async void CancelCommandHandler(object sender) => await
Task.Run(() =>
{
    if (this.IsLoading == false) return;
    this.cancellationSource.Cancel();

    this.cancellationSource = new CancellationTokenSource();
    (this.IsLoading) = (default);
});
protected async void LaunchCancelableTask(Func<Task> cancelableTask) => await
Task.Run(async () =>
{
    this.IsLoading = true; await cancelableTask.Invoke();
    this.IsLoading = false;
});
public virtual byte[] FileToByteArray(string filename)
{
    using (var fileStream =
Assembly.GetExecutingAssembly().GetManifestResourceStream(filename))
    {
        using var binaryReader = new BinaryReader(fileStream);
        return binaryReader.ReadBytes((int)fileStream.Length);
    }
}
protected virtual async Task GetPublishedListCommandHandler() => await
UserManager.SendRequest(async (token) =>
{
    var requestResult = await this.cookingRecipes.GetPublishedList(new
RequestInfo<GetPublisherRecipesListRequestModel>()
    {
        RequestModel = new GetPublisherRecipesListRequestModel()
        {
            TextFilter = this.TextFilter == string.Empty ? null : this.TextFilter,
            Category = this.Category == DefaultCategory ? null : this.Category,
        },
        CancelToken = this.cancellationSource.Token, ProfileToken = token,
    });
    foreach (var friendRecord in requestResult.Recipes)

```

```

{
    friendRecord.Image = friendRecord.Image.Length != 0
        ? friendRecord.Image : this.FileToByteArray(DefaultRecipeImage);
}
this.CookingRecipes = new(requestResult.Recipes);
this.AllCount = requestResult.AllCount;

this.RecipesReload.Invoke(this, new EventArgs());
if (!this.RecipesLoaded) this.RecipesLoaded = true;
var categoriesResult = await this.cookingRecipes.GetCategoriesList(token,
this.cancellationSource.Token);

this.Categories = new(categoriesResult.Categories);
this.Categories.Insert(0, DefaultCategory);

this.CategoriesReload.Invoke(this, new EventArgs());
}, (errorInfo) =>
{
    (this.Categories, this.CookingRecipes) = (new() { DefaultCategory }, new());
    (this.RecipesLoaded, this.AllCount) = (default, 0);

    this.RecipesReload.Invoke(this, new EventArgs());
    this.CategoriesReload.Invoke(this, new EventArgs());
});
protected virtual async Task DeletePublishedCommandHandler(int id) => await
userManager.SendRequest(async (token) =>
{
    await this.cookingRecipes.DeleteRecipeInfo(new
RequestInfo<DeleteRecipeRequestModel>()
    {
        RequestModel = new DeleteRecipeRequestModel() { Id = id },
        CancelToken = this.cancellationSource.Token, ProfileToken = token,
    });
    await this.GetPublishedListCommandHandler();
    this.DisplayInfo.Invoke(this, "Рецепт удален");
});
public ObservableCollection<GetRecipeResponseModel> CookingRecipes { get;
protected set; } = new();
public ObservableCollection<string> Categories { get; protected set; } = new()
{ DefaultCategory };

private protected string textFilter = string.Empty;
public string TextFilter { get => this.textFilter; set { this.textFilter = value;
OnPropertyChanged(); } }
public string Category { get; set; } = DefaultCategory;

private protected int allCount = default!;
public int AllCount { get => this.allCount; set { this.IsEmpty = (this.allCount =
value) <= 0; OnPropertyChanged(); } }

private protected bool isEmpty = default!;
public bool IsEmpty { get => this.isEmpty; set { this.isEmpty = value;
OnPropertyChanged(); } }

private protected bool isLoading = default;
public bool IsLoading { get => this.isLoading; set { this.isLoading = value;
OnPropertyChanged(); } }

private protected bool recipesLoaded = default;
public bool RecipesLoaded { get => this.recipesLoaded; set { this.recipesLoaded =
value; OnPropertyChanged(); } }

public virtual void OnPropertyChanged([CallerMemberName] string name = "")
{
    PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));
}

```

```

    }
}

```

Файл «RecipesListViewModel.cs»

```

using MauiLabs.View.Services.ApiModels.Commons.RecipeModels;
using MauiLabs.View.Services.ApiModels.RecipeModels.CookingRecipe.Requests;
using MauiLabs.View.Services.ApiModels.RecipeModels.CookingRecipe.Responses;
using MauiLabs.View.Services.Commons;
using MauiLabs.View.Services.Interfaces;
using Microsoft.Maui.Graphics;
using Microsoft.Maui.Storage;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Input;

namespace MauiLabs.View.Commons.ViewModels.RecipesViewModels
{
    public partial class RecipesListViewModel : INotifyPropertyChanged
    {
        public static readonly string DefaultRecipeImage =
            $"MauiLabs.View.Resources.Images.Recipe.defaultrecipe.jpg";
        public static readonly string DefaultSortingType = "По дате";
        public static readonly string DefaultCategory = "Любая категория";
        public static readonly int RecordsOnPage = 5;

        protected internal readonly ICookingRecipes recipeService = default!;
        protected internal CancellationTokenSource cancellationSource = new();

        public ICommand CancelCommand { get; protected set; } = default!;
        public ICommand GetCategoriesListCommand { get; protected set; } = default!;
        public ICommand GetRecipesListCommand { get; protected set; } = default!;

        public event EventHandler CategoriesReload = delegate { };
        public event EventHandler RecipesReload = delegate { };
        public event PropertyChangedEventHandler PropertyChanged = default!;
        public RecipesListViewModel(ICookingRecipes recipeService) : base()
        {
            this.recipeService = recipeService;
            this.GetCategoriesListCommand = new Command(() =>
            {
                this.LaunchCancelableTask(() => this.GetCategoriesListCommandHandler());
            });
            this.GetRecipesListCommand = new Command(() =>
            {
                this.LaunchCancelableTask(() => this.GetRecipesListCommandHandler());
            });
            this.CancelCommand = new Command(this.CancelCommandHandler);
        }
        protected virtual async void CancelCommandHandler(object sender) => await
Task.Run(() =>
{
    if (this.isLoading == false) return;
    this.cancellationSource.Cancel();

    this.cancellationSource = new CancellationTokenSource();
    this.isLoading = default;
}

```



```

    });
    protected async void LaunchCancelableTask(Func<Task> cancelableTask) => await
Task.Run(async () =>
{
    this.IsLoading = true; await cancelableTask.Invoke();
    this.IsLoading = false;
});
    private Task<byte[]> FileToByteArray(string filename)
    {
        using (var fileStream =
Assembly.GetExecutingAssembly().GetManifestResourceStream(filename))
        {
            using var binaryReader = new BinaryReader(fileStream);
            return Task.FromResult(binaryReader.ReadBytes((int)fileStream.Length));
        }
    }
    protected virtual async Task GetRecipesListCommandHandler() => await
UserManager.SendRequest(async (token) =>
{
    var requestResult = await this.recipeService.GetRecipesList(new
RequestInfo<GetRecipesListRequestModel>()
    {
        RequestModel = new GetRecipesListRequestModel()
        {
            Take = (RecordsOnPage * (this.PageIndex - 1)) + RecordsOnPage,
            Skip = RecordsOnPage * (this.PageIndex - 1),
            SortingType = this.SortingTypeConverter.Invoke(this.SortingType),
            TextFilter = this.TextFilter == string.Empty ? null : this.TextFilter,
            Category = this.Category == DefaultCategory ? null : this.Category,
        },
        CancelToken = this.cancellationSource.Token, ProfileToken = token,
    });
    foreach (var recipeRecord in requestResult.Recipes)
    {
        recipeRecord.Image = recipeRecord.Image.Length != 0
            ? recipeRecord.Image : await this.FileToByteArray(DefaultRecipeImage);
    }
    this.CookingRecipes = new(requestResult.Recipes);
    (this.AllCount, this.PageCount) = (requestResult.AllCount,
(int)Math.Ceiling(requestResult.AllCount / (double)RecordsOnPage));

    this.RecipesReload.Invoke(this, new EventArgs());
}, (errorInfo) =>
{
    (this.CookingRecipes, this.PageCount, this.PageIndex, this.AllCount) = (new(),
1, 1, 0);

    this.RecipesReload.Invoke(this, new EventArgs());
});
    protected virtual async Task GetCategoriesListCommandHandler() => await
UserManager.SendRequest(async (token) =>
{
    var categoriesResult = await this.recipeService.GetCategoriesList(token,
this.cancellationSource.Token);
    this.Categories = new(categoriesResult.Categories);
    this.Categories.Insert(0, DefaultCategory);

    this.CategoriesReload.Invoke(this, new EventArgs());
}, (errorInfo) =>
{
    this.Categories = new() { DefaultCategory };
    this.CategoriesReload.Invoke(this, new EventArgs());
});
    public ObservableCollection<GetRecipeResponseModel> CookingRecipes { get;
protected set; } = new();

```



```

        public ObservableCollection<string> Categories { get; protected set; } = new()
        { DefaultCategory };

        public Func<string, RecipeSortingType> SortingTypeConverter = delegate (string
        sortingName)
        {
            return sortingName switch
            {
                "По дате" => RecipeSortingType.ByDate, "По рейтингу" =>
RecipeSortingType.ByRating,
                "По названию" => RecipeSortingType.ByName, _ => RecipeSortingType.ByDate,
            };
        };
        public ObservableCollection<string> SortingTypes = new() { DefaultSortingType, "По
        рейтингу", "По названию" };

        private protected int pageIndex = default!;
        public int PageIndex { get => this.pageIndex; set { this.pageIndex = value;
        OnPropertyChanged(); } }

        private protected int pageCount = default!;
        public int PageCount { get => this.pageCount; set { this.pageCount = value;
        OnPropertyChanged(); } }

        private protected string textFilter = string.Empty;
        public string TextFilter { get => this.textFilter; set { this.textFilter = value;
        OnPropertyChanged(); } }
        public string Category { get; set; } = DefaultCategory;
        public string SortingType { get; set; } = DefaultSortingType;

        private protected int allCount = default!;
        public int AllCount { get => this.allCount; set { this.IsEmpty = (this.allCount =
        value) <= 0; OnPropertyChanged(); } }

        private protected bool isEmpty = default!;
        public bool IsEmpty { get => this.isEmpty; set { this.isEmpty = value;
        OnPropertyChanged(); } }

        private protected bool isLoading = default!;
        public bool IsLoading { get => this.isLoading; set { this.isLoading = value;
        OnPropertyChanged(); } }

        public virtual void OnPropertyChanged([CallerMemberName] string name = "")
        {
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));
        }
    }
}

```

Файл «AuthorizationPage.xaml»

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:custom="clr-namespace:MauiLabs.View.Commons.ContentViews"
    xmlns:system="clr-namespace:System;assembly=netstandard"
    xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"
    x:Class="MauiLabs.View.Pages.ProfilePages.AuthorizationPage"
    IconImageSource="/Profile/login.png"
    Title="Авторизация" Shell.BackgroundColor="{x:StaticResource Key=FirstColor}">
    <ContentPage.Resources>
        <ResourceDictionary>
            <toolkit:InvertedBoolConverter x:Key="InvertedBoolConverter" />
            <Style x:Key="ButtonStyle" TargetType="Button">
                <Setter Property="CornerRadius" Value="16" />
            </Style>
        </ResourceDictionary>
    </ContentPage.Resources>

```

```

        <Setter Property="BackgroundColor" Value="{x:StaticResource
Key=FirstColor}" />
        <Setter Property="BorderColor" Value="White" />
        <Setter Property="FontSize" Value="18"/>
        <Setter Property="TextColor" Value="White"/>
        <Setter Property="BorderWidth" Value="2" />
    </Style>
    <Style x:Key="TextFieldStyle" TargetType="Entry">
        <Setter Property="PlaceholderColor" Value="{x:StaticResource
Key=FirstColor}"/>
        <Setter Property="BackgroundColor" Value="White" />
        <Setter Property="FontSize" Value="18" />
        <Setter Property="TextColor" Value="{x:StaticResource Key=SecondColor}" />
    </Style>
</ResourceDictionary>
</ContentPage.Resources>
<Grid>
    <Image Source="background.png" Aspect="AspectFill"/>
    <ScrollView Orientation="Vertical" BackgroundColor="Transparent"
x:Name="PageScroller"
        IsEnabled="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}}">

        <StackLayout x:Name="LoginPanel" Opacity="0.0" Scale="1.5"
Orientation="Vertical" HorizontalOptions="Fill" Margin="30, 50" >
            <Label HorizontalTextAlignment="Center"
                TextColor="White" FontSize="26" FontAttributes="Bold"
                HorizontalOptions="Center" WidthRequest="300">
                <Label.FormattedText>
                    <FormattedString>
                        <Span Text="Укажите данные для" TextColor="White"/>
                        <Span Text="{x:Static system:Environment.NewLine}"/>
                        <Span Text="Входа в аккаунт" TextColor="{x:StaticResource
FirstColor}" TextDecorations="Underline"/>
                    </FormattedString>
                </Label.FormattedText>
            </Label>
            <FlexLayout AlignItems="Center" HorizontalOptions="Center"
Direction="Column">
                <Border StrokeThickness="2" MaximumWidthRequest="400" Padding="30"
                    BackgroundColor="{x:StaticResource Key=FirstColor}"
HeightRequest="280" Margin="0, 30, 0, 20"
                    FlexLayout.AlignSelf="Stretch">
                    <Border.StrokeShape>
                        <RoundRectangle CornerRadius="20" />
                    </Border.StrokeShape>
                    <Border.Stroke>
                        <SolidColorBrush Color="White" />
                    </Border.Stroke>

                    <VerticalStackLayout Spacing="10">
                        <custom:ValidationEntryView x:Name="LoginTextField"
ErrorText="Значение от 5 до 50 символов"
                            IsValidated="{x:Binding IsLoginValid}" TextValue="{x:Binding
UserLogin}"
                            CanInput="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}}">
                                DefaultText="Укажите логин" LabelText="Логин:" />
                        <custom:ValidationEntryView x:Name="PasswordTextField"
ErrorText="Значение от 5 до 50 символов"
                            IsValidated="{x:Binding IsPasswordValid}"
TextValue="{x:Binding UserPassword}"
                            CanInput="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}}">

```

```

                DefaultText="Укажите пароль" LabelText="Пароль:"
IsHidden="True"/>
            </VerticalStackLayout>

            </Border>
            <Button x:Name="LoginButton" Clicked="LoginButton_Clicked"
                FlexLayout.AlignSelf="Stretch" Text="Войти в профиль"
HeightRequest="60" MaximumWidthRequest="400"
                Style="{x:StaticResource Key=ButtonStyle}" Command="{Binding
AuthorizeCommand}"
                IsEnabled="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}}"/>
            </FlexLayout>
        </StackLayout>
    </ScrollView>
    <custom:LoadingContentView IsLoading="{x:Binding IsLoading}"
CancelCommand="{x:Binding CancelCommand}"/>
</Grid>

</ContentPage>

using MauiLabs.View.Commons.ViewModels.ProfilesViewModels;
using MauiLabs.View.Services.Commons;

namespace MauiLabs.View.Pages.ProfilePages;

public partial class AuthorizationPage : ContentPage
{
    protected internal readonly AuthorizationViewModel viewModel = default!;

    protected internal bool isPageLoaded = default!;
    public AuthorizationPage(AuthorizationViewModel viewModel) : base()
    {
        this.InitializeComponent();
        this.BindingContext = this.viewModel = viewModel;

        this.Loaded += delegate (object sender, EventArgs args) { this.isPageLoaded =
true; };
    }
    protected override void OnAppearing() => this.Dispatcher.Dispatch(async () =>
    {
        if (await UserManager.JwtToken() != null)
        {
            this.OnDisappearing();
            await Shell.Current.GoToAsync("//recipes", true);
        }
        await Task.Run(() => { while (!this.isPageLoaded); });
        await Task.WhenAll(new Task[]
        {
            this.LoginPanel.ScaleTo(1.0, length: 600, easing: Easing.SinInOut),
            this.LoginPanel.FadeTo(1.0, length: 600, easing: Easing.SinInOut),
        });
        (this.LoginPanel.Opacity, this.LoginPanel.Scale) = (1.0, 1.0);
    });
    protected override async void OnDisappearing()
    {
        this.viewModel.CancelCommand.Execute(null);
        (this.LoginPanel.Opacity, this.LoginPanel.Scale) = (0, 1.5);

        this.PasswordTextField.TextValue = string.Empty;
        this.LoginTextField.TextValue = string.Empty;
        await this.PageScroller.ScrollToAsync(0, 0, false);
    }
    protected virtual async void LoginButton_Clicked(object sender, EventArgs args)
    {
        if (!this.LoginTextField.IsValidated || !this.PasswordTextField.IsValidated)
        {

```

```

        await this.DisplayAlert("Произошла ошибка", "Неверно заполнены поля", "Назад");
    }
}
}

```

Файл «BookmarksListPage.xaml»

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MauiLabs.View.Pages.ProfilePages.BookmarksListPage"
    xmlns:custom="clr-namespace:MauiLabs.View.Commons.ContentViews"
    xmlns:system="clr-namespace:System;assembly=netstandard"
    xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"
    Title="Список заметок" x:Name="BookmarksPage"
    Shell.BackgroundColor="{x:StaticResource Key=FirstColor}">
    <ContentPage.Resources>
        <toolkit:InvertedBoolConverter x:Key="InvertedBoolConverter" />
    </ContentPage.Resources>
    <Grid>
        <Image Source="background.png" Aspect="AspectFill"/>
        <VerticalStackLayout>
            <Border BackgroundColor="{x:StaticResource FirstColor}" StrokeThickness="2"
                Stroke="White" Padding="16" >
                <Border.StrokeShape>
                    <RoundRectangle CornerRadius="0, 0, 24, 24" />
                </Border.StrokeShape>
                <VerticalStackLayout Spacing="5">
                    <Label TextColor="White" FontSize="14" Text="Возможные действия:"
                        TextDecorations="Underline"/>
                    <ScrollView Orientation="Vertical" HeightRequest="96"
                        VerticalScrollBarVisibility="Always">
                        <Grid ColumnDefinitions="0.8*, 0.2*" RowDefinitions="*, *"
                            ColumnSpacing="10" RowSpacing="10">
                            <Border Stroke="White" Padding="2" BackgroundColor="White"
                                Grid.Column="0">
                                    <Border.StrokeShape>
                                        <RoundRectangle CornerRadius="10"/>
                                    </Border.StrokeShape>
                                    <Entry PlaceholderColor="{x:StaticResource FirstColor}"
                                        TextColor="{x:StaticResource SecondColor}"
                                        FontSize="16" Placeholder="Введите название рецепта"
                                        BackgroundColor="White" ClearButtonVisibility="WhileEditing"
                                        Text="{x:Binding TextFilter, Source={x:Reference
                                            BookmarksPage}}" x:Name="FilterTextField"
                                        MaxLength="50" Completed="FilterTextField_Completed"
                                        IsEnabled="{Binding IsLoading, Converter={StaticResource
                                            InvertedBoolConverter}, Source={x:Reference BookmarksPage}}"/>
                                    </Border>
                                    <ImageButton Grid.Column="2" Source="reloadicon.png"
                                        BorderColor="White" CornerRadius="10" BorderWidth="2" Grid.Row="0"
                                        Padding="8" HeightRequest="{OnIdiom Desktop=40, Phone=50}"
                                        x:Name="RefreshButton"
                                        BackgroundColor="{x:StaticResource FirstColor}"
                                        Clicked="RefreshButton_Clicked"
                                        IsEnabled="{Binding IsLoading, Converter={StaticResource
                                            InvertedBoolConverter}, Source={x:Reference BookmarksPage}}"/>
                                <VerticalStackLayout Spacing="5" Grid.Column="0" Grid.Row="1">
                                    <Border BackgroundColor="Transparent" Padding="2"
                                        Stroke="White" StrokeThickness="2">
                                        <Border.StrokeShape>
                                            <RoundRectangle CornerRadius="10" />
                                        </Border.StrokeShape>
                                        <Picker x:Name="CategoriesPicker" FontSize="16"
                                            WidthRequest="240" HorizontalTextAlignment="Start"

```

```

SelectedIndexChanged="CategoriesPicker_SelectedIndexChanged"/>
    </Border>
</VerticalStackLayout>

    </Grid>
</ScrollView>
</VerticalStackLayout>
</Border>
<Grid x:Name="BookmarksListPanel">
    <VerticalStackLayout IsVisible="{x:Binding IsEmpty,
Converter={StaticResource InvertedBoolConverter}, Source={x:Reference BookmarksPage}}">
        <ScrollView Orientation="Vertical" BackgroundColor="Transparent"
x:Name="PageScroller" HeightRequest="500"
            <CollectionView.Resources>
                <ResourceDictionary>
                    <toolkit:ByteArrayToImageSourceConverter
x:Key="ByteArrayToImageConverter" />
                    <Style x:Key="CardBorderStyle" TargetType="Border">
                        <Setter Property="StrokeThickness" Value="2"/>
                        <Setter Property="BackgroundColor"
Value="{x:StaticResource Key=FirstColor}"/>
                        <Setter Property="Margin" Value="16, 10"/>
                        <Setter Property="Padding" Value="10"/>
                        <Setter Property="StrokeShape">
                            <Setter.Value>
                                <RoundRectangle CornerRadius="16" />
                            </Setter.Value>
                        </Setter>
                        <Setter Property="Stroke">
                            <Setter.Value>
                                <SolidColorBrush Color="White" />
                            </Setter.Value>
                        </Setter>
                        <Setter Property="HorizontalOptions" Value="Fill"/>
                        <Setter Property="HeightRequest" Value="120" />
                    </Style>
                </ResourceDictionary>
            </CollectionView.Resources>
            <CollectionView.ItemTemplate>
                <DataTemplate>
                    <Grid>
                        <VisualStateManager.VisualStateGroups>
                            <VisualStateGroup Name="CommonStates">
                                <VisualState Name="Normal" />
                                <VisualState Name="Selected">
                                    <VisualState.Setters>
                                        <Setter Property="BackgroundColor"
Value="Transparent" />
                                    </VisualState.Setters>
                                </VisualState>
                            </VisualStateGroup>
                        </VisualStateManager.VisualStateGroups>
                        <Border Style="{StaticResource CardBorderStyle}">
                            <Grid ColumnSpacing="10">
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="Auto" />
                                    <ColumnDefinition Width="*" />
                                    <ColumnDefinition Width="Auto"/>
                                </Grid.ColumnDefinitions>
                            </Grid>
                        </Border>
                    </Grid>
                </DataTemplate>
            </CollectionView.ItemTemplate>
        </VerticalStackLayout>
    </Grid>
</Page>

```

```

        </Grid.ColumnDefinitions>
        <Border Grid.Column="0" HeightRequest="90"
WidthRequest="90" StrokeThickness="2">
            <Border.StrokeShape>
                <RoundRectangle CornerRadius="16" />
            </Border.StrokeShape>
            <Border.Stroke>
                <SolidColorBrush Color="White" />
            </Border.Stroke>
            <Image BackgroundColor="White"
Aspect="AspectFill" HeightRequest="86" WidthRequest="86"
                HorizontalOptions="Center"
VerticalOptions="Center"
                Source="{Binding Image,
Converter={x:StaticResource ByteArrayToImageConverter}}"/>
            </Border>
            <StackLayout Grid.Column="1"
Orientation="Vertical" Spacing="5" HorizontalOptions="Fill">
                <Label Text="{Binding Name}"
TextColor="White" FontSize="18"
                TextDecorations="Underline"
                <Label Text="{Binding Description}"
LineBreakMode="TailTruncation"/>
                <HorizontalStackLayout Spacing="10">
                    <custom:RatingStarView
                        NullIcon="starregular.png"
ValueIcon="starsolid.png" MaxValue="5" Value="{x:Binding Rating}">
                    </custom:RatingStarView>
                    <Label FontSize="14" TextColor="White"
LineBreakMode="TailTruncation">
                        <Label.FormattedText>
                            <FormattedString>
                                <Span Text="Оценка: "/>
                                <Span Text="{Binding Rating,"
StringFormat='{0:F2}}'"/>
                                </FormattedString>
                            </Label.FormattedText>
                        </Label>
                    </HorizontalStackLayout>
                    <Label FontSize="14" TextColor="White"
LineBreakMode="TailTruncation">
                        <Label.FormattedText>
                            <FormattedString>
                                <Span Text="Категория: "/>
                                <Span Text="{Binding Category,"
StringFormat='{0:F2}}'"/>
                                </FormattedString>
                            </Label.FormattedText>
                        </Label>
                    </StackLayout>
                <ImageButton Grid.Column="2"
Source="clearicon.png"
                HeightRequest="{OnIdiom Desktop=40,
Phone=50}" WidthRequest="{OnIdiom Desktop=40, Phone=50}"
                Command="{x:Binding DeleteBookmarkCommand,"
Source={x:Reference BookmarksPage}}"
                CommandParameter="{x:Binding Id}"
BorderWidth="2" Margin="1" BorderColor="White" CornerRadius="10" Padding="8"
                IsEnabled="{Binding IsLoading,"
Converter={StaticResource InvertedBoolConverter}, Source={x:Reference BookmarksPage}}"/>
            </Grid>
        </Border>
    </Grid>
</DataTemplate>

```

```

        </CollectionView.ItemTemplate>
    </CollectionView>
</ScrollView>
</VerticalStackLayout>
<Border BackgroundColor="{x:StaticResource FirstColor}" Stroke="White"
StrokeThickness="3" Padding="30"
    HeightRequest="200" WidthRequest="200" IsVisible="{x:Binding IsEmpty,
Source={x:Reference BookmarksPage}}">
    HorizontalOptions="Center" VerticalOptions="Center" Margin="0, 120">
        <Border.StrokeShape>
            <RoundRectangle CornerRadius="16" />
        </Border.StrokeShape>
        <VerticalStackLayout Spacing="10">
            <Image Source="nothing.png" HeightRequest="90" WidthRequest="90"/>
            <Label Text="Список нуст" FontSize="20" TextColor="White"
BackgroundColor="Transparent" FontAttributes="Bold"
                VerticalOptions="Center" HorizontalOptions="Center"/>
        </VerticalStackLayout>
    </Border>
</Grid>
</VerticalStackLayout>
<custom:LoadingContentView IsLoading="{x:Binding IsLoading, Source={x:Reference
BookmarksPage}}">
    CancelCommand="{x:Binding CancelCommand,
Source={x:Reference BookmarksPage}}"/>
</Grid>
</ContentPage>

```

```

using MauiLabs.View.Pages.RecipePages;
using MauiLabs.View.Services.ApiModels.ProfileModels.Bookmarks.Requests;
using MauiLabs.View.Services.ApiModels.ProfileModels.Bookmarks.Responses;
using MauiLabs.View.Services.ApiModels.RecipeModels.CookingRecipe.Requests;
using MauiLabs.View.Services.ApiModels.RecipeModels.CookingRecipe.Responses;
using MauiLabs.View.Services.Commons;
using MauiLabs.View.Services.Interfaces;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Reflection;
using System.Windows.Input;

namespace MauiLabs.View.Pages.ProfilePages;

public partial class BookmarksListPage : ContentPage, INotifyPropertyChanged
{
    protected internal readonly INavigationService navigationService = default!;
    protected internal readonly IBookmarksList bookmarksService = default!;
    protected internal readonly ICookingRecipes cookingRecipes = default!;
    protected internal CancellationTokenSource cancellationSource = new();

    public static readonly string DefaultRecipeImage =
        $"MauiLabs.View.Resources.Images.Recipe.defaultrecipe.jpg";
    public static readonly string DefaultCategory = "Любая категория";

    public ICommand GetBookmarksListCommand { get; protected set; } = default!;
    public ICommand DeleteBookmarkCommand { get; protected set; } = default!;
    public ICommand CancelCommand { get; protected set; } = default!;

    public event EventHandler CategoriesReload = delegate { };
    public event EventHandler RecipesReload = delegate { };
    public BookmarksListPage(IBookmarksList bookmarksService, ICookingRecipes
cookingRecipes, INavigationService navigationService) : base()
    {
        this.InitializeComponent();
    }
}

```



```

        (this.bookmarksService, this.navigationService, this.cookingRecipes) =
(bookmarksService, navigationService, cookingRecipes);
        this.RecipesReload += (sender, args) => this.Dispatcher.Dispatch(() =>
this.RecipesListView.ItemsSource = this.CookingRecipes);
        this.CategoriesReload += (sender, args) => this.Dispatcher.Dispatch(() =>
        {
            this.CategoriesPicker.ItemsSource = this.Categories;
            if (this.CategoriesPicker.Items.Count > 0)
            {
                if (!this.Categories.Contains(this.Category))
this.CategoriesPicker.SelectedIndex = default!;
                else this.CategoriesPicker.SelectedIndex =
this.CategoriesPicker.Items.IndexOf(this.Category);
            }
        });
        this.GetBookmarksListCommand = new Command(() => this.LaunchCancelableTask(() =>
this.GetBookmarksListCommandHandler()));
        this.DeleteBookmarkCommand = new Command<int>(async (id) =>
        {
            if (await this.DisplayAlert("Подтверждение", "Удалить данную заметку?", "Ок",
"Назад"))
            {
                this.LaunchCancelableTask(() => this.DeleteBookmarkCommandHandler(id));
            }
        });
        this.CancelCommand = new Command(this.CancelCommandHandler);
    }
    private protected string textFilter = string.Empty;
    public string TextFilter { get => this.textFilter; set { this.textFilter = value;
OnPropertyChanged(); } }
    public string Category { get; set; } = DefaultCategory;

    private protected int allCount = default!;
    public int AllCount { get => this.allCount; set { this.IsEmpty = (this.allCount =
value) <= 0; OnPropertyChanged(); } }

    private protected bool isEmpty = default!;
    public bool IsEmpty { get => this.isEmpty; set { this.isEmpty = value;
OnPropertyChanged(); } }

    private protected bool isLoading = default;
    public bool IsLoading { get => this.isLoading; set { this.isLoading = value;
OnPropertyChanged(); } }

    public ObservableCollection<GetRecipeResponseModel> CookingRecipes { get; protected
set; } = new();
    public ObservableCollection<string> Categories { get; protected set; } = new()
{ DefaultCategory };

    protected virtual async void CancelCommandHandler(object sender) => await Task.Run(()
=>
    {
        if (this.isLoading == false) return;
        this.cancellationSource.Cancel();

        this.cancellationSource = new CancellationTokentSource();
        (this.IsLoading) = (default);
    });
    protected async void LaunchCancelableTask(Func<Task> cancelableTask) => await
Task.Run(async () =>
    {
        this.IsLoading = true; await cancelableTask.Invoke();
        this.IsLoading = false;
    });
    public virtual byte[] FileToByteArray(string filename)

```



```

{
    using (var fileStream =
Assembly.GetExecutingAssembly().GetManifestResourceStream(filename))
    {
        using var binaryReader = new BinaryReader(fileStream);
        return binaryReader.ReadBytes((int)fileStream.Length);
    }
}
protected virtual async Task GetBookmarksListCommandHandler() => await
userManager.SendRequest(async (token) =>
{
    var requestResult = await this.bookmarksService.GetBookmarks(new
RequestInfo<GetBookmarksRequestModel>()
{
    RequestModel = new GetBookmarksRequestModel()
    {
        TextFilter = this.TextFilter == string.Empty ? null : this.TextFilter,
        Category = this.Category == DefaultCategory ? null : this.Category,
    },
    CancelToken = this.cancellationSource.Token, ProfileToken = token,
});
foreach (var bookmarkrecord in requestResult.Bookmarks)
{
    bookmarkrecord.Recipe.Image = bookmarkrecord.Recipe.Image.Length != 0
        ? bookmarkrecord.Recipe.Image : this.FileToByteArray(DefaultRecipeImage);
}
this.CookingRecipes = new(requestResult.Bookmarks.Select(p => p.Recipe));
this.AllCount = requestResult.AllCount;

this.RecipesReload.Invoke(this, new EventArgs());
var categoriesResult = await this.cookingRecipes.GetCategoriesList(token,
this.cancellationSource.Token);

this.Categories = new(categoriesResult.Categories);
this.Categories.Insert(0, DefaultCategory);

this.CategoriesReload.Invoke(this, new EventArgs());
}, (errorInfo) =>
{
    (this.Categories, this.CookingRecipes) = (new() { DefaultCategory }, new());
    this.AllCount = default!;

    this.RecipesReload.Invoke(this, new EventArgs());
    this.CategoriesReload.Invoke(this, new EventArgs());
});
protected virtual async Task DeleteBookmarkCommandHandler(int id) => await
userManager.SendRequest(async (token) =>
{
    await this.bookmarksService.DeleteBookmark(new
RequestInfo<DeleteBookmarkRequestModel>()
{
    RequestModel = new DeleteBookmarkRequestModel() { RecipeId = id },
    CancelToken = this.cancellationSource.Token, ProfileToken = token,
});
await this.GetBookmarksListCommandHandler();
this.Dispatcher.Dispatch(async () => await this.DisplayAlert("Успешное действие",
"Заметка удалена", "Назад"));
});

protected virtual async void RecipesListView_SelectionChanged(object sender,
SelectionChangedEventArgs args)
{
    if (this.IsLoading) return;
    var queryParam = new Dictionary<string, object>()
    {

```

```

        ["RecipeId"] = ((GetRecipeResponseModel)this.RecipesListView.SelectedItem).Id,
        ["EnablePublisher"] = true,
    };
    await this.navigationService.NavigateToPage<RecipeInfoPage>(Shell.Current,
queryParam);
}
protected virtual void RefreshButton_Clicked(object sender, EventArgs args) =>
this.GetBookmarksListCommand.Execute(null);
protected virtual void CategoriesPicker_SelectedIndexChanged(object sender, EventArgs
args)
{
    if (this.CategoriesPicker.SelectedItem == null) return;
    this.Category = (this.CategoriesPicker.SelectedItem as string) ?? DefaultCategory;
}
protected virtual void FilterTextField_Completed(object sender, EventArgs args)
{
    this.GetBookmarksListCommand.Execute(null);
}
protected override void OnAppearing() => this.Dispatcher.Dispatch(() =>
{
    this.GetBookmarksListCommand.Execute(this);
    this.CategoriesPicker.ItemsSource = this.Categories;

    this.CategoriesPicker.SelectedIndex = default!;
});
protected override void OnDisappearing() => this.Dispatcher.Dispatch(async () =>
{
    this.CancelCommand.Execute(this);
    (this.Category, this.FilterTextField.Text) = (DefaultCategory, string.Empty);

    await this.PageScroller.ScrollToAsync(0, 0, false);
});
}

```

Файл «ProfileInfoPage.xaml»

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MauiLabs.View.Pages.ProfilePages.ProfileInfoPage"
    xmlns:custom="clr-namespace:MauiLabs.View.Commons.ContentViews"
    xmlns:system="clr-namespace:System;assembly=netstandard"
    xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"
    Title="Информация о профиле"
    Shell.BackgroundColor="{x:StaticResource Key=FirstColor}">
    <ContentPage.Resources>
        <ResourceDictionary>
            <toolkit:InvertedBoolConverter x:Key="InvertedBoolConverter" />
            <toolkit:ByteArrayToImageSourceConverter x:Key="ByteArrayToImageConverter" />
            <Style x:Key="ButtonStyle" TargetType="Button">
                <Setter Property="CornerRadius" Value="16" />
                <Setter Property="BackgroundColor" Value="{x:StaticResource
Key=FirstColor}" />
                <Setter Property="BorderColor" Value="White" />
                <Setter Property="FontSize" Value="18"/>
                <Setter Property="TextColor" Value="White"/>
                <Setter Property="BorderWidth" Value="2" />
            </Style>
            <Style x:Key="TextFieldStyle" TargetType="Entry">
                <Setter Property="PlaceholderColor" Value="{x:StaticResource
Key=FirstColor}" />
                <Setter Property="BackgroundColor" Value="White" />
                <Setter Property="FontSize" Value="18" />
                <Setter Property="TextColor" Value="{x:StaticResource Key=SecondColor}" />
            </Style>
            <Style x:Key="ImagePickerStyle" TargetType="ImageButton">

```

```

        <Setter Property="CornerRadius" Value="10" />
        <Setter Property="BorderColor" Value="White"/>
        <Setter Property="BorderWidth" Value="2"/>
        <Setter Property="BackgroundColor" Value="{x:StaticResource FirstColor}"/>
    </Style>
    <Style x:Key="ImageButtonStyle" TargetType="ImageButton">
        <Setter Property="HeightRequest" Value="{OnIdiom Desktop=46, Phone=68}" />
        <Setter Property="Padding" Value="{OnIdiom Desktop=10, Phone=12}" />
        <Setter Property="CornerRadius" Value="12"/>
        <Setter Property="BorderColor" Value="White"/>
        <Setter Property="BorderWidth" Value="2"/>
        <Setter Property="BackgroundColor" Value="{x:StaticResource FirstColor}"/>
    </Style>

</ResourceDictionary>
</ContentPage.Resources>
<Grid>
    <Image Source="background.png" Aspect="AspectFill"/>
    <ScrollView Orientation="Vertical" BackgroundColor="Transparent"
x:Name="PageScroller"
        Enabled="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}}">
        <StackLayout x:Name="ProfilePanel" Orientation="Vertical"
HorizontalOptions="Fill" Margin="30, 10, 30, 40"
            Opacity="0.0" Scale="1.5">

            <Border StrokeThickness="2" MaximumWidthRequest="260" Padding="20"
                BackgroundColor="{x:StaticResource Key=FirstColor}" Margin="0, 20, 0,
10"
                FlexLayout.AlignSelf="Stretch" Stroke="White">
                <Border.StrokeShape>
                    <RoundRectangle CornerRadius="20" />
                </Border.StrokeShape>
                <VerticalStackLayout Spacing="5">
                    <Label Text="Изображение профиля:" TextColor="White" FontSize="18"
Margin="4, 0, 0, 5" TextDecorations="Underline"/>
                    <Grid RowDefinitions="*">
                        <Grid.ColumnDefinitions>
                            <ColumnDefinition Width="0.6*"/>
                            <ColumnDefinition Width="0.4*"/>
                        </Grid.ColumnDefinitions>
                        <Border x:Name="ProfileImage" HeightRequest="112"
WidthRequest="112" Stroke="White"
                            StrokeThickness="2" BackgroundColor="White"
Grid.Column="0" Padding="0">
                            <Border.StrokeShape>
                                <RoundRectangle CornerRadius="20" />
                            </Border.StrokeShape>
                            <Image Aspect="AspectFill" HeightRequest="112"
WidthRequest="112"/>
                        </Border>

                        <Grid RowDefinitions="*, *" ColumnDefinitions="*" Margin="5, 5"
Grid.Column="1">
                            <ImageButton Grid.Row="0" Grid.Column="0"
                                Clicked="ImagePickerButton_Clicked"
Style="{x:StaticResource ImagePickerStyle}"
                                Source="loadimageicon.png" Margin="4" Padding="3"
WidthRequest="60" HeightRequest="46"
                                Enabled="{Binding IsLoading,
Converter={StaticResource InvertedBoolConverter}}"/>
                            <ImageButton Grid.Row="1" Grid.Column="0"
                                Clicked="ImageClearButton_Clicked"
Style="{x:StaticResource ImagePickerStyle}"

```

```

                Source="clearicon.png" WidthRequest="60"
HeightRequest="46" Margin="4" Padding="8"
                IsEnabled="{Binding IsLoading,
Converter={StaticResource InvertedBoolConverter}}"/>
            </Grid>

        </Grid>
    </VerticalStackLayout>
</Border>

    <custom:ExpanderView ButtonIcon="arrowicon.png" ExpanderHeight="340"
ButtonText="Изменить пароль"
        Margin="0, 10, 0, 10" x:Name="PasswordExpander"
MaximumWidthRequest="400" ButtonHeight="52"
        CanExpand="{Binding ProfileLoaded}">
        <custom:ExpanderView.Item>
            <Border BackgroundColor="Transparent" Padding="20, 5">

                <VerticalStackLayout Spacing="10">
                    <custom:ValidationEntryView x:Name="OldPasswordTextField"
ErrorText="Значение от 5 до 50 символов"
                        IsValidated="{x:Binding PasswordValidationState[Old]}"
TextValue="{x:Binding OldPassword}"
                        CanInput="{Binding IsLoading,
Converter={StaticResource InvertedBoolConverter}}"
                        DefaultText="Укажите старый пароль" LabelText="Старый
пароль:"
                        IsReadOnly="{Binding ProfileLoaded,
Converter={StaticResource InvertedBoolConverter}}"/>

                    <custom:ValidationEntryView x:Name="NewPasswordTextField"
ErrorText="Значение от 5 до 50 символов"
                        IsValidated="{x:Binding PasswordValidationState[New]}"
TextValue="{x:Binding NewPassword}"
                        CanInput="{Binding IsLoading,
Converter={StaticResource InvertedBoolConverter}}"
                        DefaultText="Укажите новый пароль" LabelText="Новый
пароль:"
                        IsReadOnly="{Binding ProfileLoaded,
Converter={StaticResource InvertedBoolConverter}}"/>

                    <Button x:Name="PasswordChangeButton" Command="{x:Binding
ChangePasswordCommand}"
                        IsEnabled="{Binding IsLoading,
Converter={StaticResource InvertedBoolConverter}}"
Text="Изменить" CornerRadius="10" Margin="0, 5"
TextColor="{x:StaticResource SecondColor}"
                        BackgroundColor="White" BorderColor="White"
BorderWidth="2" HeightRequest="40" FontSize="16"/>
                </VerticalStackLayout>

            </Border>
        </custom:ExpanderView.Item>
    </custom:ExpanderView>

    <Border StrokeThickness="2" MaximumWidthRequest="400" Padding="20"
BackgroundColor="{x:StaticResource Key=FirstColor}"
        Margin="0, 10, 0, 16" FlexLayout.AlignSelf="Stretch" Stroke="White">
        <Border.StrokeShape>
            <RoundRectangle CornerRadius="20" />
        </Border.StrokeShape>
        <VerticalStackLayout Spacing="10">
            <Label Text="Данные профиля:" TextColor="White" FontSize="18"
Margin="0, 0, 0, 10" TextDecorations="Underline"/>

```

```

        <custom:ValidationEntryView x:Name="NameTextField"
ErrorText="Значение от 4 до 50 символов"
        IsValidated="{x:Binding ValidationState[UserName]}"
TextValue="{x:Binding UserName}"
        CanInput="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}}"
        MinLenght="4" DefaultText="Укажите имя" LabelText="Имя
пользователя:"
        IsReadOnly="{Binding ProfileLoaded, Converter={StaticResource
InvertedBoolConverter}}"/>

        <custom:ValidationEntryView x:Name="SurnameTextField"
ErrorText="Значение от 4 до 50 символов"
        IsValidated="{x:Binding ValidationState[Surname]}"
TextValue="{x:Binding UserSurname}"
        CanInput="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}}"
        MinLenght="4" DefaultText="Укажите фамилию"
LabelText="Фамилия пользователя:"
        IsReadOnly="{Binding ProfileLoaded, Converter={StaticResource
InvertedBoolConverter}}"/>

        <BoxView BackgroundColor="White" Margin="5, 14" CornerRadius="1"
HeightRequest="2"/>

        <custom:ValidationEntryView x:Name="EmailTextField"
ErrorText="Значение от 5 до 100 символов"
        IsValidated="{x:Binding ValidationState[Email]}"
TextValue="{x:Binding UserEmail}"
        CanInput="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}}"
        MaxLenght="100" Regex="^([\\w\\.\\-]{4,})@([\\w\\-
]+)((\\.([\\w]{2,3})+)$"
        DefaultText="Пример: test@gmail.com" LabelText="Электронная
почта:"
        IsReadOnly="{Binding ProfileLoaded, Converter={StaticResource
InvertedBoolConverter}}"/>

        <Button x:Name="ReferenceLinkButton"
Clicked="ReferenceLinkButton_Clicked"
        Text="Ссылка для добавления друзей" HeightRequest="48"
FontSize="16"
        Style="{x:StaticResource Key=ButtonStyle}" CornerRadius="10"
Margin="0, 10"
        IsEnabled="{Binding ProfileLoaded}"/>

    </VerticalStackLayout>
</Border>

    <Grid Margin="0, 0, 0, 20" ColumnDefinitions="*, *, *" RowDefinitions="*">
        <ImageButton Grid.Column="0" Grid.Row="0" Margin="3" WidthRequest="100"
Source="deleteicon.png" Command="{x:Binding DeleteProfileCommand}"
Style="{x:StaticResource ImageButtonStyle}"
        IsEnabled="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}}"/>

        <ImageButton Grid.Column="1" Grid.Row="0" Margin="3"
WidthRequest="{OnIdiom Desktop=140, Phone=130}"
Source="reloadicon.png" Command="{x:Binding GetProfileCommand}"
Style="{x:StaticResource ImageButtonStyle}"
        IsEnabled="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}}"/>

        <ImageButton Grid.Column="2" Grid.Row="0" Margin="3" WidthRequest="100"

```

```

                Source="logouticon.png" Clicked="ExitProfile_Clicked"
Style="{x:StaticResource ImageButtonStyle}"
                IsEnabled="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}}"/>
            </Grid>

            <Button x:Name="UpdateButton" Text="Обновить данные"
                HeightRequest="54" MaximumWidthRequest="400"
                Style="{x:StaticResource Key=ButtonStyle}" Command="{Binding
UpdateProfileCommand}"
                IsEnabled="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}}"/>

        </StackLayout>
    </ScrollView>
    <custom:LoadingContentView IsLoading="{x:Binding IsLoading}"
CancelCommand="{x:Binding CancelCommand}"/>
</Grid>

</ContentPage>

using MauiLabs.View.Commons.ViewModels.ProfilesViewModels;
using MauiLabs.View.Services.Commons;
using MauiLabs.View.Services.Interfaces;
using SixLabors.ImageSharp.Formats.Png;
using SixLabors.ImageSharp.Processing;
using System.Collections;
using System.ComponentModel;
using System.IO;
using System.Runtime.CompilerServices;

namespace MauiLabs.View.Pages.ProfilePages;

public partial class ProfileInfoPage : ContentPage
{
    protected internal readonly ProfileInfoViewModel viewModel = default!;
    public virtual double ImageSize { get => 96; }

    protected internal bool isPageLoaded = default!;
    public ProfileInfoPage(ProfileInfoViewModel viewModel) : base()
    {
        this.InitializeComponent();
        this.Loaded += delegate (object sender, EventArgs args) { this.isPageLoaded =
true; };
        this.BindingContext = this.viewModel = viewModel;

        this.viewModel.DisplayAlert += (sender, message) => this.DisplayMessage("Произошла
ошибка", message);
        this.viewModel.DisplayInfo += (sender, message) => this.DisplayMessage("Успешное
действие", message);

        this.viewModel.ReloadImage += (sender, image) => this.ReloadProfileImage(image);
        this.viewModel.CheckConfirm += (message) => this.DisplayAlert("Подтверждение",
message, "Ок", "Назад");
    }
    protected virtual void DisplayMessage(string title, string message)
    {
        this.Dispatcher.Dispatch(async () => await this.DisplayAlert(title, message,
"Назад"));
    }
    protected virtual void ReloadProfileImage(byte[] image) => this.Dispatcher.Dispatch(()
=>
    {
        this.ProfileImage.Content = new Image()
        {
            Source = ImageSource.FromStream(() => new MemoryStream(image)),
            Margin = Thickness.Zero, Aspect = Aspect.AspectFill,

```

```

    });
    protected virtual async void ExitProfile_Clicked(object sender, EventArgs args)
    {
        await UserManager.LogoutUser();
        await Application.Current!.Dispatcher.DispatchAsync(async () =>
        {
            await Shell.Current.Navigation.PopToRootAsync();
            await Shell.Current.GoToAsync(UserManager.AuthorizationRoute);
        });
    }
    protected virtual async void ImagePickerButton_Clicked(object sender, EventArgs args)
    {
        if (!this.viewModel.ProfileLoaded) { this.DisplayMessage("Произошла ошибка",
"Необходимо перезагрузить профиль"); return; }
        var fileFilter = (FileResult result, string extension) =>
        {
            return result.FileName.EndsWith(extension, StringComparison.OrdinalIgnoreCase);
        };
        var pickerOption = new PickOptions() { FileTypes = FilePickerFileType.Images,
PickerTitle = "Выберите изображение" };
        try {
            var pickerResult = await FilePicker.Default.PickAsync(pickerOption);
            if (pickerResult != null && (fileFilter(pickerResult, "jpg") ||
fileFilter(pickerResult, "png")))
            {
                using var stream = await pickerResult.OpenReadAsync();
                using var image = SixLabors.ImageSharp.Image.Load(stream);
                image.Mutate(prop => prop.Resize((int)this.ImageSize, (int)this.ImageSize,
false));

                using var outputStream = new MemoryStream();
                image.Save(outputStream, new PngEncoder());
                this.ReloadProfileImage(this.viewModel.UserImage = outputStream.ToArray());
            }
        }
        catch (Exception errorInfo) { this.DisplayMessage("Произошла ошибка",
errorInfo.Message); }
    }
    protected virtual void ImageClearButton_Clicked(object sender, EventArgs args)
    {
        if (!this.viewModel.ProfileLoaded) { this.DisplayMessage("Произошла ошибка",
"Необходимо перезагрузить профиль"); return; }
        this.viewModel.UserImage = null;

        this.ReloadProfileImage(this.viewModel.FileToByteArray(ProfileInfoViewModel.DefaultProfileImage));
    }
    protected virtual async void ReferenceLinkButton_Clicked(object sender, EventArgs
args)
    {
        await Share.Default.RequestAsync(new ShareTextRequest
        {
            Text = string.Format("Используйте ссылку в приложение: {0}\n",
this.viewModel.ReferenceLink),
            Title = "Ссылка для добавление в друзья",
            Uri = @"https://github.com/mo0nchild/cs-maui-labs",
        });
    }

    protected override void OnAppearing() => this.Dispatcher.Dispatch(async() =>
    {
        this.ReloadProfileImage(this.viewModel.FileToByteArray(ProfileInfoViewModel.DefaultProfileImage));
    });

```



```

this.viewModel.GetProfileCommand.Execute(this);
await Task.Run(() => { while (!this.isPageLoaded) ; });
await Task.WhenAll(new Task[]
{
    this.ProfilePanel.ScaleTo(1.0, length: 600, easing: Easing.SinInOut),
    this.ProfilePanel.FadeTo(1.0, length: 600, easing: Easing.SinInOut),
});
(this.ProfilePanel.Opacity, this.ProfilePanel.Scale) = (1.0, 1.0);
});
protected override void OnDisappearing() => this.Dispatcher.Dispatch(async () =>
{
    this.viewModel.CancelCommand.Execute(this);
    (this.ProfilePanel.Opacity, this.ProfilePanel.Scale) = (0, 1.5);

    this.EmailTextField.TextValue = this.SurnameTextField.TextValue =
this.NameTextField.TextValue = "Не загружено";
    this.PasswordExpander.ResetExpander();
    this.OldPasswordTextField.TextValue = this.NewPasswordTextField.TextValue =
string.Empty;

this.ReloadProfileImage(this.viewModel.FileToByteArray(ProfileInfoViewModel.DefaultProfileImage));
    this.viewModel.UserImage = null;
    await this.PageScroller.ScrollToAsync(0, 0, false);
});
}

```

Файл «FriendInfoPage.xaml»

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MauiLabs.View.Pages.ProfilePages.FriendInfoPage"
    xmlns:custom="clr-namespace:MauiLabs.View.Commons.ContentViews"
    xmlns:system="clr-namespace:System;assembly=netstandard"
    xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"
    Title="Данные о дпуре" x:Name="FriendPage"
    Shell.BackgroundColor="{x:StaticResource Key=FirstColor}"
    Shell.FlyoutBehavior="Disabled">
    <ContentPage.Resources>
        <ResourceDictionary>
            <toolkit:InvertedBoolConverter x:Key="InvertedBoolConverter" />
            <toolkit:ByteArrayToImageSourceConverter x:Key="ByteArrayToImageConverter" />
            <Style x:Key="TextFieldStyle" TargetType="Entry">
                <Setter Property="PlaceholderColor" Value="{x:StaticResource
Key=FirstColor}"/>
                <Setter Property="BackgroundColor" Value="White" />
                <Setter Property="FontSize" Value="16" />
                <Setter Property="TextColor" Value="{x:StaticResource Key=SecondColor}" />
            </Style>
            <Style x:Key="TextFieldLabelStyle" TargetType="Label">
                <Setter Property="BackgroundColor" Value="Transparent" />
                <Setter Property="TextColor" Value="White"/>
                <Setter Property="FontSize" Value="14" />
            </Style>
            <Style x:Key="ImageButtonStyle" TargetType="ImageButton">
                <Setter Property="HeightRequest" Value="{OnIdiom Desktop=46, Phone=68}" />
                <Setter Property="Padding" Value="{OnIdiom Desktop=10, Phone=12}" />
                <Setter Property="CornerRadius" Value="12"/>
                <Setter Property="BorderColor" Value="White"/>
                <Setter Property="BorderWidth" Value="2"/>
                <Setter Property="BackgroundColor" Value="{x:StaticResource FirstColor}"/>
            </Style>
        </ResourceDictionary>
    </ContentPage.Resources>

```



```

<Grid>
    <Image Source="background.png" Aspect="AspectFill"/>
    <ScrollView Orientation="Vertical" BackgroundColor="Transparent"
x:Name="PageScroller"
        IsEnabled="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}, Source={x:Reference FriendPage}}">
        <StackLayout x:Name="ProfilePanel" Orientation="Vertical"
HorizontalOptions="Fill" Margin="30, 10, 30, 40" Opacity="0.0" Scale="1.5">
            <Border StrokeThickness="2" MaximumWidthRequest="260" Padding="20"
                BackgroundColor="{x:StaticResource Key=FirstColor}" Margin="0, 20, 0,
10"
                    FlexLayout.AlignSelf="Stretch" Stroke="White">
                <Border.StrokeShape>
                    <RoundRectangle CornerRadius="20" />
                </Border.StrokeShape>
                <VerticalStackLayout Spacing="5">
                    <Label Text="Изображение профиля:" TextColor="White" FontSize="18"
Margin="4, 0, 0, 5" TextDecorations="Underline"/>
                    <Grid RowDefinitions="*" ColumnDefinitions="0.6*, 0.4*">
                        <Border x:Name="ProfileImage" HeightRequest="120"
WidthRequest="120" Stroke="White"
                            StrokeThickness="2" BackgroundColor="White"
Grid.Column="0" Padding="0" Margin="0, 0, 0, 10">
                            <Border.StrokeShape>
                                <RoundRectangle CornerRadius="20" />
                            </Border.StrokeShape>
                            <Image Aspect="AspectFill" HeightRequest="120"
WidthRequest="120"/>
                        </Border>
                        <Grid RowDefinitions="*, *" ColumnDefinitions="*" Margin="5, 5"
Grid.Column="1">
                            <ImageButton Grid.Column="0" Grid.Row="1" Margin="3"
                                Source="deleteicon.png" Clicked="DeleteButton_Clicked"
Style="{x:StaticResource ImageButtonStyle}"
                                IsEnabled="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}, Source={x:Reference FriendPage}}"/>
                            <ImageButton Grid.Column="1" Grid.Row="0" Margin="3"
                                Source="reloadicon.png" Clicked="RefreshButton_Clicked"
Style="{x:StaticResource ImageButtonStyle}"
                                IsEnabled="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}, Source={x:Reference FriendPage}}"/>
                        </Grid>
                    </VerticalStackLayout>
                </Border>
            <Border StrokeThickness="2" MaximumWidthRequest="400" Padding="20"
                BackgroundColor="{x:StaticResource Key=FirstColor}"
                Margin="0, 10, 0, 16" FlexLayout.AlignSelf="Stretch" Stroke="White">
                <Border.StrokeShape>
                    <RoundRectangle CornerRadius="20" />
                </Border.StrokeShape>
                <VerticalStackLayout Spacing="10">
                    <Label Text="Данные профиля друга:" TextColor="White" FontSize="18"
Margin="0, 0, 0, 10" TextDecorations="Underline"/>
                    <VerticalStackLayout Spacing="5">
                        <Label Text="Имя пользователя:" Style="{x:StaticResource
TextFieldLabelStyle}" />
                        <Border BackgroundColor="White" Padding="2">
                            <Border.StrokeShape>
                                <RoundRectangle CornerRadius="10" />
                            </Border.StrokeShape>
                            <Entry Text="{x:Binding FriendName, Source={x:Reference
FriendPage}}">

```

```

        IsEnabled="False" x:Name="FriendNameTextField"
Style="{x:StaticResource Key=TextFieldStyle}">
    </Entry>
</Border>
</VerticalStackLayout>
<VerticalStackLayout Spacing="5">
    <Label Text="Фамилия пользователя:" Style="{x:StaticResource
TextFieldLabelStyle}" />
    <Border BackgroundColor="White" Padding="2">
        <Border.StrokeShape>
            <RoundRectangle CornerRadius="10" />
        </Border.StrokeShape>
        <Entry Text="{x:Binding FriendSurname, Source={x:Reference
FriendPage}}">
            IsEnabled="False" x:Name="FriendSurnameTextField"
Style="{x:StaticResource Key=TextFieldStyle}">
                </Entry>
            </Border>
        </VerticalStackLayout>
        <BoxView BackgroundColor="White" Margin="5, 14" CornerRadius="1"
HeightRequest="2"/>
        <VerticalStackLayout Spacing="5" Margin="0, 0, 0, 20">
            <Label Text="Email пользователя:" Style="{x:StaticResource
TextFieldLabelStyle}" />
            <Border BackgroundColor="White" Padding="2">
                <Border.StrokeShape>
                    <RoundRectangle CornerRadius="10" />
                </Border.StrokeShape>
                <Entry Text="{x:Binding FriendEmail, Source={x:Reference
FriendPage}}">
                    IsEnabled="False" x:Name="FriendEmailTextField"
Style="{x:StaticResource Key=TextFieldStyle}">
                        </Entry>
                    </Border>
                </VerticalStackLayout>
            </VerticalStackLayout>
        </Border>
        <Grid x:Name="BookmarksListPanel">
            <VerticalStackLayout IsVisible="{x:Binding IsEmpty,
Converter={StaticResource InvertedBoolConverter}, Source={x:Reference FriendPage}}">
                <ScrollView Orientation="Vertical" BackgroundColor="Transparent"
x:Name="SecondScroller" HeightRequest="300">
                    IsEnabled="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}, Source={x:Reference FriendPage}}" Margin="0, 5, 0, 0">
                        <CollectionView x:Name="RecipesListView" ItemsSource="{Binding
CookingRecipes, Source={x:Reference FriendPage}}" HorizontalOptions="Fill"
SelectionChanged="RecipesListView_SelectionChanged"
SelectionMode="Single">
                            <CollectionView.Resources>
                                <ResourceDictionary>
                                    <toolkit:ByteArrayToImageSourceConverter
x:Key="ByteArrayToImageConverter" />
                                    <Style x:Key="CardBorderStyle" TargetType="Border">
                                        <Setter Property="StrokeThickness" Value="2"/>
                                        <Setter Property="BackgroundColor"
Value="{x:StaticResource Key=FirstColor}"/>
                                        <Setter Property="Margin" Value="0, 8"/>
                                        <Setter Property="Padding" Value="10"/>
                                        <Setter Property="StrokeShape">
                                            <Setter.Value>
                                                <RoundRectangle CornerRadius="16" />
                                            </Setter.Value>

```

```

        </Setter>
        <Setter Property="Stroke">
            <Setter.Value>
                <SolidColorBrush Color="White" />
            </Setter.Value>
        </Setter>
        <Setter Property="HorizontalOptions" Value="Fill"/>
        <Setter Property="HeightRequest" Value="120" />
    </Style>
</ResourceDictionary>
</CollectionView.Resources>

<CollectionView.ItemTemplate>
    <DataTemplate>
        <Grid>
            <VisualStateManager.VisualStateGroups>
                <VisualStateGroup Name="CommonStates">
                    <VisualState Name="Normal" />
                    <VisualState Name="Selected">
                        <VisualState.Setters>
                            <Setter Property="BackgroundColor"
Value="Transparent" />
                        </VisualState.Setters>
                    </VisualState>
                </VisualStateGroup>
            </VisualStateManager.VisualStateGroups>
            <Border Style="{StaticResource CardBorderStyle}">
                <Grid ColumnSpacing="10">
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="Auto" />
                        <ColumnDefinition Width="*" />
                        <ColumnDefinition Width="Auto"/>
                    </Grid.ColumnDefinitions>
                    <Border Grid.Column="0" HeightRequest="90"
WidthRequest="90" StrokeThickness="2">
                        <Border.StrokeShape>
                            <RoundRectangle CornerRadius="16" />
                        </Border.StrokeShape>
                        <Border.Stroke>
                            <SolidColorBrush Color="White" />
                        </Border.Stroke>
                        <Image BackgroundColor="White"
Aspect="AspectFill" HeightRequest="86" WidthRequest="86"
HorizontalOptions="Center"
VerticalOptions="Center"
Source="{Binding Image,
Converter={x:StaticResource ByteArrayToImageConverter}}"/>
                    </Border>
                    <StackLayout Grid.Column="1">
                        <Label Text="{Binding Name}"
TextColor="White" FontSize="18"
TextDecorations="Underline"
LineBreakMode="TailTruncation"/>
                        <Label Text="{Binding Description}"
LineBreakMode="TailTruncation"/>
                        <HorizontalStackLayout Spacing="10">
                            <custom:RatingStarView
NullIcon="starregular.png"
ValueIcon="starsolid.png" MaxValue="5" Value="{x:Binding Rating}">
                                </custom:RatingStarView>
                            <Label FontSize="14"
TextColor="White" LineBreakMode="TailTruncation">
                                <Label.FormattedText>
                                    <FormattedString>

```



```

public partial class FriendInfoPage : ContentPage, INotifyPropertyChanged,
INavigationService.IQueryableNavigation
{
    protected internal CancellationTokenSource cancellationSource = new();
    protected internal readonly INavigationService navigationService = default!;
    protected internal readonly IBookmarksList bookmarksService = default!;

    protected internal readonly IUserProfile userProfile = default!;
    protected internal readonly IFriendsList friendProfile = default!;

    public ICommand GetBookmarksListCommand { get; protected set; } = default!;
    public ICommand CancelCommand { get; protected set; } = default!;

    public static readonly string DefaultProfileImage =
$"MauiLabs.View.Resources.Images.Profile.defaultprofile.png";
    public static readonly string DefaultRecipeImage =
$"MauiLabs.View.Resources.Images.Recipe.defaultrecipe.jpg";

    public event EventHandler RecipesReload = delegate { };
    public FriendInfoPage(IUserProfile userProfile, IBookmarksList bookmarksService,
IFriendsList friendProfile,
INavigationService navigationService) : base()
    {
        this.InitializeComponent();
        (this.userProfile, this.bookmarksService, this.friendProfile) = (userProfile,
bookmarksService, friendProfile);
        this.navigationService = navigationService;
        this.RecipesReload += (sender, args) =>
        {
            this.Dispatcher.Dispatch(() => this.RecipesListView.ItemsSource =
this.CookingRecipes);
        };
        this.CancelCommand = new Command(this.CancelCommandHandler);
    }

    private protected bool isLoading = default;
    public bool IsLoading { get => this.isLoading; set { this.isLoading = value;
OnPropertyChanged(); } }

    private protected string friendName = default!;
    public string FriendName { get => this.friendName; set { this.friendName = value;
OnPropertyChanged(); } }

    private protected string friendSurname = default!;
    public string FriendSurname { get => this.friendSurname; set { this.friendSurname =
value; OnPropertyChanged(); } }

    private protected string friendEmail = default!;
    public string FriendEmail { get => this.friendEmail; set { this.friendEmail = value;
OnPropertyChanged(); } }

    public virtual int FriendId { get; protected set; } = default!;
    public virtual int RecordId { get; protected set; } = default!;

    private protected int allCount = default!;
    public int AllCount { get => this.allCount; set { this.IsEmpty = (this.allCount =
value) <= 0; OnPropertyChanged(); } }

    private protected bool isEmpty = default!;
    public bool IsEmpty { get => this.isEmpty; set { this.isEmpty = value;
OnPropertyChanged(); } }
    public ObservableCollection<GetRecipeResponseModel> CookingRecipes { get; protected
set; } = new();

```

```

        protected async void LaunchCancelableTask(Func<Task> cancelableTask) => await
Task.Run(async () =>
{
    this.IsLoading = true; await cancelableTask.Invoke();
    this.IsLoading = false;
});
protected virtual async void CancelCommandHandler(object sender) => await Task.Run(()
=>
{
    if (this.IsLoading == false) return;

    this.cancellationSource.Cancel();
    this.cancellationSource = new CancellationTokenSource();

    (this.IsLoading) = (default);
});
public virtual byte[] FileToByteArray(string filename)
{
    using (var fileStream =
Assembly.GetExecutingAssembly().GetManifestResourceStream(filename))
    {
        using var binaryReader = new BinaryReader(fileStream);
        return binaryReader.ReadBytes((int)fileStream.Length);
    }
}
protected virtual void ReloadProfileImage(byte[] image) => this.Dispatcher.Dispatch(()
=>
{
    this.ProfileImage.Content = new Image()
    {
        Source = ImageSource.FromStream(() => new MemoryStream(image)),
        Margin = Thickness.Zero, Aspect = Aspect.AspectFill,
    };
});
protected virtual async Task GetFriendInfo() => await UserManager.SendRequest(async
(token) =>
{
    var requestResult = await this.userProfile.GetProfileInfo(token, this.FriendId,
this.cancellationSource.Token);
    if (requestResult.Image.Length == 0)
    {
        this.ReloadProfileImage(this.FileToByteArray(DefaultProfileImage));
    }
    else this.ReloadProfileImage(requestResult.Image);
    (this.FriendName, this.FriendSurname) = (requestResult.Name,
requestResult.Surname);
    this.FriendEmail = requestResult.Email;

    var bookmarkRequest = await this.bookmarksService.GetBookmarksById(new
RequestInfo<GetBookmarksByIdRequestModel>()
    {
        RequestModel = new GetBookmarksByIdRequestModel() { ProfileId =
this.FriendId },
        CancelToken = this.cancellationSource.Token, ProfileToken = token,
    });
    foreach (var bookmarkrecord in bookmarkRequest.Bookmarks)
    {
        bookmarkrecord.Recipe.Image = bookmarkrecord.Recipe.Image.Length != 0
            ? bookmarkrecord.Recipe.Image : this.FileToByteArray(DefaultRecipeImage);
    }
    this.CookingRecipes = new(bookmarkRequest.Bookmarks.Select(p => p.Recipe));
    this.AllCount = bookmarkRequest.AllCount;

    this.RecipesReload.Invoke(this, new EventArgs());
}, async (errorInfo) =>

```

```

{
    await Shell.Current.Navigation.PopAsync(animated: true);
});

protected virtual async Task DeleteFriend() => await UserManager.SendRequest(async
(token) =>
{
    if(await this.DisplayAlert("Подтверждение", "Удалить друга из списка?", "Ок",
"Назад"))
    {
        await this.friendProfile.DeleteFriend(new
RequestInfo<DeleteFriendRequestModel>()
{
            RequestModel = new DeleteFriendRequestModel() { RecordId = this.RecordId },
            CancelToken = this.cancellationSource.Token, ProfileToken = token,
        });
    }
});

protected virtual async void RecipesListView_SelectionChanged(object sender,
SelectionChangedEventArgs args)
{
    if (this.IsLoading) return;
    var queryParams = new Dictionary<string, object>()
    {
        ["RecipeId"] = ((GetRecipeResponseModel)this.RecipesListView.SelectedItem).Id,
        ["EnablePublisher"] = true,
    };
    await this.navigationService.NavigateToPage<RecipeInfoPage>(Shell.Current,
queryParams);
}

protected virtual async void DeleteButton_Clicked(object sender, EventArgs args)
{
    await this.Dispatcher.DispatchAsync(async () =>
    {
        this.LaunchCancelableTask(() => this.DeleteFriend());
        await Shell.Current.Navigation.PopAsync(animated: true);
    });
}

protected virtual void RefreshButton_Clicked(object sender, EventArgs args) =>
this.OnAppearing();

protected override void OnAppearing() => this.Dispatcher.Dispatch(async () =>
{
    this.ReloadProfileImage(this.FileToByteArray(DefaultProfileImage));
    this.LaunchCancelableTask(() => this.GetFriendInfo());
    await Task.WhenAll(new Task[]
    {
        this.ProfilePanel.ScaleTo(1.0, length: 600, easing: Easing.SinInOut),
        this.ProfilePanel.FadeTo(1.0, length: 600, easing: Easing.SinInOut),
    });
    (this.ProfilePanel.Opacity, this.ProfilePanel.Scale) = (1.0, 1.0);
});
public virtual void SetNavigationQuery(IDictionary<string, object> queries)
{
    (this.FriendId, this.RecordId) = ((int)queries["FriendId"],
(int)queries["RecordId"]);
}
protected override void OnDisappearing() => base.OnDisappearing();
}

```

Файл «CommentsListPage.xaml»

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
x:Class="MauiLabs.View.Pages.RecipePages.CommentsListPage"

```



```

xmlns:custom="clr-namespace:MauiLabs.View.Commons.ContentViews"
xmlns:system="clr-namespace:System;assembly=netstandard"
xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"
Title="Список комментариев" x:Name="CommentsPage"
Shell.BackgroundColor="{x:StaticResource Key=FirstColor}"
Shell.FlyoutBehavior="Disabled">
<ContentPage.Resources>
<ResourceDictionary>
<toolkit:InvertedBoolConverter x:Key="InvertedBoolConverter" />
<toolkit:ByteArrayToImageSourceConverter x:Key="ByteArrayToImageConverter" />
<Style x:Key="TextFieldStyle" TargetType="Entry">
<Setter Property="PlaceholderColor" Value="{x:StaticResource
Key=FirstColor}"/>
<Setter Property="BackgroundColor" Value="White" />
<Setter Property="FontSize" Value="18" />
<Setter Property="TextColor" Value="{x:StaticResource Key=SecondColor}" />
</Style>
<Style x:Key="TextFieldLabelStyle" TargetType="Label">
<Setter Property="BackgroundColor" Value="Transparent" />
<Setter Property="TextColor" Value="White"/>
<Setter Property="FontSize" Value="14" />
</Style>
<Style x:Key="PageButtonStyle" TargetType="Button">
<Setter Property="BackgroundColor" Value="{x:StaticResource FirstColor}"/>
<Setter Property="TextColor" Value="White"/>
<Setter Property="BorderColor" Value="White"/>
<Setter Property="BorderWidth" Value="2"/>
<Setter Property="CornerRadius" Value="10"/>
<Setter Property="HeightRequest" Value="40"/>
<Setter Property="FontSize" Value="18"/>
<Setter Property="Margin" Value="10, 0"/>
</Style>
</ResourceDictionary>
</ContentPage.Resources>
<Grid>
<Image Source="background.png" Aspect="AspectFill"/>
<Grid RowDefinitions="Auto, *, Auto">
<Border BackgroundColor="{x:StaticResource FirstColor}" StrokeThickness="2"
Stroke="White" Padding="16" Grid.Row="0">
<Border.StrokeShape>
<RoundRectangle CornerRadius="0, 0, 24, 24" />
</Border.StrokeShape>
<VerticalStackLayout Spacing="5">
<ScrollView Orientation="Vertical" HeightRequest="120"
VerticalScrollBarVisibility="Always">
<Grid ColumnDefinitions="0.8*, 0.2*" RowDefinitions="*, *"
ColumnSpacing="10">
<!--<Border Stroke="White" Padding="2" BackgroundColor="White"
Grid.Column="0">
<Border.StrokeShape>
<RoundRectangle CornerRadius="10"/>
</Border.StrokeShape>
<Entry PlaceholderColor="{x:StaticResource FirstColor}"
TextColor="{x:StaticResource SecondColor}"
FontSize="16" Placeholder="Введите комментарий"
BackgroundColor="White" ClearButtonVisibility="WhileEditing"
Text="{x:Binding TextFilter}" x:Name="FilterTextField"
MaxLength="50"
IsEnabled="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}}, Source={x:Reference CommentsPage}"/>
</Border>-->
<custom:ValidationEntryView ErrorText="Значение от 5 до 50
символов"
IsValidated="{x:Binding ValidationState[Text],
Source={x:Reference CommentsPage}"}

```



```

TextValue="{x:Binding Text, Source={x:Reference
CommentsPage}}}"
CanInput="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}, Source={x:Reference CommentsPage}}"
MinLength="5" DefaultText="Введите комментарий"
LabelText="Комментарий:"/>
<ImageButton Grid.Column="1" Source="addicon.png"
BorderColor="White" CornerRadius="10" BorderWidth="2" Grid.Row="0"
Padding="5" HeightRequest="{OnIdiom Desktop=40, Phone=50}"
Margin="0"
BackgroundColor="{x:StaticResource FirstColor}"
x:Name="AddCommentButton" Clicked="AddCommentButton_Clicked"
IsEnabled="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}, Source={x:Reference CommentsPage}}"/>
<ImageButton Grid.Column="1" Source="deleteicon.png"
BorderColor="White" CornerRadius="10" BorderWidth="2" Grid.Row="1"
Padding="8" HeightRequest="{OnIdiom Desktop=40, Phone=50}"
BackgroundColor="{x:StaticResource FirstColor}"
x:Name="DeleteCommentButton" Clicked="DeleteCommentButton_Clicked"
IsEnabled="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}, Source={x:Reference CommentsPage}}"/>
<Grid ColumnDefinitions="0.7*, 0.3*" ColumnSpacing="10"
Grid.Column="0" Grid.Row="1">
<Slider Minimum="0" Maximum="50" x:Name="UnitValueSlider"
MinimumTrackColor="White" ThumbColor="White" Margin="0, 10"
MaximumTrackColor="DarkSlateGrey" Grid.Column="0"
ValueChanged="UnitValueSlider_ValueChanged"
IsEnabled="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}, Source={x:Reference CommentsPage}}"/>
<Label Grid.Column="1" FontSize="14" TextColor="White"
HorizontalOptions="Center" VerticalOptions="Center">
<Label.FormattedText>
<FormattedString>
<Span Text="Оценка: "/>
<Span Text="{Binding Rating, Source={x:Reference
CommentsPage}, StringFormat='{0:F2}}'" />
</FormattedString>
</Label.FormattedText>
</Label>
</Grid>
</Grid>
</ScrollView>
</VerticalStackLayout>
</Border>
<Grid x:Name="CommentsListPanel" RowDefinitions="*" Grid.Row="1">
<VerticalStackLayout IsVisible="{x:Binding IsEmpty,
Converter={StaticResource InvertedBoolConverter}, Source={x:Reference CommentsPage}}"
Grid.Row="0">
<ScrollView Orientation="Vertical" BackgroundColor="Transparent"
x:Name="PageScroller"
IsEnabled="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}, Source={x:Reference CommentsPage}}">
<CollectionView x:Name="CommentsListView" ItemsSource="{Binding
Comments, Source={x:Reference CommentsPage}}"
HorizontalOptions="Fill" SelectionMode="Single">
<CollectionView.Resources>
<ResourceDictionary>
<toolkit:ByteArrayToImageSourceConverter
x:Key="ByteArrayToImageConverter" />
<Style x:Key="CardBorderStyle" TargetType="Border">
<Setter Property="StrokeThickness" Value="2"/>

```

```

        <Setter Property="BackgroundColor"
Value="{x:StaticResource Key=FirstColor}"/>
        <Setter Property="Margin" Value="24, 10"/>
        <Setter Property="Padding" Value="10"/>
        <Setter Property="StrokeShape">
            <Setter.Value>
                <RoundRectangle CornerRadius="16" />
            </Setter.Value>
        </Setter>
        <Setter Property="Stroke">
            <Setter.Value>
                <SolidColorBrush Color="White" />
            </Setter.Value>
        </Setter>
        <Setter Property="HorizontalOptions" Value="Fill"/>
        <Setter Property="HeightRequest" Value="120" />
    </Style>
</ResourceDictionary>
</CollectionView.Resources>
<CollectionView.ItemTemplate>
    <DataTemplate>
        <Grid>
            <VisualStateManager.VisualStateGroups>
                <VisualStateGroup Name="CommonStates">
                    <VisualState Name="Normal" />
                    <VisualState Name="Selected">
                        <VisualState.Setters>
                            <Setter Property="BackgroundColor"
Value="Transparent" />
                        </VisualState.Setters>
                    </VisualState>
                </VisualStateGroup>
            </VisualStateManager.VisualStateGroups>
            <Border Style="{StaticResource CardBorderStyle}">
                <Grid ColumnSpacing="10">
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="Auto" />
                        <ColumnDefinition Width="*" />
                        <ColumnDefinition Width="Auto"/>
                    </Grid.ColumnDefinitions>
                    <Border Grid.Column="0" HeightRequest="90"
WidthRequest="90" StrokeThickness="2">
                        <Border.StrokeShape>
                            <RoundRectangle CornerRadius="16" />
                        </Border.StrokeShape>
                        <Border.Stroke>
                            <SolidColorBrush Color="White" />
                        </Border.Stroke>
                        <Image BackgroundColor="White"
Aspect="AspectFill" HeightRequest="86" WidthRequest="86"
HorizontalOptions="Center"
VerticalOptions="Center"
Source="{Binding Profile.Image,
Converter={x:StaticResource ByteArrayToImageConverter}}"/>
                    </Border>
                    <StackLayout Grid.Column="1"
Orientation="Vertical" Spacing="5" HorizontalOptions="Fill">
                        <Label TextColor="White" FontSize="18"
TextDecorations="Underline"
LineBreakMode="TailTruncation">
                            <Label.FormattedText>
                                <FormattedString>
                                    <Span Text="{x:Binding
Profile.Name}" />
                                    <Span Text=" " />

```

```

Profile.Surname}" />
        <Span Text="{x:Binding
        </FormattedString>
        </Label.FormattedText>
    </Label>
    <Label Text="{Binding Text}"
    LineBreakMode="TailTruncation"/>
        <HorizontalStackLayout Spacing="10">
            <custom:RatingStarView
                NullIcon="starregular.png"
    ValueIcon="starsolid.png" MaxValue="5" Value="{x:Binding Rating}">
            </custom:RatingStarView>
            <Label FontSize="14" TextColor="White"
    LineBreakMode="TailTruncation">
                <Label.FormattedText>
                    <FormattedString>
                        <Span Text="Оценка: "/>
                        <Span Text="{Binding Rating,
    StringFormat='{0:F2}}'" />
                    </FormattedString>
                </Label.FormattedText>
            </Label>
        </HorizontalStackLayout>
        <Label FontSize="14" TextColor="White"
    LineBreakMode="TailTruncation">
            <Label.FormattedText>
                <FormattedString>
                    <Span Text="{Binding
    PublicationTime}" />
                </FormattedString>
            </Label.FormattedText>
        </Label>
    </StackLayout>
</Grid>
</Border>
</Grid>
</DataTemplate>
</CollectionView.ItemTemplate>
</CollectionView>
</ScrollView>
</VerticalStackLayout>
<Border BackgroundColor="{x:StaticResource FirstColor}" Stroke="White"
StrokeThickness="3" Padding="30"
    HeightRequest="200" WidthRequest="200" IsVisible="{x:Binding IsEmpty,
Source={x:Reference CommentsPage}}" Grid.Row="0"
    HorizontalOptions="Center" VerticalOptions="Center" Margin="0, 120">
    <Border.StrokeShape>
        <RoundRectangle CornerRadius="16" />
    </Border.StrokeShape>
    <VerticalStackLayout Spacing="10">
        <Image Source="nothing.png" HeightRequest="90" WidthRequest="90"/>
        <Label Text="Список нуст" FontSize="20" TextColor="White"
    BackgroundColor="Transparent" FontAttributes="Bold"
        VerticalOptions="Center" HorizontalOptions="Center"/>
    </VerticalStackLayout>
    </Border>
</Grid>
<Border BackgroundColor="{x:StaticResource FirstColor}" StrokeThickness="2"
Stroke="White" Grid.Row="2" Padding="5">
    <Border.StrokeShape>
        <RoundRectangle CornerRadius="24, 24, 0, 0" />
    </Border.StrokeShape>
    <Grid ColumnDefinitions="*, *" RowDefinitions="*, *" ColumnSpacing="10"
    Margin="15, 10" RowSpacing="10">

```

```

        <Label Grid.Row="0" HorizontalOptions="Center" FontSize="16"
Grid.ColumnSpan="2" TextDecorations="Underline"
        FontAttributes="Bold" TextColor="White">
            <Label.FormattedText>
                <FormattedString>
                    <Span Text="Страница: "/>
                    <Span Text="{Binding PageIndex, Source={x:Reference
CommentsPage}}"/>
                    <Span Text=" из "/>
                    <Span Text="{Binding PageCount, Source={x:Reference
CommentsPage}}"/>
                </FormattedString>
            </Label.FormattedText>
        </Label>
        <Button Grid.Row="1" Grid.Column="0" Text="Назад" x:Name="BackButton"
Clicked="BackButton_Clicked"
            Style="{x:StaticResource Key=PageButtonStyle}"/>
        <Button Grid.Row="1" Grid.Column="3" Text="Вперед" x:Name="NextButton"
Clicked="NextButton_Clicked"
            Style="{x:StaticResource Key=PageButtonStyle}"/>
    </Grid>
</Border>
</Grid>
    <custom:LoadingContentView IsLoading="{x:Binding IsLoading, Source={x:Reference
CommentsPage}}"/>
        CancelCommand="{x:Binding CancelCommand,
Source={x:Reference CommentsPage}}"/>
    </Grid>
</ContentPage>

using MauiLabs.View.Services.ApiModels.Commons.RecipeModels;
using MauiLabs.View.Services.ApiModels.RecipeModels.Comments.Requests;
using MauiLabs.View.Services.ApiModels.RecipeModels.Comments.Responses;
using MauiLabs.View.Services.Commons;
using MauiLabs.View.Services.Interfaces;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Net;
using System.Reflection;
using System.Windows.Input;

namespace MauiLabs.View.Pages.RecipePages;

public partial class CommentsListPage : ContentPage, INotifyPropertyChanged,
INavigationService.IQueryableNavigation
{
    protected internal CancellationTokenSource cancellationSource = new();
    protected internal readonly ICommentsList commentsList = default!;

    public static readonly string DefaultProfileImage =
$"MauiLabs.View.Resources.Images.Profile.defaultprofile.png";
    public static readonly int RecordsOnPage = 5, RatingScale = 10;
    public virtual double ImageSize { get => 96; }

    public ICommand GetCommentInfoCommand { get; protected set; } = default!;
    public ICommand GetCommentsListCommand { get; protected set; } = default!;

    public ICommand EditCommentInfoCommand { get; protected set; } = default!;
    public ICommand CancelCommand { get; protected set; } = default!;
    public CommentsListPage(ICommentsList commentsList) : base()
    {
        this.InitializeComponent();
        this.commentsList = commentsList;
        this.GetCommentInfoCommand = new Command(() =>
        {
            this.LaunchCancelableTask(() => this.GetCommentInfoCommandHandler());

```

```

    });
    this.GetCommentsListCommand = new Command(() =>
    {
        this.LaunchCancelableTask(() => this.GetCommentsListCommandHandler());
    });
    this.EditCommentInfoCommand = new Command(() =>
    {
        this.LaunchCancelableTask(() => this.EditCommentInfoCommandHandler());
    });
    this.CancelCommand = new Command(this.CancelCommandHandler);
}
private protected bool isLoading = default;
public bool IsLoading { get => this.isLoading; set { this.isLoading = value;
OnPropertyChanged(); } }

private protected double rating = default!;
public double Rating { get => this.rating; set { this.rating = value;
OnPropertyChanged(); } }

private protected string text = string.Empty;
public string Text { get => this.text; set { this.text = value;
OnPropertyChanged(); } }

public ObservableCollection<CommentInfoModel> Comments { get; protected set; } =
new();

private protected int pageIndex = default!;
public int PageIndex { get => this.pageIndex; set { this.pageIndex = value;
OnPropertyChanged(); } }

private protected int pageCount = default!;
public int PageCount { get => this.pageCount; set { this.pageCount = value;
OnPropertyChanged(); } }

private protected int allCount = default!;
public int AllCount { get => this.allCount; set { this.IsEmpty = (this.allCount =
value) <= 0; OnPropertyChanged(); } }

private protected bool isEmpty = default!;
public bool IsEmpty { get => this.isEmpty; set { this.isEmpty = value;
OnPropertyChanged(); } }
public Dictionary<string, bool> ValidationState { get; set; } = new();

public virtual bool CommentsLoaded { get; protected set; } = default;
public virtual int RecipeId { get; protected set; } = default!;

protected async void LaunchCancelableTask(Func<Task> cancelableTask) => await
Task.Run(async () =>
{
    this.IsLoading = true; await cancelableTask.Invoke();
    this.IsLoading = false;
});
protected virtual async void CancelCommandHandler(object sender) => await Task.Run(()
=>
{
    if (this.isLoading == false) return;

    this.cancellationSource.Cancel();
    this.cancellationSource = new CancellationTokensource();

    (this.IsLoading) = (default);
});
public virtual byte[] FileToByteArray(string filename)
{

```

```

        using (var fileStream =
Assembly.GetExecutingAssembly().GetManifestResourceStream(filename))
        {
            using var binaryReader = new BinaryReader(fileStream);
            return binaryReader.ReadBytes((int)fileStream.Length);
        }
    }
    protected virtual void DisplayMessage(string title, string message)
    {
        this.Dispatcher.Dispatch(async () => await this.DisplayAlert(title, message,
"Назад"));
    }
    protected virtual async Task GetCommentInfoCommandHandler() => await
userManager.SendRequest(async (token) =>
    {
        var requestModel = new RequestInfo<GetCommentRequestModel>()
        {
            RequestModel = new GetCommentRequestModel() { RecipeId = this.RecipeId },
            CancelToken = this.cancellationSource.Token, ProfileToken = token,
        };
        GetCommentResponseModel commentResult = default!;
        try { commentResult = await this.commentsList.GetCommentInfo(requestModel); }
        catch (ViewServiceException errorInfo) when (errorInfo.ExceptionType !=
HttpStatusCode.Unauthorized)
        {
            this.CommentsLoaded = default!; return;
        }
        this.CommentsLoaded = true;
        await this.Dispatcher.DispatchAsync(() =>
        {
            (this.Text, this.Rating) = (commentResult.Text, commentResult.Rating);
            this.UnitValueSlider.Value = commentResult.Rating * RatingScale;
        });
    }, async (errorInfo) => await Shell.Current.Navigation.PopAsync(animated: true));
    protected virtual async Task GetCommentsListCommandHandler() => await
userManager.SendRequest(async (token) =>
    {
        var commentsResult = await this.commentsList.GetCommentsList(new
RequestInfo<GetRecipeCommentsRequestModel>()
        {
            RequestModel = new GetRecipeCommentsRequestModel()
            {
                RecipeId = this.RecipeId, SortingType = CommentSortingType.ByDate,
                Skip = RecordsOnPage * (this.PageIndex - 1),
                Take = (RecordsOnPage * (this.PageIndex - 1)) + RecordsOnPage,
            },
            CancelToken = this.cancellationSource.Token, ProfileToken = token,
        });
        await this.Dispatcher.DispatchAsync(() =>
        {
            foreach (var commentsRecord in commentsResult.Comments)
            {
                commentsRecord.Profile.Image = commentsRecord.Profile.Image.Length != 0
                ? commentsRecord.Profile.Image :
this.FileToByteArray(DefaultProfileImage);
            }
            (this.AllCount, this.Comments) = (commentsResult.AllCount,
new(commentsResult.Comments));
            this.PageCount = (int)Math.Ceiling(commentsResult.AllCount /
(double)RecordsOnPage);

            this.CommentsListView.ItemsSource = this.Comments;
        });
    }, async (errorInfo) => await Shell.Current.Navigation.PopAsync(animated: true));

```

```

        protected virtual async Task EditCommentInfoCommandHandler() => await
        UserManager.SendRequest(async (token) =>
        {
            if (this.CommentsLoaded) await this.commentsList.EditComment(new
        RequestInfo<EditCommentRequestModel>()
            {
                RequestModel = new EditCommentRequestModel() { RecipeId = this.RecipeId,
        Rating = this.Rating, Text = this.Text, },
                CancelToken = this.cancellationSource.Token, ProfileToken = token,
            });
            else await this.commentsList.AddComment(new RequestInfo<AddCommentRequestModel>()
            {
                RequestModel = new AddCommentRequestModel() { RecipeId = this.RecipeId, Rating
        = this.Rating, Text = this.Text, },
                CancelToken = this.cancellationSource.Token, ProfileToken = token,
            });
            this.DisplayMessage("Успешное действие", "Комментарий успешно сохранен");
            (this.PageIndex, this.CommentsLoaded) = (1, true);
            this.GetCommentsListCommand.Execute(null);
        }, async (errorInfo) => await Shell.Current.Navigation.PopAsync(animated: true));
        protected virtual void BackButton_Clicked(object sender, EventArgs args)
        {
            if (this.PageIndex <= 1) return;
            this.PageIndex--;
            this.GetCommentsListCommand.Execute(false);
        }
        protected virtual void NextButton_Clicked(object sender, EventArgs args)
        {
            if (this.PageIndex >= this.PageCount) return;
            this.PageIndex++;
            this.GetCommentsListCommand.Execute(false);
        }
        protected virtual void UnitValueSlider_ValueChanged(object sender,
        ValueChangedEventArgs args)
        {
            this.Rating = ((int)this.UnitValueSlider.Value / RatingScale);
        }
        protected virtual void AddCommentButton_Clicked(object sender, EventArgs args)
        {
            if (this.ValidationState["Text"]) this.EditCommentInfoCommand.Execute(null);
            else this.DisplayMessage("Произошла ошибка", "Неверное заполнено поле
        комментария");
        }
        protected virtual void DeleteCommentButton_Clicked(object sender, EventArgs args)
        {
            this.LaunchCancelableTask(async () => await UserManager.SendRequest(async (token)
        =>
            {
                if (!this.CommentsLoaded) return;
                await this.commentsList.DeleteComment(new
        RequestInfo<DeleteCommentRequestModel>()
                {
                    RequestModel = new DeleteCommentRequestModel() { RecipeId = this.RecipeId },
                    CancelToken = this.cancellationSource.Token,
                    ProfileToken = token,
                });
                await this.Dispatcher.DispatchAsync(() =>
                {
                    (this.Text, this.UnitValueSlider.Value, this.PageIndex) = (string.Empty,
        default!, 1);
                });
                this.CommentsLoaded = default!;
                this.DisplayMessage("Успешное действие", "Комментарий успешно удален");
                this.GetCommentsListCommand.Execute(null);
            },
            },

```



```

        async (errorInfo) => await Shell.Current.Navigation.PopAsync(animated: true)));
    }
    protected override void OnAppearing() => this.Dispatcher.Dispatch(() =>
    {
        this.GetCommentInfoCommand.Execute(null);
        this.PageIndex = 1;
        this.GetCommentsListCommand.Execute(null);
    });
    public virtual void SetNavigationQuery(IDictionary<string, object> queries) =>
    this.RecipeId = (int)queries["RecipeId"];
    protected override void OnDisappearing() => base.OnDisappearing();
    }

```

Файл «RecipeInfoPage.xaml»

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    x:Class="MauiLabs.View.Pages.RecipePages.RecipeInfoPage"
    xmlns:custom="clr-namespace:MauiLabs.View.Commons.ContentViews"
    xmlns:system="clr-namespace:System;assembly=netstandard"
    xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"
    Title="Просмотр рецепта" x:Name="RecipePage"
    Shell.BackgroundColor="{x:StaticResource Key=FirstColor}"
    Shell.FlyoutBehavior="Disabled">
    <ContentPage.Resources>
        <ResourceDictionary>
            <toolkit:InvertedBoolConverter x:Key="InvertedBoolConverter" />
            <toolkit:ByteArrayToImageSourceConverter x:Key="ByteArrayToImageConverter" />
            <Style x:Key="ButtonStyle" TargetType="Button">
                <Setter Property="CornerRadius" Value="16" />
                <Setter Property="BackgroundColor" Value="{x:StaticResource
Key=FirstColor}" />
                <Setter Property="BorderColor" Value="White" />
                <Setter Property="FontSize" Value="18"/>
                <Setter Property="TextColor" Value="White"/>
                <Setter Property="BorderWidth" Value="2" />
            </Style>
            <Style x:Key="TextFieldStyle" TargetType="Entry">
                <Setter Property="PlaceholderColor" Value="{x:StaticResource
Key=FirstColor}"/>
                <Setter Property="BackgroundColor" Value="White" />
                <Setter Property="FontSize" Value="18" />
                <Setter Property="TextColor" Value="{x:StaticResource Key=SecondColor}" />
            </Style>
            <Style x:Key="TextFieldLabelStyle" TargetType="Label">
                <Setter Property="BackgroundColor" Value="Transparent" />
                <Setter Property="TextColor" Value="White"/>
                <Setter Property="FontSize" Value="14" />
            </Style>
            <Style x:Key="ImagePickerStyle" TargetType="ImageButton">
                <Setter Property="CornerRadius" Value="10" />
                <Setter Property="BorderColor" Value="White"/>
                <Setter Property="BorderWidth" Value="2"/>
                <Setter Property="BackgroundColor" Value="{x:StaticResource FirstColor}"/>
            </Style>
            <Style x:Key="CardBorderStyle" TargetType="Border">
                <Setter Property="StrokeThickness" Value="2"/>
                <Setter Property="BackgroundColor" Value="{x:StaticResource
Key=FirstColor}"/>
                <!--<Setter Property="Margin" Value="20, 10, 30, 20"/>-->
                <Setter Property="Padding" Value="10"/>
                <Setter Property="StrokeShape">
                    <Setter.Value>

```



```

        <RoundRectangle CornerRadius="16" />
    </Setter.Value>
</Setter>
<Setter Property="Stroke">
    <Setter.Value>
        <SolidColorBrush Color="White" />
    </Setter.Value>
</Setter>
<Setter Property="HorizontalOptions" Value="Fill"/>
<!--<Setter Property="HeightRequest" Value="160" />-->
</Style>
</ResourceDictionary>
</ContentPage.Resources>
<Grid>
    <Image Source="background.png" Aspect="AspectFill"/>
    <ScrollView Orientation="Vertical" BackgroundColor="Transparent"
x:Name="PageScroller"
        IsEnabled="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}, Source={x:Reference RecipePage}}">
        <StackLayout x:Name="RecipePanel" Orientation="Vertical"
HorizontalOptions="Fill" Margin="30, 10, 30, 40"
            Opacity="0.0" Scale="1.5">

            <Border StrokeThickness="2" MaximumWidthRequest="260" Padding="20"
10"
                BackgroundColor="{x:StaticResource Key=FirstColor}" Margin="0, 20, 0,
10"
                FlexLayout.AlignSelf="Stretch" Stroke="White">
                <Border.StrokeShape>
                    <RoundRectangle CornerRadius="20" />
                </Border.StrokeShape>
                <VerticalStackLayout Spacing="5">
                    <Label Text="Изображение рецепта:" TextColor="White" FontSize="18"
Margin="4, 0, 0, 5" TextDecorations="Underline"/>
                    <Grid RowDefinitions="*">
                        <Grid.ColumnDefinitions>
                            <ColumnDefinition Width="0.6*" />
                            <ColumnDefinition Width="0.4*" />
                        </Grid.ColumnDefinitions>
                        <Border x:Name="RecipeImageContent" HeightRequest="112"
WidthRequest="112" Stroke="White"
                            StrokeThickness="2" BackgroundColor="White"
Grid.Column="0" Padding="0">
                            <Border.StrokeShape>
                                <RoundRectangle CornerRadius="20" />
                            </Border.StrokeShape>
                            <Image Aspect="AspectFill" HeightRequest="112"
WidthRequest="112"/>
                        </Border>
                        <Grid RowDefinitions="*, *" ColumnDefinitions="*" Margin="5, 5"
Grid.Column="1">
                            <ImageButton Grid.Row="0" Grid.Column="0"
x:Name="BookmarkButton"
                                Clicked="BookmarkButton_Clicked" Style="{x:StaticResource
ImagePickerStyle}"
                                Source="bookmark.png" Margin="4" Padding="3"
WidthRequest="60" HeightRequest="46"
                                IsEnabled="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}, Source={x:Reference RecipePage}}"/>
                            <ImageButton Grid.Row="1" Grid.Column="0"
x:Name="RecommendButton"
                                Clicked="RecommendButton_Clicked" Style="{x:StaticResource
ImagePickerStyle}"
                                Source="recommend.png" WidthRequest="60"
HeightRequest="46" Margin="4" Padding="4"

```

```

        IsEnabled="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}}, Source={x:Reference RecipePage}}"/>
    </Grid>
</Grid>
</VerticalStackLayout>
</Border>
<Border StrokeThickness="2" MaximumWidthRequest="400" Padding="20, 20, 20, 40"
BackgroundColor="{x:StaticResource Key=FirstColor}"
    Margin="0, 10, 0, 16" FlexLayout.AlignSelf="Stretch" Stroke="White">
    <Border.StrokeShape>
        <RoundRectangle CornerRadius="20" />
    </Border.StrokeShape>
    <VerticalStackLayout Spacing="10">
        <Label Text="Данные рецепта:" TextColor="White" FontSize="18" Margin="0, 0, 0, 5"
TextDecorations="Underline"/>

        <VerticalStackLayout Spacing="5" BindingContext="{x:Reference this}">
            <Label Text="Название рецепта:" Style="{x:StaticResource TextFieldLabelStyle}"
/>

            <Border BackgroundColor="White" Padding="2">
                <Border.StrokeShape>
                    <RoundRectangle CornerRadius="10" />
                </Border.StrokeShape>
                <Entry Placeholder="Укажите название" Text="{x:Binding RecipeName,
Source={x:Reference RecipePage}}"
                    IsEnabled="False" FontSize="16" Style="{x:StaticResource
Key=TextFieldStyle}">
                </Entry>
            </Border>
        </VerticalStackLayout>

        <VerticalStackLayout Spacing="5" BindingContext="{x:Reference this}">
            <Label Text="Категория рецепта:" Style="{x:StaticResource
TextFieldLabelStyle}" />
            <Border BackgroundColor="White" Padding="2">
                <Border.StrokeShape>
                    <RoundRectangle CornerRadius="10" />
                </Border.StrokeShape>
                <Entry Placeholder="Укажите категорию" Text="{x:Binding RecipeCategory,
Source={x:Reference RecipePage}}"
                    IsEnabled="False" FontSize="16" Style="{x:StaticResource
Key=TextFieldStyle}">
                </Entry>
            </Border>
        </VerticalStackLayout>
        <VerticalStackLayout Spacing="5">
            <Label Text="Описание рецепта:" Style="{x:StaticResource TextFieldLabelStyle}"
/>

            <Border BackgroundColor="{x:StaticResource FirstColor}" Padding="5, 2"
Stroke="White" StrokeThickness="2">
                <Border.StrokeShape>
                    <RoundRectangle CornerRadius="10" />
                </Border.StrokeShape>
                <Editor BackgroundColor="Transparent" MaxLength="160"
                    HeightRequest="100" FontSize="16" Placeholder="Укажите описание рецепта"
                    PlaceholderColor="White" IsSpellCheckEnabled="False" TextColor="White"
                    Text="{x:Binding Description, Source={x:Reference RecipePage}}"
                    IsEnabled="False"/>
            </Border>
        </VerticalStackLayout>

        <custom:ExpanderView ButtonIcon="arrowicon.png" ExpanderHeight="200"
ButtonText="Посмотреть ингредиенты"
            Margin="0, 5, 0, 5" ButtonHeight="52" WidthRequest="{OnIdiom Phone=280,
Desktop=340}">

```

```

<custom:ExpanderView.Item>
    <Border BackgroundColor="Transparent" Margin="0, 46, 0, 0">
        <Grid RowDefinitions="Auto, Auto, *" >
            <Grid Grid.Row="0" ColumnDefinitions="0.7*, 0.3*" RowDefinitions="*"
Margin="20, 10, 20, 0" ColumnSpacing="5">
                <Border Grid.Column="0" StrokeThickness="0"
BackgroundColor="Transparent">
                    <Label Text="Название:" TextColor="White" FontSize="14"
TextDecorations="Underline" FontAttributes="Bold"/>
                </Border>
                <Border Grid.Column="1" StrokeThickness="0"
BackgroundColor="Transparent">
                    <Label Text="Кол-во:" TextColor="White" FontSize="14"
TextDecorations="Underline" FontAttributes="Bold"/>
                </Border>
            </Grid>
            <ScrollView Orientation="Vertical" BackgroundColor="Transparent"
Grid.Row="2"
                <IsVisible="{x:Binding IngredientsEmpty, Source={x:Reference
RecipePage}, Converter={StaticResource InvertedBoolConverter}}" >
                    <CollectionView x:Name="IngredientsView" SelectionMode="Single">
                        <CollectionView.Resources>
                            <ResourceDictionary>
                                <Style x:Key="CardBorderStyle" TargetType="Border">
                                    <Setter Property="StrokeThickness" Value="2"/>
                                    <Setter Property="BackgroundColor"
Value="{x:StaticResource Key=FirstColor}"/>
                                    <Setter Property="Margin" Value="10, 4"/>
                                    <Setter Property="Padding" Value="10"/>
                                    <Setter Property="Stroke" Value="White"/>
                                    <Setter Property="StrokeShape">
                                        <Setter.Value>
                                            <RoundRectangle CornerRadius="10"/>
                                        </Setter.Value>
                                    </Setter>
                                    <Setter Property="HorizontalOptions" Value="Fill"/>
                                </Style>
                            </ResourceDictionary>
                        </CollectionView.Resources>
                        <CollectionView.ItemTemplate>
                            <DataTemplate>
                                <Grid HorizontalOptions="Fill">
                                    <VisualStateManager.VisualStateGroups>
                                        <VisualStateGroup Name="CommonStates">
                                            <VisualState Name="Normal" />
                                            <VisualState Name="Selected">
                                                <VisualState.Setters>
                                                    <Setter Property="BackgroundColor"
Value="Transparent" />
                                                </VisualState.Setters>
                                            </VisualState>
                                        </VisualStateGroup>
                                    </VisualStateManager.VisualStateGroups>
                                <Border Style="{StaticResource CardBorderStyle}">
                                    <Grid ColumnDefinitions="0.7*, 0.3*"
ColumnSpacing="5">
                                        <Label Text="{x:Binding Name}" FontSize="14"
TextColor="White" LineBreakMode="TailTruncation"/>
                                        <Label Grid.Column="1" FontSize="14"
TextColor="White" LineBreakMode="TailTruncation">
                                            <Label.FormattedText>
                                                <FormattedString>
                                                    <Span Text="{x:Binding Value}"/>

```

```

        <Span Text=" [/>
        <Span Text="{x:Binding Unit}"/>
        <Span Text="]"/>
    </FormattedString>
    </Label.FormattedText>
</Label>
</Grid>
</Border>
</Grid>
</DataTemplate>
</CollectionView.ItemTemplate>
</CollectionView>
</ScrollView>
<Label Grid.Row="2" Text="Нет ингредиентов" TextColor="White"
FontSize="18" TextDecorations="Underline"
HorizontalOptions="Center" VerticalOptions="Center"
IsVisible="{x:Binding IngredientsEmpty, Source={x:Reference
RecipePage}}"/>
    </Grid>

</Border>
</custom:ExpanderView.Item>
</custom:ExpanderView>
</VerticalStackLayout>
</Border>

<Border Style="{StaticResource CardBorderStyle}" Margin="0, 0, 0, 20">
    <Grid ColumnSpacing="10" RowDefinitions="Auto, *">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
        <Label Text="Данные автора:" TextColor="White" FontSize="18"
Margin="0, 0, 0, 2" TextDecorations="Underline"
Grid.Row="0"/>
        <Border x:Name="PublisherImageContent" Grid.Row="1"
Stroke="White" Grid.Column="0" HeightRequest="100"
WidthRequest="100" StrokeThickness="2">
            <Border.StrokeShape>
                <RoundRectangle CornerRadius="16" />
            </Border.StrokeShape>
            <Image BackgroundColor="White" Aspect="AspectFill"
HeightRequest="90" WidthRequest="90"
HorizontalOptions="Center" VerticalOptions="Center"
Source="{Binding PublisherImage, Converter={x:StaticResource
ByteArrayToImageConverter}, Source={x:Reference RecipePage}}"/>
        </Border>
        <StackLayout Grid.Column="1" Orientation="Vertical" Spacing="5"
HorizontalOptions="Fill" Grid.Row="1">
            <Label LineBreakMode="TailTruncation" FontSize="16" Margin="0, 5,
0, 0" TextColor="White" FontAttributes="Bold">
                <Label.FormattedText>
                    <FormattedString>
                        <Span Text="{Binding PublisherName,
Source={x:Reference RecipePage}}"/>
                        <Span Text=" "/>
                        <Span Text="{Binding PublisherSurname,
Source={x:Reference RecipePage}}"/>
                    </FormattedString>
                </Label.FormattedText>
            </Label>
            <Label LineBreakMode="TailTruncation" FontSize="16" Margin="0, 5,
0, 0" TextColor="White" TextDecorations="Underline">
                <Label.FormattedText>
                    <FormattedString>

```

```

        <Span Text="{Binding PublisherTime,
Source={x:Reference RecipePage}}"/>
        </FormattedString>
    </Label.FormattedText>
</Label>

    <Button Grid.Column="2" x:Name="PublisherRecipesButton"
Text="Публикации" TextColor="White" FontSize="16"
        BackgroundColor="{x:StaticResource FirstColor}" Margin="0, 10,
10, 10"
        BorderWidth="2" BorderColor="White" CornerRadius="10"
Clicked="PublisherRecipesButton_Clicked"
        IsEnabled="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}, Source={x:Reference RecipePage}}"/>
    </StackLayout>
</Grid>
</Border>
<Button x:Name="CommentsButton" Clicked="CommentsButton_Clicked"
Text="Просмотр комментариев"
        FontSize="18" CornerRadius="12" BorderColor="White"
TextColor="White" BorderWidth="2"
        BackgroundColor="{x:StaticResource FirstColor}" HeightRequest="56"
        IsEnabled="{Binding IsLoading, Converter={StaticResource
InvertedBoolConverter}, Source={x:Reference RecipePage}}"/>
    </StackLayout>
</ScrollView>
<custom:LoadingContentView IsLoading="{x:Binding IsLoading, Source={x:Reference
RecipePage}}">
        CancelCommand="{x:Binding CancelCommand,
Source={x:Reference RecipePage}}"/>
</Grid>
</ContentPage>

```

```

using MauiLabs.View.Services.ApiModels.ProfileModels.Bookmarks.Requests;
using MauiLabs.View.Services.ApiModels.RecipeModels.CookingRecipe.Requests;
using MauiLabs.View.Services.Commons;
using MauiLabs.View.Services.Interfaces;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.Reflection;
using System.Windows.Input;

namespace MauiLabs.View.Pages.RecipePages;

public partial class RecipeInfoPage : ContentPage, INotifyPropertyChanged,
INavigationService.IQueryableNavigation
{
    protected internal CancellationTokenSource cancellationSource = new();
    protected internal readonly ICookingRecipes cookingRecipes = default!;
    protected internal readonly IBookmarksList bookmarks = default!;
    protected internal readonly INavigationService navigationService = default!;

    public static readonly string DefaultRecipeImage =
    $"MauiLabs.View.Resources.Images.Recipe.defaultrecipe.jpg";
    public static readonly string DefaultProfileImage =
    $"MauiLabs.View.Resources.Images.Profile.defaultprofile.png";
    public virtual double ImageSize { get => 96; }

    public ICommand GetRecipeCommand { get; protected set; } = default!;
    public ICommand CancelCommand { get; protected set; } = default!;

    public RecipeInfoPage(ICookingRecipes cookingRecipes, IBookmarksList bookmarks,
INavigationService navigationService) : base()
    {
        this.InitializeComponent();
    }
}

```

```

        (this.cookingRecipes, this.navigationService, this.bookmarks) = (cookingRecipes,
navigationService, bookmarks);
        this.GetRecipeCommand = new Command(() =>
        {
            this.LaunchCancelableTask(() => this.GetRecipeCommandHandler());
        });
        this.CancelCommand = new Command(this.CancelCommandHandler);
    }
    private protected bool isLoading = default;
    public bool IsLoading { get => this.isLoading; set { this.isLoading = value;
OnPropertyChanged(); } }

    private protected string recipeName = default!;
    public string RecipeName { get => this.recipeName; set { this.recipeName = value;
OnPropertyChanged(); } }

    private protected string recipeCategory = default!;
    public string RecipeCategory { get => this.recipeCategory; set { this.recipeCategory
= value; OnPropertyChanged(); } }

    private protected string description = default!;
    public string Description { get => this.description; set { this.description = value;
OnPropertyChanged(); } }

    private protected byte[] recipeImage = new byte[0];
    public byte[] RecipeImage { get => this.recipeImage; set { this.recipeImage = value;
OnPropertyChanged(); } }

    private protected byte[] publisherImage = new byte[0];
    public byte[] PublisherImage { get => this.publisherImage; set { this.publisherImage
= value; OnPropertyChanged(); } }

    private protected string publisherName = default!;
    public string PublisherName { get => this.publisherName; set { this.publisherName =
value; OnPropertyChanged(); } }

    private protected string publisherSurname = default!;
    public string PublisherSurname { get => this.publisherSurname; set
{ this.publisherSurname = value; OnPropertyChanged(); } }

    private protected string publisherTime = default!;
    public string PublisherTime { get => this.publisherTime; set { this.publisherTime =
value; OnPropertyChanged(); } }

    public virtual int PublisherId { get; protected set; } = default!;

    public sealed class IngredientModel : object
    {
        public required string Name { get; set; } = default!;
        public required string Unit { get; set; } = default!;
        public required string Value { get; set; } = default!;
    }
    private protected bool ingredientsEmpty = true;
    public bool IngredientsEmpty { get => this.ingredientsEmpty; set
{ this.ingredientsEmpty = value; OnPropertyChanged(); } }
    public virtual ObservableCollection<IngredientModel> Ingredients { get; protected
set; } = new();
    public virtual int RecipeId { get; protected set; } = default!;
    public virtual bool EnablePublisher { get; protected set; } = default!;

    protected async void LaunchCancelableTask(Func<Task> cancelableTask) => await
Task.Run(async () =>
    {
        this.IsLoading = true; await cancelableTask.Invoke();
        this.IsLoading = false;
    }

```

```

});
protected virtual async void CancelCommandHandler(object sender) => await Task.Run(()
=>
{
    if (this.isLoading == false) return;

    this.cancellationSource.Cancel();
    this.cancellationSource = new CancellationTokensource();

    (this.IsLoading) = (default);
});
public virtual byte[] FileToByteArray(string filename)
{
    using (var fileStream =
Assembly.GetExecutingAssembly().GetManifestResourceStream(filename))
    {
        using var binaryReader = new BinaryReader(fileStream);
        return binaryReader.ReadBytes((int)fileStream.Length);
    }
}
protected virtual void DisplayMessage(string title, string message)
{
    this.Dispatcher.Dispatch(async () => await this.DisplayAlert(title, message,
"Назад"));
}
protected virtual void ReloadProfileImage(byte[] image, Border border) =>
this.Dispatcher.Dispatch(() =>
{
    border.Content = new Image()
    {
        Source = ImageSource.FromStream(() => new MemoryStream(image)),
        Margin = Thickness.Zero, Aspect = Aspect.AspectFill,
    };
});
protected virtual void RecommendButton_Clicked(object sender, EventArgs args)
{
}
protected virtual void BookmarkButton_Clicked(object sender, EventArgs args)
{
    this.LaunchCancelableTask(() => UserManager.SendRequest(async token =>
    {
        await this.bookmarks.AddBookmark(new RequestInfo<AddBookmarkRequestModel>()
        {
            RequestModel = new AddBookmarkRequestModel() { RecipeId = this.RecipeId },
            CancelToken = this.cancellationSource.Token, ProfileToken = token,
        });
        this.DisplayMessage("Успешное действие", "Рецепт сохранен в заметках");
    }));
}
protected virtual async void PublisherRecipesButton_Clicked(object sender, EventArgs
args)
{
    if (this.EnablePublisher)
    {
        var queryParam = new Dictionary<string, object>() { ["PublisherId"] =
this.PublisherId };
        await this.navigationService.NavigateToPage<PublisherInfoPage>(Shell.Current,
queryParam);
    }
}
protected virtual async void CommentsButton_Clicked(object sender, EventArgs args)
{
    var queryParam = new Dictionary<string, object>() { ["RecipeId"] = this.RecipeId };

```



```

        await this.navigationService.NavigateToPage<CommentsListPage>(Shell.Current,
queryParam);
    }
    protected virtual async Task GetRecipeCommandHandler() => await
userManager.SendRequest(async (token) =>
    {
        var recipeResult = await this.cookingRecipes.GetRecipeInfo(new
RequestInfo<GetRecipeRequestModel>()
        {
            RequestModel = new GetRecipeRequestModel() { RecipeId = this.RecipeId },
            CancelToken = this.cancellationSource.Token, ProfileToken = token,
        });
        await this.Dispatcher.DispatchAsync(() =>
        {
            if (recipeResult.Image.Length <= 0)
            {
                this.ReloadProfileImage(this.FileToByteArray(DefaultRecipeImage),
this.RecipeImageContent);
                this.RecipeImage = null;
            }
            else this.ReloadProfileImage(this.RecipeImage = recipeResult.Image,
this.RecipeImageContent);
            (this.RecipeName, this.Description) = (recipeResult.Name,
recipeResult.Description);
            this.RecipeCategory = recipeResult.Category;

            if (recipeResult.Publisher.Image.Length <= 0)
            {
                this.ReloadProfileImage(this.FileToByteArray(DefaultProfileImage),
this.PublisherImageContent);
                this.RecipeImage = null;
            }
            else this.ReloadProfileImage(this.RecipeImage = recipeResult.Publisher.Image,
this.PublisherImageContent);
            (this.PublisherName, this.PublisherSurname) = (recipeResult.Publisher.Name,
recipeResult.Publisher.Surname);
            this.PublisherTime = recipeResult.PublicationTime.ToString();

            this.PublisherId = recipeResult.PublisherId;

            if (recipeResult.Ingredients.Count > 0)
            {
                this.Ingredients = new(recipeResult.Ingredients.Select(item => new
IngredientModel()
                {
                    Name = item.Name, Unit = item.Unit, Value = item.Value.ToString()
                }));
                (this.IngredientsView.ItemsSource, this.IngredientsEmpty) =
(this.Ingredients, default!);
            }
        });
    }, async (errorInfo) => await Shell.Current.Navigation.PopAsync(animated: true));
    protected override void OnAppearing() => this.Dispatcher.Dispatch(async () =>
    {
        this.ReloadProfileImage(this.FileToByteArray(DefaultRecipeImage),
this.RecipeImageContent);
        this.ReloadProfileImage(this.FileToByteArray(DefaultProfileImage),
this.PublisherImageContent);
        this.GetRecipeCommand.Execute(null);
        await Task.WhenAll(new Task[]
        {
            this.RecipePanel.ScaleTo(1.0, length: 600, easing: Easing.SinInOut),
            this.RecipePanel.FadeTo(1.0, length: 600, easing: Easing.SinInOut),
        });
        (this.RecipePanel.Opacity, this.RecipePanel.Scale) = (1.0, 1.0)
    });
}

```