Пермский филиал федерального государственного автономного

образовательного учреждения высшего образования

«Национальный исследовательский университет

«Высшая школа экономики»

Факультет экономики, менеджмента и бизнес-информатики

Цидвинцев Захар Викторович

**Разработка приложений для автоматического подбора рецептов**

*Курсовой проект*

студента образовательной программы бакалавриата

«Программная инженерия»

направление подготовки *09.03.04 Программная инженерия*

Старший преподаватель каф. ИТБ

Красилич Н.В.

Пермь, 2022 год

# Аннотация

Данная работа описывает процесс разработки приложений для управления рецептами. Работа состоит из 54 страниц формата A4, включает в себя пять глав.

В основной части содержится 6 таблиц и 28 рисунков, которые иллюстрируют прототипы интерфейсов и примеры кода.

В работе содержится три приложения: ссылки на код приложений, техническое задание и руководство пользователя.

# Оглавление

[Аннотация 2](#_Toc99272282)

[Оглавление 3](#_Toc99272283)

[Введение 5](#_Toc99272284)

[Глава 1 Анализ предметной области 7](#_Toc99272285)

[1.1 Назначение системы и основные характеристики 7](#_Toc99272286)

[1.2 Определение бизнес-процессов и их операций 8](#_Toc99272287)

[1.3 Описание программных требований 12](#_Toc99272288)

[1.4 Описание функциональных требований 12](#_Toc99272289)

[1.5 Результаты 13](#_Toc99272290)

[Глава 2 Технологии и средства реализации приложения 14](#_Toc99272291)

[2.1 Выбор СУБД для Базы Данных 14](#_Toc99272292)

[2.2 Выбор программных средств реализации 14](#_Toc99272293)

[2.3 Выбор шаблона проектирования 15](#_Toc99272294)

[Глава 3 Проектирование Базы данных 17](#_Toc99272295)

[Глава 4 Проектирование интерфейсов приложения 19](#_Toc99272296)

[4.1 Выбор цветовой схемы оформления 19](#_Toc99272297)

[4.2 Определение общих правил оформления 19](#_Toc99272298)

[4.3 Распределение функций по экранам приложения для Администратора 20](#_Toc99272299)

[4.4 Создание прототипа интерфейса приложения для Администратора 24](#_Toc99272300)

[4.5 Распределение функций по экранам приложения для Посетителя 31](#_Toc99272301)

[4.6 Создание прототипа интерфейса приложения для Посетителя 33](#_Toc99272302)

[4.7 Результаты 35](#_Toc99272303)

[Глава 5 Реализация приложения 36](#_Toc99272304)

[5.1 Реализация Базы данных 36](#_Toc99272305)

[5.2 Реализация шаблона MVVM 36](#_Toc99272306)

[5.2.1 Создание классов моделей 36](#_Toc99272307)

[5.2.2 Создание классов представлений моделей 38](#_Toc99272308)

[5.2.3 Создание представлений 39](#_Toc99272309)

[5.2.4 Привязка данных 40](#_Toc99272310)

[5.2.5 Создание команд 41](#_Toc99272311)

[5.2.6 Создание поведений 43](#_Toc99272312)

[5.3 Реализация интерфейса 44](#_Toc99272313)

[5.4 Диалоговый сервис 45](#_Toc99272314)

[5.4.1 Dialog Service 45](#_Toc99272315)

[5.4.2 Базовая ViewModel 45](#_Toc99272316)

[5.4.2 DialogYesNoViewModel 46](#_Toc99272317)

[5.5 Реализация запросов к базе данных 47](#_Toc99272318)

[5.6 Результаты реализации 48](#_Toc99272319)

[Заключение 49](#_Toc99272320)

[Приложение А. Репозитории GitHub 50](#_Toc99272321)

# Введение

В настоящее время информационные технологии широко используются в разных сферах деятельности человека: производство, образование, банковское дело, коммерция и другие. Программные инструменты имеют большую важность и значимость при решении прикладных и бытовых задач современного человека. На смену старым “аналоговым” устройствам приходят цифровые решения. Вокруг современного человека присутствует множество примеров такой замены, которую некоторые могут даже не замечать. Телефонные книги, карты города, новостные издания, календари и записные книги – все эти и многие другие привычные вещи из прошлого в настоящее время заменены на приложения на смартфонах или компьютерах.

Тенденция цифровизации процессов и вещей в жизни человека приносит положительные черты в жизнь человека. Уменьшается количество предметов в каждодневном обиходе, многие из них заменяет смартфон. Доступ к средствам для выполнения различных становится легче. Например, раньше при необходимости найти в городе какую-либо организацию, нужно было покупать карту или справочник и искать ее там, теперь необходимо всего лишь установить необходимое приложение.

Можно сказать, что наблюдается стремление к упрощению жизни человека. Но на данный момент еще существуют сферы нашей жизни, которые недостаточно автоматизированы. Например, такой сложный для каждого человека вопрос, что приготовить себе поесть. Существуют приложения, которые помогают подобрать рецепты исходя из предпочтений пользователя (различные национальные кухни, типы диеты или ограничения по здоровью). Подобные приложения решают проблему подбора рецептов, но не другую, не менее важную. Понравившийся рецепт может состоять из сложных ингредиентов, которых не будет у пользователя в наличии, тогда процесс приготовления растянется еще и на покупку необходимых продуктов. Решением этой проблемы становится добавление в концепцию приложения по выбору рецептов по его характеристикам еще и учет ингредиентов, которые содержатся у пользователя. Это позволит подбирать рецепты исходя не только из предпочтений, но и продуктовых возможностей пользователя.

Данная идея приложения кроме установленной проблемы сможет решить еще одну – пользователь получит контроль над своими продуктовыми закупками. Заранее выбрав понравившиеся рецепты, пользователь при походе в магазин будут знать, какие продукты ему необходимо купить.

Целью данной работы является разработка системы для автоматического подбора рецептов исходя из его предпочтений и необходимых продуктов.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Проанализировать предметную область и сформулировать функциональные и нефункциональные требования к приложению.
2. Выбрать технологии и средства реализации приложения.
3. Спроектировать базу данных на основе данных анализа предметной области.
4. Спроектировать приложение.
5. Реализовать приложение.
6. Провести тестирование и отладку, при необходимости внести корректировки и исправить выявленные ошибки.

# Глава 1 Анализ предметной области

Эта глава посвящен анализу предметной области. В ходе анализа будет определено назначение системы и бизнес-процессы, которые необходимо автоматизировать.

## Назначение системы и основные характеристики

Разрабатываемое приложение предназначено для автоматизации подбора рецептов и учета ингредиентов пользователя.

В приложении подразумевается две категории пользователей: посетитель и администратор.

Посетитель – зарегистрированный пользователь, не имеет доступа к редактированию информации о рецептах и продуктах. Ищет и сохраняет понравившееся рецепты.

Администратор – участник команды редактуры. Занимается исправлением и удалением старых, добавлением новых рецептов и продуктов.

У каждого рецепта необходимо выделить следующие характеристики:

* Кухня – это принадлежность рецепта к определенной кухне страны, например русская, итальянская и т. д.;
* Прием пищи – это принадлежность рецепта к приему пищи: завтрак, обед, полдник, ужин;
* Категория – одна из следующих категорий: закуски, первое, второе, десерт, напиток и другие.

Так как подбор рецептов будет может быть основан не только на их характеристиках, но и на продуктах, которыми обладает пользователь, то их необходимо хранить. Продукты, которые имеются у пользователя в наличии, будут храниться в “Холодильнике”.

Кроме того, важно хранить и те продукты, которых пользователю не хватает для приготовления желаемых блюд по рецептам. Для этого необходимо хранить “Список покупок” пользователя.

Для сохранения рецептов, которые понравились пользователю, понадобятся “Избранные” рецепты.

## Определение бизнес-процессов и их операций

Для реализации системы необходима автоматизация операций следующих бизнес-процессов: Учет пользователей, учет продуктов и рецептов администратором, управление холодильником посетителя, взаимодействие посетителя со списком покупок, взаимодействие пользователя со списком избранных рецептов, просмотр рецептов.

Операции перечисленных бизнес-процессов представлены в таблицах 1,2,3,4,5 и 6.

Таблица 1. Операции автоматизируемого бизнес-процесса «Учет пользователей»

| **Название операции** | **Исполнитель** | **Периодичность выполнения** | **Входные данные (документы)** | **Результат выполнения операции (выходные данные, документы)** |
| --- | --- | --- | --- | --- |
| Регистрация посетителя | Посетитель | Один раз, при регистрации нового аккаунта | Фамилия, имя, электронная почта и пароль | Запись в БД о новом посетителе; |
| Регистрация Администратора | Старший администратор | Один раз, при регистрации нового аккаунта администратора, | Фамилия, имя, Электронная почта, пароль | Запись в БД о новом администраторе; |
| Изменение личной информации пользователя | Посетитель, администратор | По необходимости | Код пользователя | Изменение в записи пользователя |
| Удаление аккаунта посетителя | Посетитель, администратор | По необходимости | Код пользователя | Каскадное удаление данных, связанных с конкретным пользователем |
| Просмотр информации пользователя | Посетитель, администратор | По необходимости | Код пользователя | Отображение карточки с информацией пользователя |

Таблица 2. Операции автоматизируемого бизнес-процесса «Учет продуктов и рецептов Администратором»

| **Название операции** | **Исполнитель** | **Периодичность выполнения** | **Входные данные (документы)** | **Результат выполнения операции (выходные данные, документы)** |
| --- | --- | --- | --- | --- |
| Добавление рецепта | администратор | Периодичность не установлена | Название рецепта,  Категория, 10. Прием пищи, кухня, ингредиенты и их количество, шаги рецепта, фотография готового блюда, КБЖУ (калории, белки, жиры,) | Запись в БД нового рецепта |
| Добавление новой категории | Администратор | Периодичность не установлена | Название | Запись в БД новой категории |
| Добавление новой кухни | Администратор | Периодичность не установлена | Название | Запись в БД новой кухни |
| Добавление нового приема пищи | Администратор | Периодичность не установлена | Название | Запись в БД нового приема пищи |
| Добавление продукта в БД | Администратор | Периодичность не установлена | Название продукта, фотография продукта | Запись в БД нового продукта |
| Добавление новой меры измерения | Администратор | Периодичность не установлена | Название меры | Запись в бд новой меры измерения |
| Изменение категории | Администратор | По необходимости | Название | Изменение в БД категории |
| Изменение кухни | Администратор | По необходимости | Название | Изменение в БД кухни |
| Изменение приема пищи | Администратор | По необходимости | Название | Изменение в БД приема пищи |
| Изменение рецепта | Администратор | По необходимости |  | Изменение записи рецепта |
| Изменение продукта | Администратор | По необходимости | Название продукта | Изменение записи продукта |
| Изменение меры измерения | Администратор | По необходимости | Название меры | Изменение записи меры |
| Удаление меры измерения | Администратор | По необходимости | Название меры | Удаление в БД меры |
| Удаление категории | Администратор | По необходимости | Название | Удаление в БД категории |
| Удаление кухни | Администратор | По необходимости | Название | Удаление в БД кухни |
| Удаление приема пищи | Администратор | По необходимости | Название | Удаление в БД приема пищи |
| Удаление рецепта из БД | Администратор | По необходимости |  | Удаление записи рецепта и каскадное удаление всех данных связанных с ним. Удаление записи рецепта из избранного всех пользователей |
| Удаление продукта из БД | Администратор | По необходимости | Код продукта | Удаление записи продукта и всех рецептов с этим продуктом |

Таблица 3. Операции автоматизируемого бизнес-процесса «Взаимодействие посетителя с холодильником»

| **Название операции** | **Исполнитель** | **Периодичность выполнения** | **Входные данные (документы)** | **Результат выполнения операции (выходные данные, документы)** |
| --- | --- | --- | --- | --- |
| Добавление продукта в “холодильник” посетителя | Посетитель | После каждой совершенной покупки | Код продукта, количество, код пользователя | Запись в “холодильник” посетителя нового продукта |
| Изменение количества продукта в “холодильнике” посетителя | Посетитель | При изменении количества определенного продукта | Код продукта, код пользователя | Изменение записи продукта в “холодильнике пользователя” |
| Удаление продукта из “Холодильника” посетителя | Посетитель | По необходимости | Код продукта, код пользователя | Удаление продукта из “холодильника” пользователя |
| Просмотр продуктов, находящихся в “холодильнике” | Посетитель | По необходимости | Код пользователя | Отображение продуктов, находящихся в холодильнике |

Таблица 4. Операции автоматизируемого бизнес-процесса «Взаимодействие посетителя со списком покупок»

| **Название операции** | **Исполнитель** | **Периодичность выполнения** | **Входные данные (документы)** | **Результат выполнения операции (выходные данные, документы)** |
| --- | --- | --- | --- | --- |
| Добавление продукта в “список покупок” пользователя | Посетитель | При недостатке определенного продукта для рецепта | Код пользователя, код продукта | Добавление продукта в список покупок |
| Изменение записи продукта в списке покупок | Посетитель | По необходимости | Код пользователя, код продукта | Изменение в записи продукта |
| Удаление записи продукта из списка покупок | Посетитель | При приобретении нужного продукта | Код пользователя, код продукта | Удаление записи продукта из списка пользователя |
| Просмотр продуктов, находящихся в списке покупок | Посетитель | По необходимости | Код пользователя | Отображение продуктов, находящихся в списке покупок |

Таблица 5. Операции автоматизируемого бизнес-процесса «Взаимодействие посетителя со списком избранных рецептов»

| **Название операции** | **Исполнитель** | **Периодичность выполнения** | **Входные данные (документы)** | **Результат выполнения операции (выходные данные, документы)** |
| --- | --- | --- | --- | --- |
| Добавление рецепта в список избранных рецептов | Посетитель | Периодичность не установлена | Код рецепта, код пользователя | Добавление рецепта в список избранных конкретного пользователя |
| Удаление рецепта из избранного посетителя | Посетитель | По необходимости | Код пользователя, код рецепта | Удаление рецепта из списка избранных посетителя |

Таблица 6. Операции автоматизируемого бизнес-процесса «Просмотр рецептов»

| **Название операции** | **Исполнитель** | **Периодичность выполнения** | **Входные данные (документы)** | **Результат выполнения операции (выходные данные, документы)** |
| --- | --- | --- | --- | --- |
| Просмотр рецептов по заданным ингредиентам и характеристикам | Посетитель | По необходимости | Продукты, характеристики рецепта | Список подходящих рецептов |
| Просмотр рецептов по ингредиентам из “холодильника” пользователя | Посетитель | По необходимости | Код пользователя, продукты из “холодильника” | Список подходящих рецептов |
| Просмотр рецептов по подходящему времени | Посетитель | Каждый завтра, обед, полдник и ужин | Продукты из холодильника пользователя, Прием пищи | Список рекомендованных рецептов |
| Просмотр информации о продукте | Посетитель | По необходимости | Код продукта | Информация о продукте |
| Просмотр рецепта | Посетитель | По необходимости | Код рецепта | Отображение карточки рецепта со всеми необходимыми ингредиентами и шагами рецепта |
| Просмотр необходимых ингредиентов для рецепта | Посетитель | По необходимости | Код рецепта | Отображение всех ингредиентов и их количества для рецепта |

## Описание программных требований

В ходе анализа было установлено, что в системе подразумевается две категории пользователей. Задачи этих пользователей различны, а значит и приложения для них должны отличаться. Следовательно, необходимо разработать два клиентских приложения для доступа к базе данных, но с разным функционалом. Оба приложения будут связываться с базой данных на сервере.

## Описание функциональных требований

Система предполагает два клиентских приложения (для посетителя и для администратора) с разным функционалом, соответственно и требования к ним будут разные.

В приложении для Посетителя должны быть реализованы операции бизнес-процессов, исполнителем которым значится “Посетитель”.

В приложении для Администратора должны быть реализованы операции бизнес-процессов, исполнителем которым значится “ Администратор”.

Среди общих функциональных требований можно выявить:

* просмотр списка рецептов;
* просмотр характеристик рецепта;
* просмотр шагов рецепта.

## Результаты

Таким образом были получены программные требования для реализации системы и функциональные требования, на основе операций описанных бизнес-процессов. Было принято решение о разработке двух клиентских приложений с разным функционалом для доступа к базе данных.

# Глава 2 Технологии и средства реализации приложения

В этой главе будут выбраны и рассмотрены средства реализации и технологии для разработки клиентских приложений, а также удаленной базы данных.

## 2.1 Выбор СУБД для Базы Данных

Для реализации операций и хранения данных, описанных в ходе анализа, потребуется реляционная база данных. Для базы данных требуется определить систему управления базами данных, в которой будут созданы необходимые таблицы и проведены связи между ними. База данных будет размещена на серверной части.

Выбор был сделан в пользу MS SQL Server, так как .NET (платформа разработки) имеет наилучшую совместимость с данной СУБД.

## 2.2 Выбор программных средств реализации

В качестве основного фреймворка выбран .NET версии 6.0. Для реализации интерфейсов выбрана графическая подсистема WPF, так как она позволяет легко реализовать архитектурный шаблон разработки MVVM, который будет описан далее. Платформа разработки WPF поддерживает широкий набор компонентов для разработки приложений, включая модель приложения, ресурсы, элементы управления, графику, макет, привязки данных, документы и безопасность. WPF использует расширяемый язык разметки для приложений (XAML), чтобы предоставить декларативную модель для программирования приложений.

Работа с базой данных MS SQL Server будет осуществляться с помощью Entity Framework. Это объектно-ориентированная технология доступа к данным, является object-relational mapping решением для .NET. Предоставляет возможность взаимодействия с объектами как посредством LINQ в виде LINQ to Entities, так и с использованием Entity SQL. Работа с Базой данных как с объектами класса существенно упрощает процесс написания и отладки кода. Кроме того, Entity Framework позволяет использовать Database-first подход при разработке приложения, то есть для создания классов по сущностям из базы данных не потребуется лишних усилий.

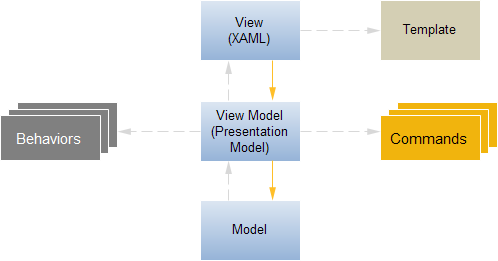
Для разработки прототипов интерфейсов был выбран онлайн-сервис Figma, так как в нем содержатся все необходимые компоненты и возможности для создания интерактивного прототипа приложения.

## 2.3 Выбор шаблона проектирования

Для масштабируемости проекта следует придерживаться одного из архитектурных шаблонов. В будущем это позволит очень легко и просто расширять проект, добавлять новые возможности, удалять старые. В качестве шаблона проектирования был выбран MVVM (Model-View-ViewModel).

Шаблон MVVM имеет три основных компонента: модель, которая представляет бизнес-логику приложения, представление пользовательского интерфейса XAML, и представление-модель, в котором содержится вся логика построения графического интерфейса и ссылка на модель, поэтому он выступает в качестве модели для представления. Диаграмма реализации шаблона MVVM представлена на Рисунок 1.

**Рисунок 1 ­– Шаблон MVVM**



Важное условие MVVM – в модели не должна содержаться информация о модели-представлении, а в модели-представлении не должна содержаться информация о представлении. При выполнении этого условия логика приложения становится независимой от интерфейса, что позволяет оставлять все функции управления прежними и с легкостью менять визуальное оформление.

В MVVM для выполнения действия при нажатии на кнопку вместо обработчиков событий используются команды. Они могут быть как универсальными для всего приложения, так и описываться для конкретной модели-представления.

Для реализации других вещей, не связанных с обработкой данных (например закрытие или перетаскивание окна) используются поведения (Behaviors).

Для сообщения представлению об изменениях в модели-представлении необходимо в последней реализовать интерфейс INotifyPropertyChanged.

Реализация шаблона проектирования MVVM будет описана далее.

# Глава 3 Проектирование Базы данных

Необходимо создать модель данных, по которой будет спроектирована база данных. Для создания модели необходимо определить все основные сущности и связи между ними. При создании модели стоит учесть следующие функциональные зависимости:

* каждый пользователь может добавлять множество рецептов в свой список избранных, рецепты не уникальны для пользователя;
* каждый рецепт может содержать множество ингредиентов, ингредиенты не уникальны для рецепта;
* каждый рецепт может содержать множество шагов, шаги рецепта уникальны для рецепта;
* каждый пользователь может добавлять продукты в свой «холодильник», продукты не уникальны для пользователя;
* каждый пользователь может добавлять продукты в свой «список покупок», продукты не уникальны для пользователя.

При разработке модели данных было выделено универсальное отношение атрибутов. С помощью алгоритмов нормализации модель приведена к Третьей нормальной форме.

Итоговая модель содержит 11 сущностей: Избранное, Пользователь, Холодильник, Мера измерения, Список покупок, Продукт, Категория, Шаг рецепта, Рецепт, Прием пищи, Кухня. Выполняя условия нормализации, таблицы связаны между собой с помощью внешних ключей.

В таблице “Избранное” содержатся: код рецепта и код пользователя, что позволяет связать пользователя и рецепт, который он хочет сохранить.

В таблице “Пользователь” содержатся: код пользователя (идентификатор), фамилия, имя, электронная почта, пароль и информация, является ли пользователь администратором.

В таблице “Холодильник” содержатся: код пользователя, код продукта, количество и код меры измерения. Данная таблица хранит информацию о тех продуктах, которые есть у пользователя в наличии.

В таблице “Мера измерения” содержатся: код меры (идентификатор) и название меры.

В таблице “Список покупок” содержатся: код пользователя, код продукта, количество продукта и код меры измерения. По своей структуре эта таблице идентична таблице “Холодильник”, однако выполняет другую функцию. Данная таблица нужна для сохранения продуктов, которые пользователь собирается купить.

В таблице “Продукт” содержатся: код продукта (идентификатор), название продукта, описание продукта.

В таблице “Категория” содержатся: код категории (идентификатор) и название категории.

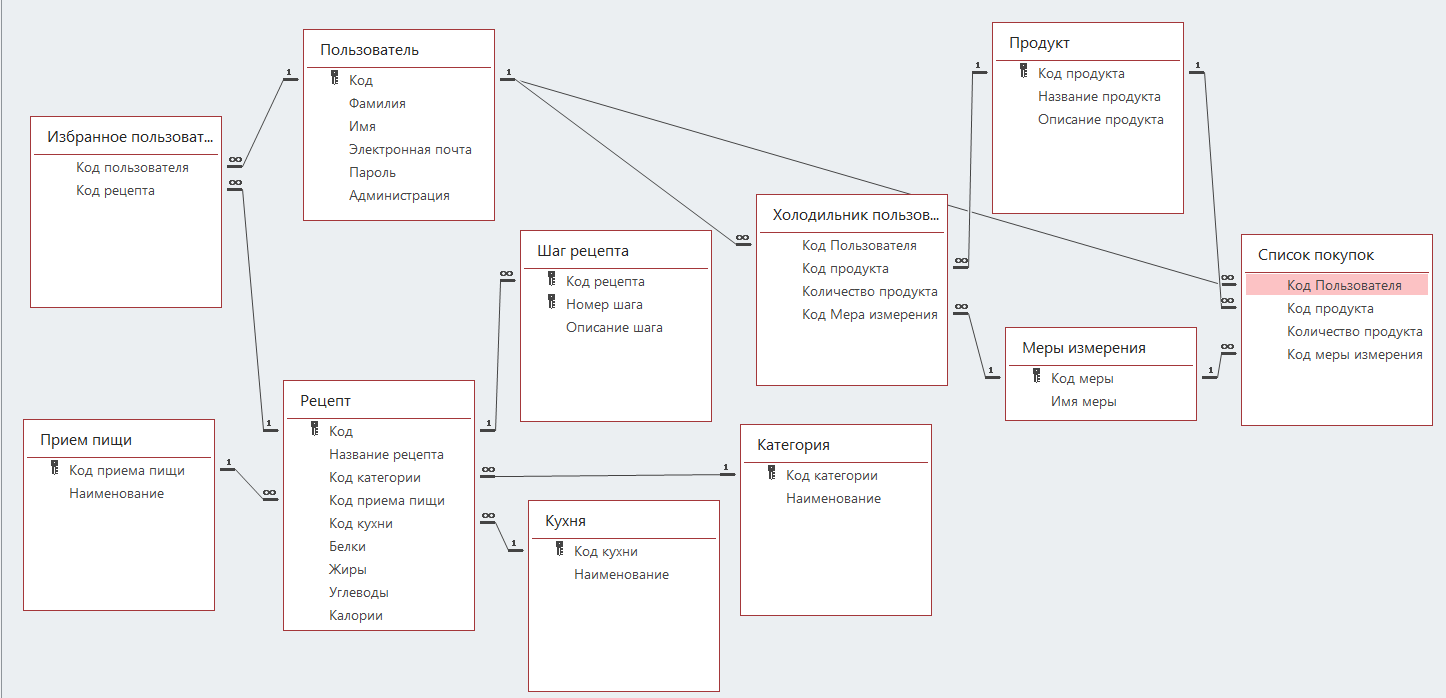
В таблице “Рецепт” содержатся: код рецепта (идентификатор), название рецепта, код категории, код кухни, белки, жиры, углеводы, калории.

В таблице “Шаг рецепта” содержатся: код рецепта, номер шага рецепта, описание.

В таблице “Прием пищи” содержатся: код приема пищи и название приема пищи.

В таблице “Кухня” содержатся: код кухни и название кухни.

Описанные сущности представлены на Рисунке 2.



**Рисунок 2 – Диаграмма сущностей**

Таким образом была получена схема базы данных для дальнейшей реализации.

# Глава 4 Проектирование интерфейсов приложения

При проектировании интерфейса прежде всего стоит распределить различные функции приложения по группам. Это позволит спроектировать страницы.

## 4.1 Выбор цветовой схемы оформления

Для выбора цветовой схемы оформления был использован сайт coolors.co, который подбирает подходящую палитру в зависимости от выбранных пользователем цветов.

Для реализации приложения было выбрано 4 цвета: для фона, для текстов и два контрастных.

Цветовая схема представлена на Рисунке 3.



**Рисунок 3 – Палитра цветов приложения**

## 4.2 Определение общих правил оформления

Для единообразного внешнего вида всех экранов приложения требуется определить некоторые общие правила оформления. Это поможет пользователю составить ассоциации внешнего вида элемента интерфейса и его назначения, что позволит дополнительно улучшить понимание работы приложения.

Крупные заголовки, чтобы быть более заметными, оформляются жирным начертанием контрастного цвета.

Основные элементы навигации – кнопки, полосы прокрутки, выбранные элементы списков – оформляются выделением контрастным цветом. При этом кнопки, которые отвечают за удаление информации, всегда имеют красный контрастный цвет, а для просмотра, изменения или добавления – зеленый. Но навигационные кнопки могут быть как красного, так и зеленого цвета. Неактивные кнопки имеют светлое оформление.

Для понимания, какой элемент является кнопкой, а какой нет, стоит выделять кнопки легкой тенью. Кроме того, тенью выделяются все поля для ввода, так как они тоже являются интерактивными элементами.

Во всех интерфейсных текстах используется шрифт Montserrat.

## 4.3 Распределение функций по экранам приложения для Администратора

Приложение для администратора должно содержать следующие функции:

* авторизация администратора;
* просмотр списка рецептов;
* просмотр списка пользователей;
* просмотр списка продуктов;
* просмотр списка категорий;
* просмотр списка приемов пищи;
* просмотр списка кухонь;
* просмотр списка мер измерений;
* добавление новой меры измерения;
* добавление новой кухни;
* добавление нового приема пищи;
* добавление новой категории;
* удаление категории;
* удаление приема пищи;
* удаление кухни;
* удаление меры измерения;
* редактирование информации о мере измерения;
* редактирование информации о приеме пищи;
* редактирование информации о кухне;
* редактирование информации о категории;
* добавление нового продукта;
* редактирование информации о продукте;
* удаление продукта;
* назначение пользователя администратором;
* просмотр холодильника пользователя;
* просмотр списка покупок пользователя;
* удаление продукта из списка покупок пользователя;
* удаление продукта из холодильника пользователя;
* добавление нового рецепта;
* просмотр рецепта;
* удаление рецепта;
* изменение шага рецепта;
* добавление шага рецепта;
* добавление продукта к рецепту;
* удаление продукта из рецепта;
* удаление шага рецепта;
* изменение информации о рецепте;
* удаление пользователя.

Для начала разделим описанные функции на 4 категории, по элементу взаимодействия. Получившиеся категории:

* рецепты;
* пользователи;
* продукты;
* другое.

Распределим функции по категориям:

Категория “Рецепты”:

* изменение информации о рецепте;
* добавление нового рецепта;
* просмотр рецепта;
* удаление рецепта;
* изменение шага рецепта;
* добавление шага рецепта;
* добавление продукта к рецепту;
* удаление продукта из рецепта;
* удаление шага рецепта;
* просмотр списка рецептов.

Категория “Пользователи”:

* удаление пользователя;
* просмотр холодильника пользователя;
* просмотр списка покупок пользователя;
* удаление продукта из списка покупок пользователя;
* удаление продукта из холодильника пользователя;
* назначение пользователя администратором;
* просмотр списка пользователей.

Категория “Продукт”:

* добавление нового продукта;
* редактирование информации о продукте;
* удаление продукта;
* просмотр списка продуктов.

Категория “Другое”:

* просмотр списка категорий;
* просмотр списка приемов пищи;
* просмотр списка кухонь;
* просмотр списка мер измерений;
* добавление новой меры измерения;
* добавление новой кухни;
* добавление нового приема пищи;
* добавление новой категории;
* удаление категории;
* удаление приема пищи;
* удаление кухни;
* удаление меры измерения;
* редактирование информации о мере измерения;
* редактирование информации о приеме пищи;
* редактирование информации о кухне;
* редактирование информации о категории.

После разбиения на категории в некоторых из все еще слишком большое количество функций, чтобы уместиться на один экран. Поэтому необходимо разделить некоторые категории на подкатегории.

Итоговая схема распределения выглядит следующим образом:

Категория “Другое”:

Подкатегория “Меры измерения”:

* удаление меры измерения;
* редактирование информации о мере измерения;
* просмотр списка мер измерений;
* добавление новой меры измерения.

Подкатегория “Кухни”:

* удаление кухни;
* редактирование информации о кухне;
* просмотр списка кухонь;
* добавление новой кухни.

Подкатегория “Категории”:

* удаление категории;
* редактирование информации о категории;
* просмотр списка категорий;
* добавление новой категории.

Подкатегория “Приемы пищи”:

* удаление приема пищи;
* редактирование информации о приеме пищи;
* просмотр списка приемов пищи;
* добавление нового приема пищи.

Категория “Продукт”:

* добавление нового продукта;
* редактирование информации о продукте;
* удаление продукта;
* просмотр списка продуктов.

Категория “Пользователь”:

Подкатегория “Основное”:

* назначение пользователя администратором;
* просмотр списка пользователей;
* удаление пользователя.

Подкатегория “Холодильник”:

* просмотр холодильника пользователя;
* удаление продукта из холодильника пользователя.

Подкатегория “Список покупок”:

* просмотр списка покупок пользователя;
* удаление продукта из списка покупок пользователя.

Категория “Рецепты”:

Подкатегория “Основное”:

* просмотр списка рецептов;
* добавление нового рецепта;
* изменение информации о рецепте;
* удаление рецепта;
* просмотр информации о рецепте.

Подкатегория “Просмотр информации”:

* изменение шага рецепта;
* добавление шага рецепта;
* добавление продукта к рецепту;
* удаление продукта из рецепта;
* удаление шага рецепта.

Из всех функций нераспределенной осталась только “авторизация администратора”. Поэтому эта функциональность будет вынесена в отдельное окно.

## 4.4 Создание прототипа интерфейса приложения для Администратора

Для создания прототипа воспользуемся сервисом для прототипирования Figma.

Страница для авторизации показан на Рисунке 4. На данной странице расположено название приложения, кнопки закрытия и скрытия окна, кнопка авторизации и два текстовых поля для ввода пароля и электронной почты.



**Рисунок 4 – Прототип страницы авторизации**

Предусмотрено три варианта некорректной авторизации:

1. При некорректном вводе электронной почты появляется сообщение, что такого пользователя не существует.
2. При корректном вводе электронной почты, но при некорректном вводе пароля, система выдаст сообщение, что пароль не верный.
3. Если и электронная почта, и пароль верны, но пользователь не обладает правами администратора, то система ему об этом сообщит.

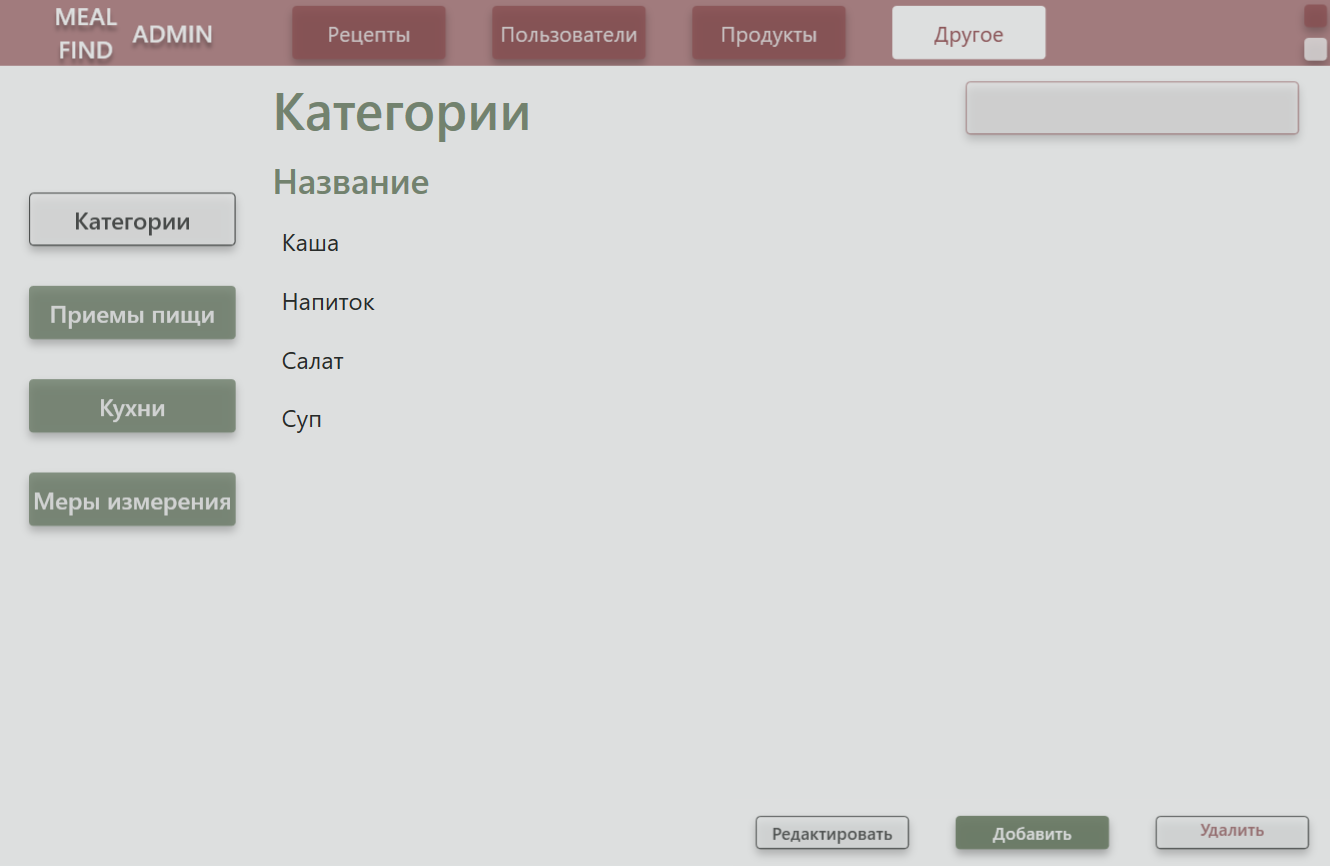
При входе в приложение пользователь видит приветственное сообщение. Данная страница продемонстрирован на Рисунке 5.



**Рисунок 5 – Прототип приветственного страницы**

На данной странице присутствует основная панель управления, которая будет задействована и на остальных страницах. Она необходима для осуществления навигации по основным четырем страницам. В правом верхнем углы видны кнопки закрытия и скрытия приложения. В центре экрана находится приветствие пользователя. Эту страницу можно увидеть только после авторизации, пока никакой пункт меню не выбран.

Страница “Другое” продемонстрирована на Рисунке 6.

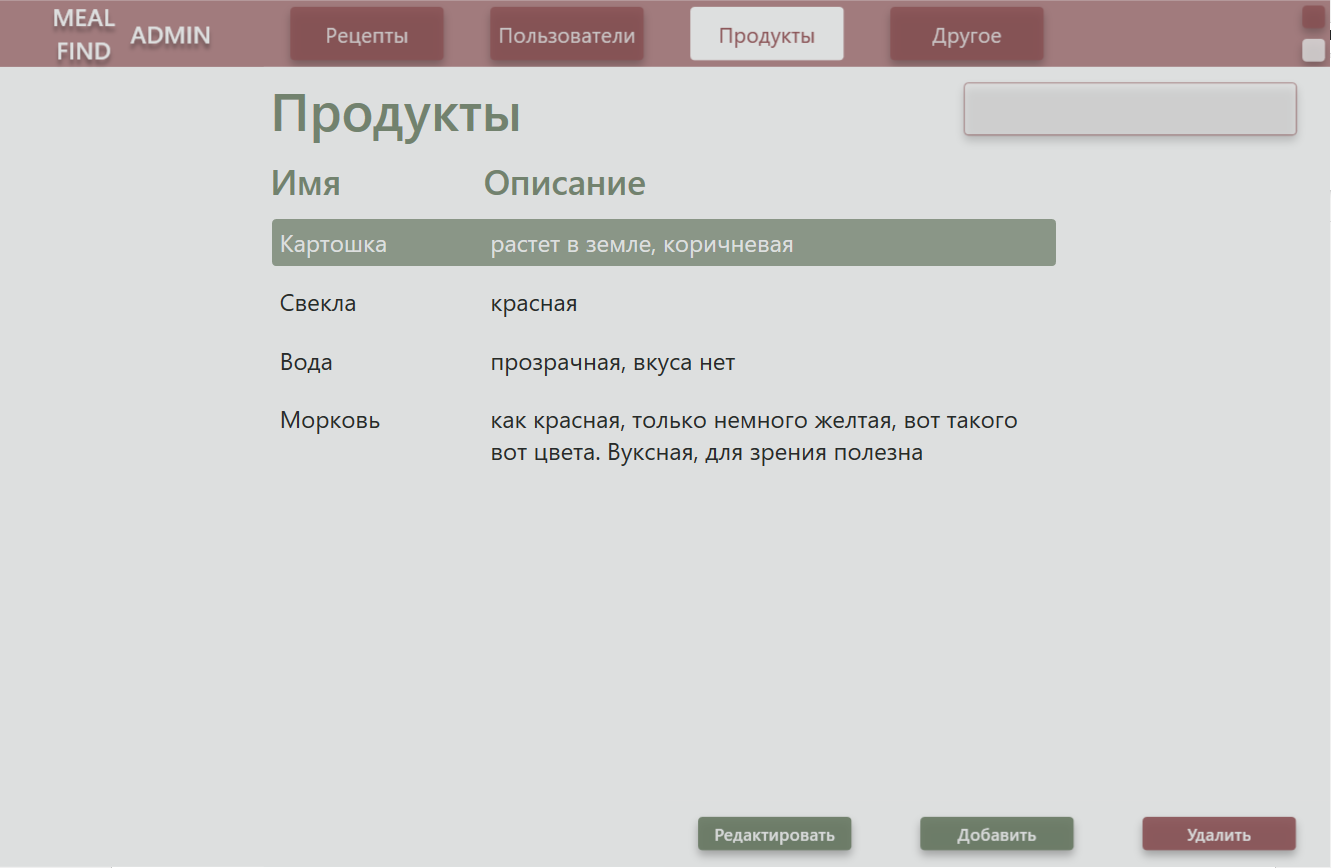


**Рисунок 6 – прототип страницы “Другое”**

При первом открытии этой страницы на экран выводится список категорий. Шаблон для остальных страниц такой же, как и для этой. На странице располагаются ссылки на списки других сущностей и кнопки для редактирования, добавления и удаления записей. Кнопки удаления и редактирования становятся активными только при выборе элементе списка. В одной строке с названием списка расположен поиск среди категорий (или других сущностей) по имени.

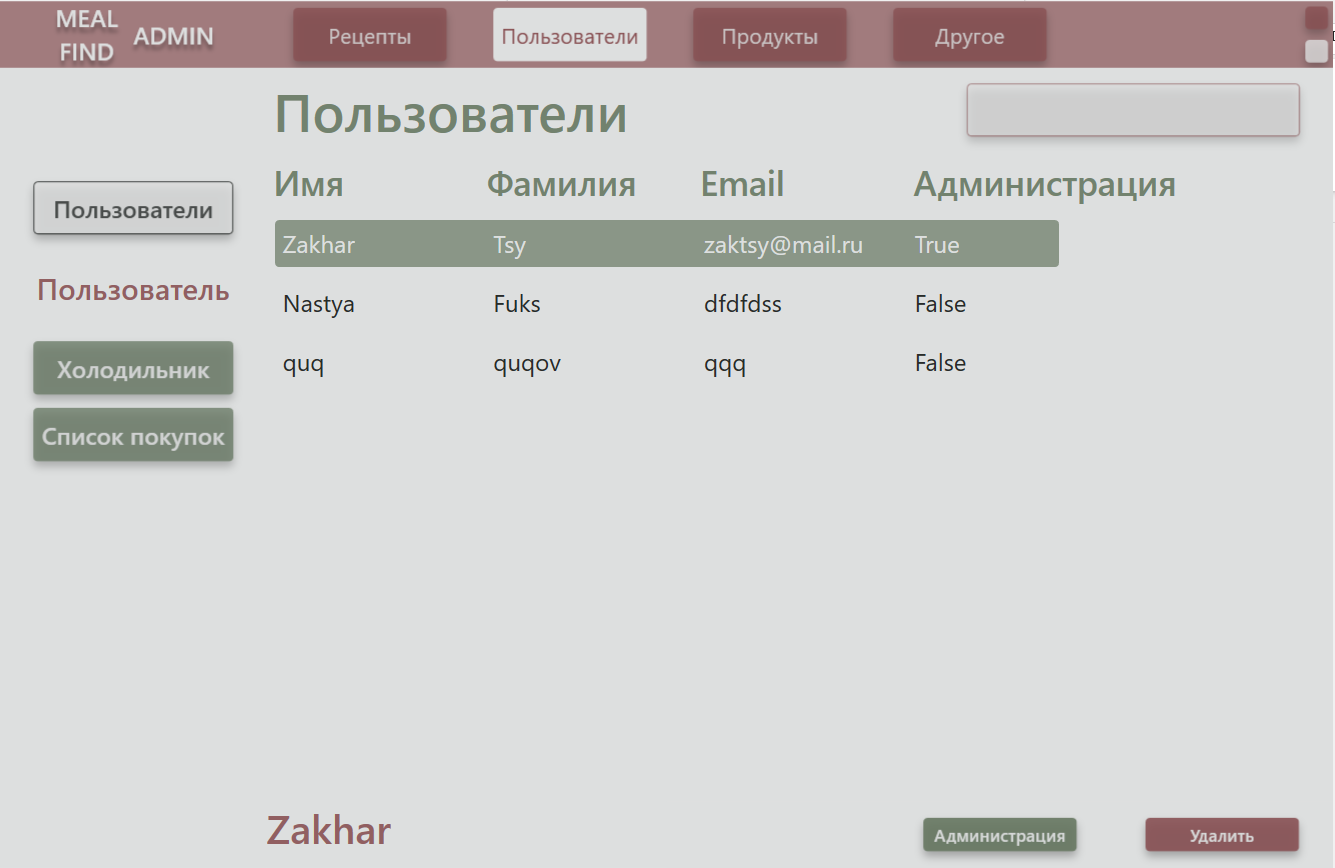
Страница “Продукты” продемонстрирована на Рисунке 7.

На данной странице присутствует список всех продуктов, кнопки для добавления новых и редактирования, удаления старых продуктов. Кроме того, в строке с названием страницы располагается поиск в списке по имени продукта. Кнопки редактирования и удаления становятся активны только при выборе элемента списка.



**Рисунок 7 – прототип страницы “Продукты”**

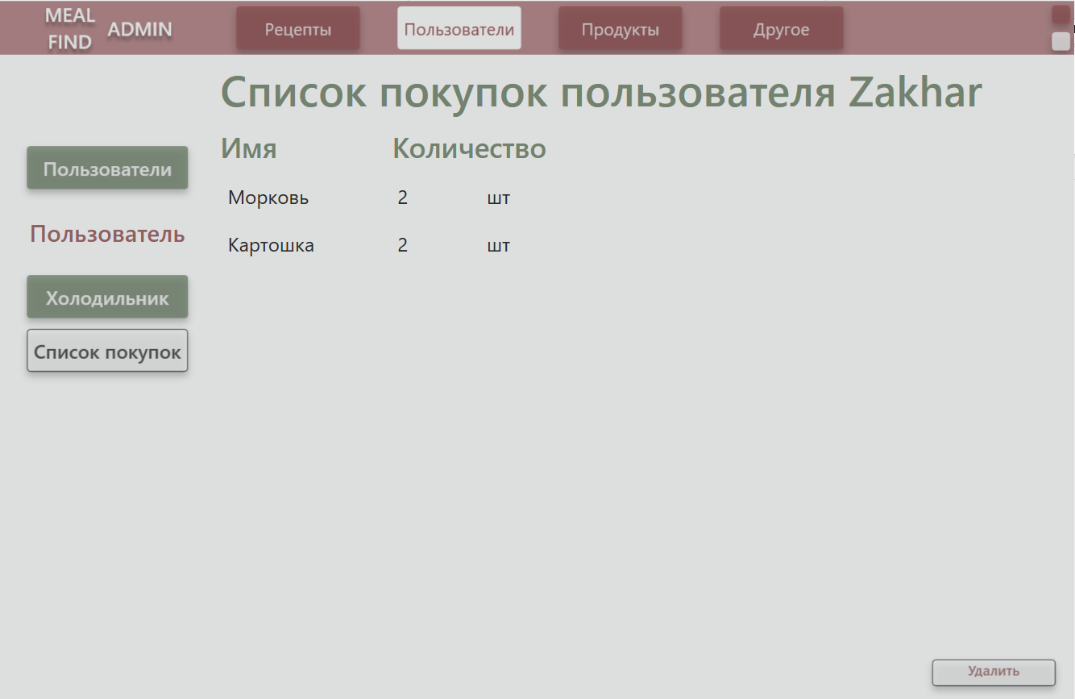
Страница “Пользователи” продемонстрирована на Рисунке 8.



**Рисунок 8 – прототип страницы “Пользователи”**

На данной странице находятся: список пользователей, кнопка удаления пользователя, кнопка смены уровня доступа пользователя и кнопки для отображения холодильника и списка покупок определенного пользователя. Все кнопки активны только при выбранном элементе списка.

Страница “Список покупок” продемонстрирована на Рисунке 9.



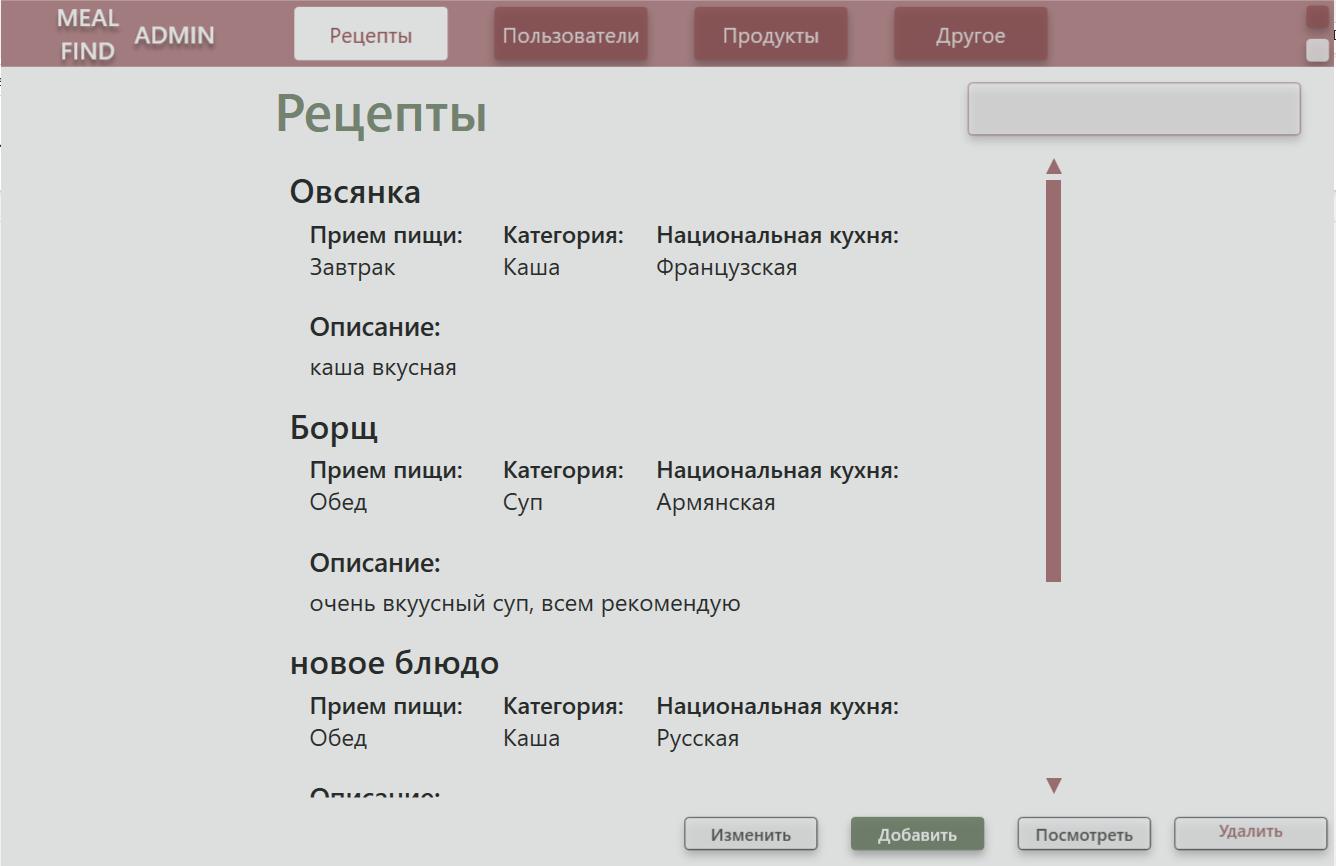
**Рисунок 9 – прототип страницы “Список покупок”**

На данной странице располагается список покупок определенного пользователя. Кнопка удаления продукта из списка покупок становится активна только при выбранном элементе списка. Кнопка открытия холодильника всегда активна.

Прототип страницы для холодильника будет таким же, за исключением заголовков, поэтому не он не демонстрируется.

Страница “Рецепты” продемонстрирована на Рисунке 10.

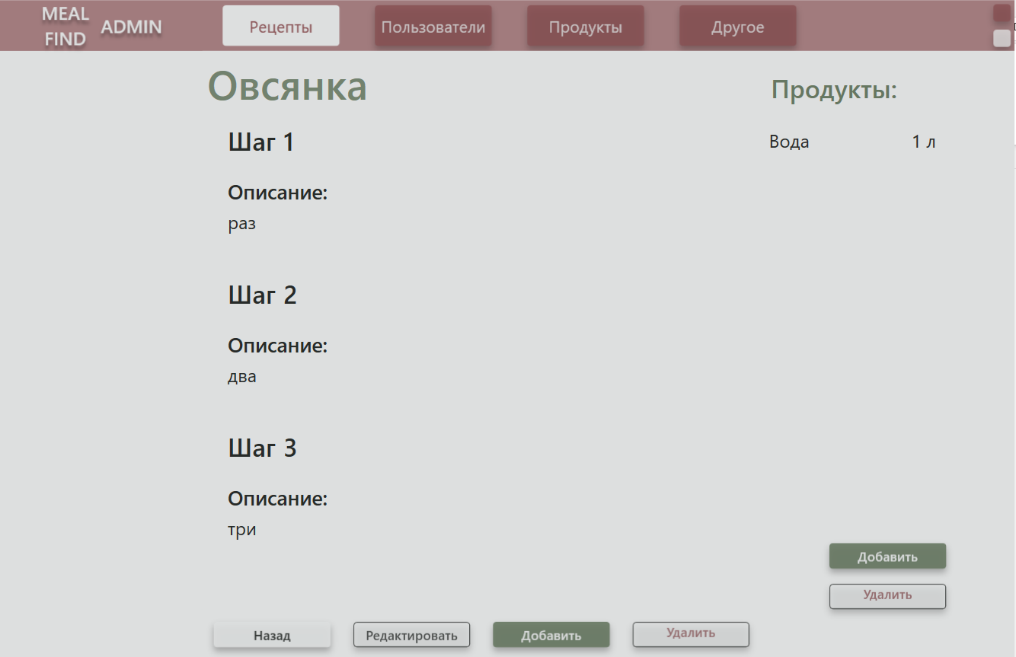
**Рисунок 10 – прототип страницы “Рецепты”**



На данной странице располагается список рецептов. Внизу находятся кнопки для изменения, добавления, просмотра и удаления рецепта, они становятся активны только при выборе элемента из списка. Справа от заголовка страницы находится поиск рецептов по имени.

Страница “Просмотр рецепта” продемонстрирована на Рисунке 11.

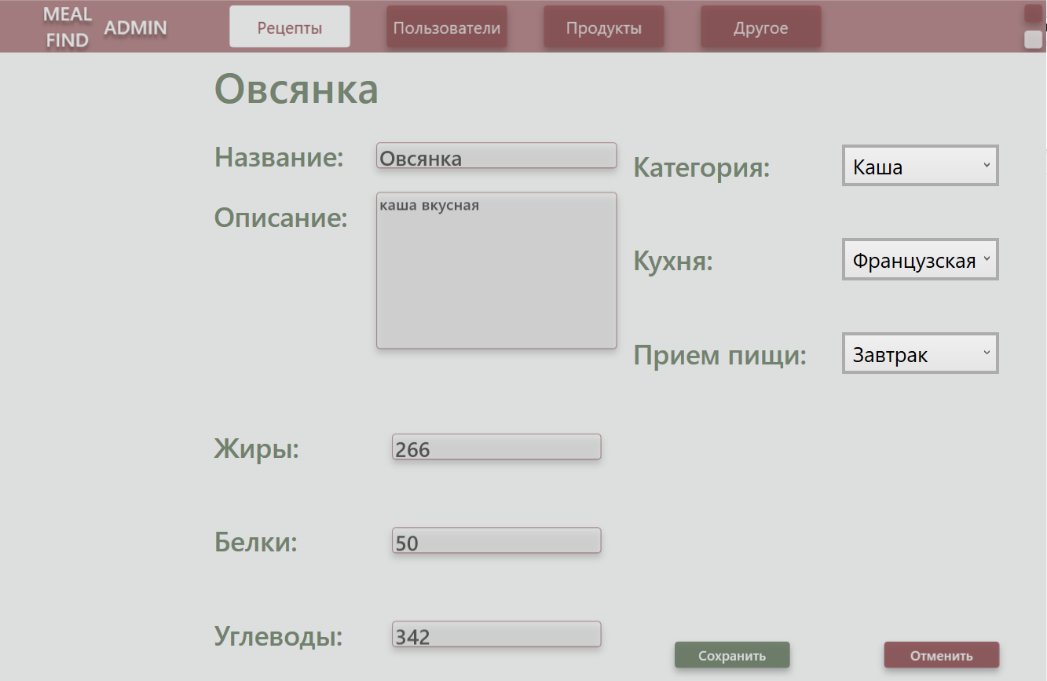
**Рисунок 11 – прототип страницы “Просмотр рецепта”**



На странице расположены списки шагов рецепта и продуктов, необходимых для рецепта. Кнопки внизу позволяют добавлять, редактировать или удалять шаги рецепта. При этом кнопки редактирования и удаления доступны только при выборе шага из списка. Справа расположены кнопки добавления и удаления продуктов. Кнопка “Назад” возвращает на страницу с рецептами.

Страница “Редактирование рецепта” продемонстрирована на Рисунке 12.

**Рисунок 12 – прототип страницы “Редактирование рецепта”**



На данной странице находятся текстовые поля для внесения изменений в информацию о рецепте. Кроме того, присутствуют кнопки сохранения и отмены изменений.

Таким образом были разработаны для приложения Администратора страницы, которые уместили в себе все необходимые функции.

## 4.5 Распределение функций по экранам приложения для Посетителя

Приложение для Посетителя должно содержать следующие функции:

* авторизация Посетителя;
* регистрация Посетителя;
* изменение информации о Посетителе;
* добавление продукта в холодильник;
* добавление продукта в список покупок;
* просмотр продуктов в холодильнике;
* просмотр продуктов в списке покупок;
* удаление продукта из списка покупок;
* удаление продукта из холодильника;
* просмотр списка рецептов;
* просмотр списка избранных рецептов;
* просмотр списка рецептов, состоящих их ингредиентов в холодильнике;
* применение фильтров к списку рецептов;
* добавление рецепта в список избранных;
* просмотр шагов рецепта;
* просмотр продуктов, необходимых для приготовления рецепта;
* добавление продуктов, необходимых для рецепта, в список покупок.

Для начала разделим получившиеся функции на 4 категории, по элементу взаимодействия. Получившиеся категории:

* рецепты;
* покупки;
* холодильник;
* профиль.

Разделим функции по категориям:

Рецепты:

* просмотр списка рецептов;
* просмотр списка избранных рецептов;
* просмотр списка рецептов, состоящих их ингредиентов в холодильнике;
* применение фильтров к списку рецептов;
* добавление рецепта в список избранных;
* просмотр шагов рецепта;
* просмотр продуктов, необходимых для приготовления рецепта;
* добавление продуктов, необходимых для рецепта, в список покупок.

Холодильник:

* добавление продукта в холодильник;
* просмотр продуктов в холодильнике;
* удаление продукта из холодильника.

Список покупок:

* добавление продукта в список покупок;
* просмотр продуктов в списке покупок;
* удаление продукта из списка покупок.

Профиль – редактирование информации о посетителе.

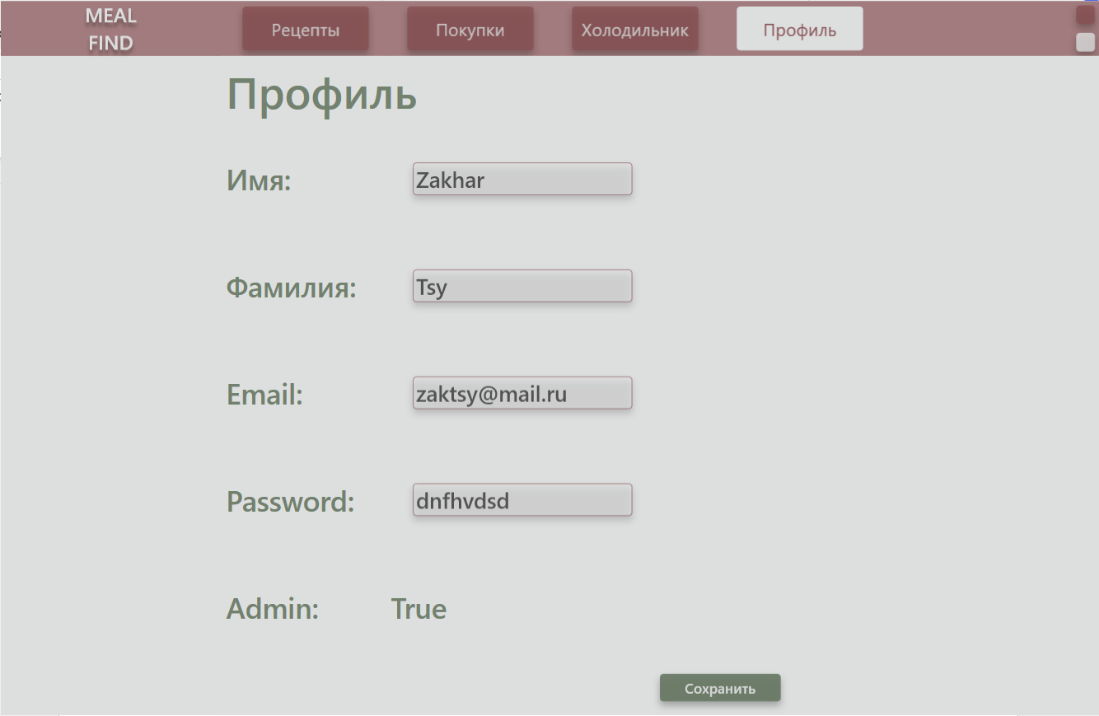
Из всех функций нераспределенными остались только авторизация и регистрация пользователей. Эта функциональность будет вынесена в отдельное окно.

## 4.6 Создание прототипа интерфейса приложения для Посетителя

Для разработки прототипов страниц интерфейса воспользуемся сервисом для прототипирования Figma.

Страница “Профиль” продемонстрирована на Рисунке 13.

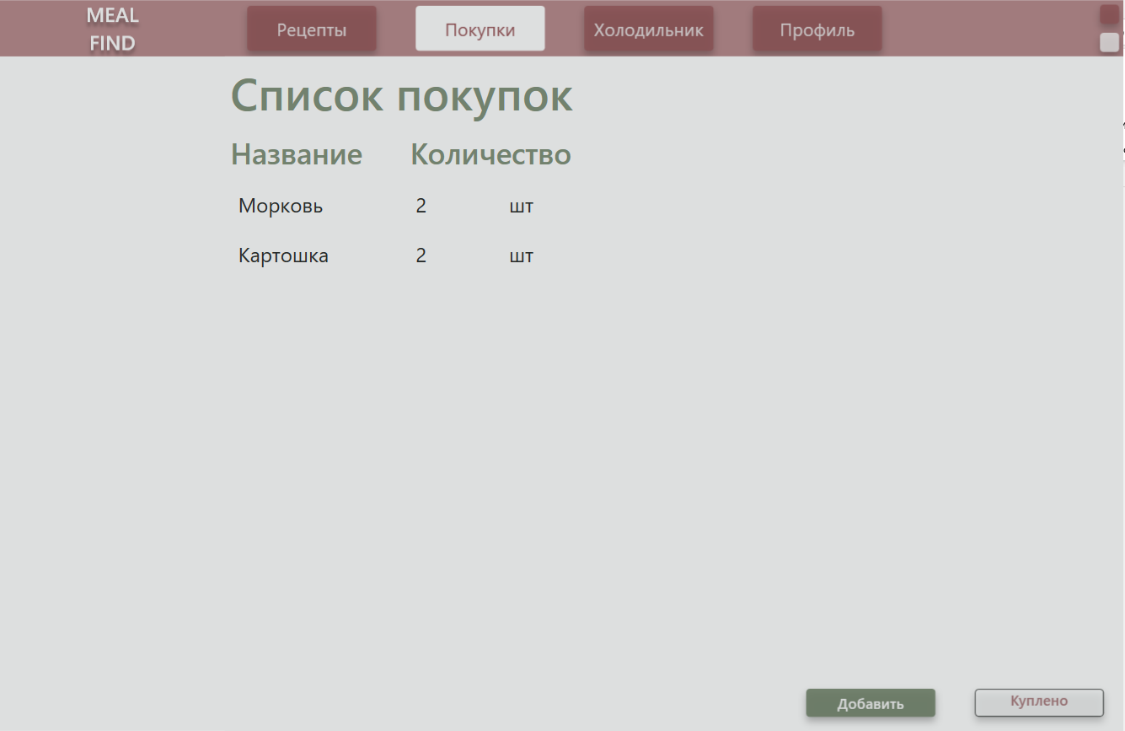
На данной странице находятся текстовые поля для изменения информации о посетителе. Внизу располагается кнопка для сохранения внесенных изменений.



**Рисунок 13 – прототип страницы “Профиль”**

Страница “Список покупок” продемонстрирована на Рисунке 14.

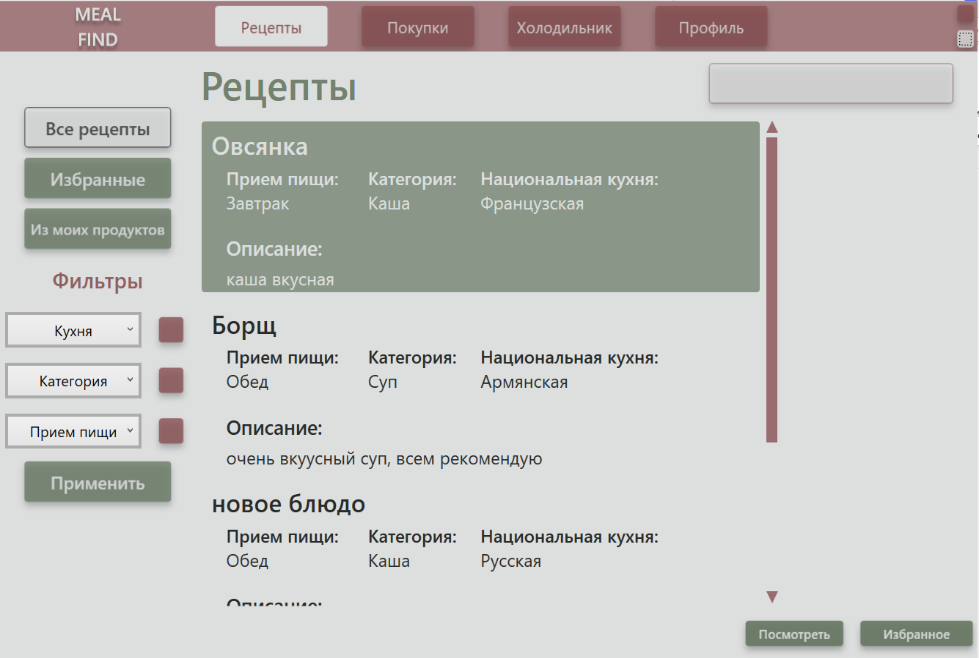
**Рисунок 14 – прототип страницы “Список покупок”**



На данной странице располагается список покупок посетителя. Внизу находятся две кнопки. Первая используется для добавления продукта в список покупок. Вторая нужна для удаления тех товаров, которые были куплены. При нажатии этой кнопки товар перемещается в “холодильник”.

Внешний вид и расположение элементов страницы “Холодильник” отличается только кнопкой “Куплено” (на странице “Холодильник” она называется удалить), которая удаляет продукт из холодильника. Поэтому прототип данной страницы не приводится.

Страница “Рецепты” продемонстрирована на Рисунке 15.

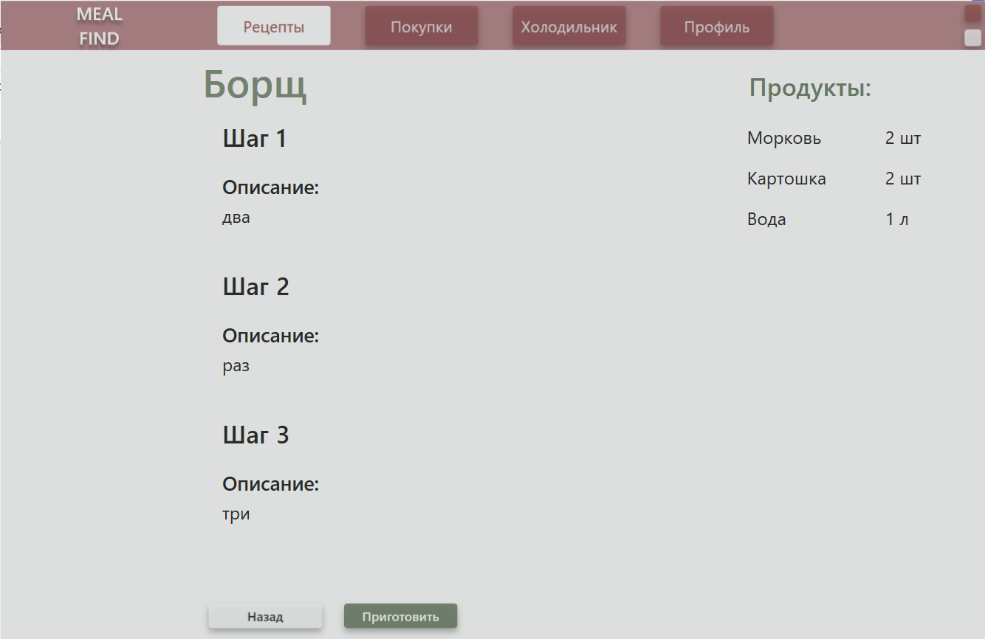


**Рисунок 15 – прототип страницы “Рецепты”**

На данной странице располагаются: список всех рецептов, кнопки для открытия рецептов из списка избранного и рецептов из продуктов холодильника, фильтры для рецептов по кухне, категории и приему пищи, кнопка применения фильтров, кнопки очистки фильтров. Снизу располагаются кнопки для просмотра и добавления рецепта в избранное.

Внешний вид страницы “Избранные рецепты” и “Рецепты из продуктов холодильника” не отличается от страницы “Рецепты”. Расположение всех элементом интерфейса такое же, поэтому эти страны продемонстрированы не будут.

Страница “Просмотр рецепта” продемонстрирована на Рисунке 16.



**Рисунок 16 – прототип страницы “Просмотр рецепта”**

На данной странице располагаются списки шагов рецепта и продуктов, необходимых для приготовления рецепта. Снизу располагаются две кнопки. Первая позволяет вернуться назад к списку рецептов. А вторая действует следующим образом:

1. Если все необходимые продукты содержатся в холодильнике, то они оттуда удаляются.
2. Если продуктов не хватает, то они в нужно количестве добавляются в список покупок, а сам рецепт в список избранных.

## 4.7 Результаты

Таким образом, были разработаны прототипы для страниц, которые включают в себя все определенные ранее функции. По прототипам можно заметить, что многие из них обладают схожими визуальными элементами, что позволяет на этапе разработки оптимизировать работу, создать шаблоны.

# Глава 5 Реализация приложения

После проектирования модели базы данных, интерфейсов и определения основной функциональности приложения можно приступать к его реализации. В этой главе описана разработка базы данных, интерфейсов и функций приложения и собственная реализация шаблона проектирования MVVM. Реализация основных архитектурных компонентов и диалогового сервиса не отличается у приложения для администратора и посетителя. Их разница заключается в конкретных функциях и представлениях. Поэтому описываемые классы архитектурных шаблонов и сервисов применимы к обоим приложениям.

## 5.1 Реализация Базы данных

Для реализации Базы данных была выбрана СУБД MS SQL Server. Данная СУБД является серверной, что означает, что потребуется воспользоваться утилитой SQL Server Management Studio для конфигурирования, управления и администрирования всех компонентов Microsoft SQL Server.

Создаем и настраиваем таблицы в Базе данных с помощью SSMS.

## 5.2 Реализация шаблона MVVM

Для реализации шаблона проектирования MVVM необходимо создать три слоя управления приложения. Model, View и ViewModel. В этом разделе описаны способы и инструменты создания этих архитектурных элементов. Кроме того, функционирование программы, созданной в рамках этого шаблона проектирования, невозможно без дополнительных технологий, таких как команды, поведения и диалоговые окна. Команды будут затронуты в этой части, как непосредственная часть шаблона MVVM, так как без них невозможна работоспособность. Устройство поведений и диалогового сервиса будет рассмотрено в дальнейших частях.

### 5.2.1 Создание классов моделей

В рамках шаблона MVVM модели отвечают за доступ к данным и их хранение. В данном случае являются связью между базой данных и приложением. Для создания классов моделей по существующим таблицам воспользуемся подходом Database First. Это позволит нам не описывать сами модели в коде, Entity Framework сам сделает это работу.

Для работы с существующей БД MS SQL Server нам надо добавить два пакета:

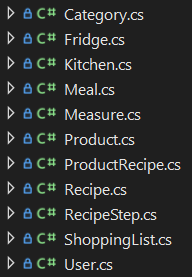
* Microsoft.EntityFrameworkCore.SqlServer (необходим для работы с MS SQL Server);
* Microsoft.EntityFrameworkCore.Tools (необходим для создания классов по базе данных, то есть reverse engineering).

Затем в консоль менеджера пакетов требуется ввести следующую команду:

“Scaffold-DbContext "Data Source=Имя сервера;Initial Catalog=recipesdb;Integrated Security=True;" Microsoft.EntityFrameworkCore.SqlServer”

Данная команда создаст классы моделей, список которых продемонстрирован на Рисунке 17.

**Рисунок 17 – список сформированных моделей**



Пример того, как выглядит класс готовой модели, продемонстрирован на Рисунке 18.



**Рисунок 18 – пример класса модели**

Как пример приведена реализация класса “Прием пищи”. Можно заметить, что помимо атрибутов соответствующей таблицы (которые стали свойствами класса), присутствует еще и виртуальное свойство, которое отвечает за внешнюю связь с таблицей рецептов. Таким образом в Entity Framework описываются связи между таблицами. При этом загрузка этих объектов класса Recipe производится только после специальной команды, то есть коллекция не будет без надобности занимать память.

Помимо классов моделей создается еще и так называемый Контекст базы данных. Это конфигурационный файл, который позволяет приложению связываться с базой данных и содержит описание моделей для их построения.

### 5.2.2 Создание классов представлений моделей

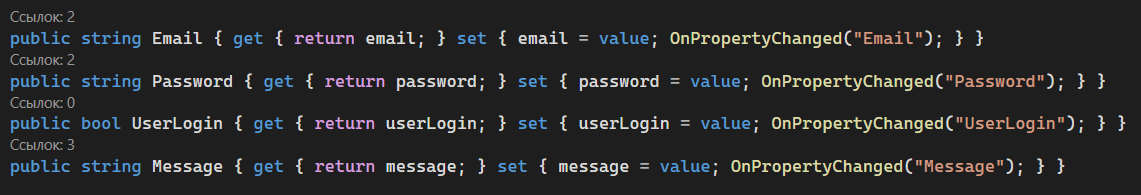
Работа ViewModel, а именно уведомление View о случившихся изменениях осуществляется с помощью интерфейса “INotifyPropertyChanged”. Создадим на его основе класс базовой ViewModel, от которого будут наследованы все остальные представления моделей. Итоговый код класса BaseViewModel продемонстрирован на Рисунке 19.



**Рисунок 19 - класс BaseViewModel**

Рассмотрим, как пример создания ViewModel, класс AuthViewModel приложения для Администратора, который отвечает за авторизацию.

В классе присутствуют свойства, продемонстрированные на Рисунке 20.



**Рисунок 20 – свойства класса AuthViewModel**

Данные свойства отвечают за сохранение информации в полях класса. При установке нового значения через свойство вызывается метод базового класса OnPropertyChanged, который уведомляет представление о изменении во ViewModel. Представление получает информацию из этих свойств с помощью механизма связывания, к котором будет рассказано далее.

Кроме того, в классе присутствует описание команд, необходимых для выполнения функций приложения. Но их устройство будет рассмотрено позднее, в части, объясняющей как работают команды.

По примеру AuthViewModel организованы и другие представления моделей, в них имеются поля данных, свойства для доступа к ним и команды взаимодействия.

Стоит рассмотреть устройство MainViewModel, так как именно через нее осуществляется управление другими представлениями моделей. В MainViewModel имеются следующие поля и свойства доступа к ним:

* LoginedUser - поле типа User, содержит информацию о пользователе, который авторизовался;
* SelectedViewModel – поле типа BaseViewModel, содержит информацию об основной ViewModel, с которой сейчас взаимодействует пользователь;
* SelectedSideViewModel – поле типа BaseViewModel, содержит информацию об боковой ViewModel, с которой сейчас взаимодействует пользователь;
* ViewModels – коллекция объектов всех основных ViewModel, созданных в ходе работы приложения;
* SideViewModels – коллекция объектов всех боковых ViewModel, созданных в ходе работы приложения.

В MainViewModel всего три команды: для закрытия приложения, для скрытия приложения и для переключения представлений моделей. При вызове в команду передается параметр, с какой именно основной и боковой ViewModel пользователю необходимо взаимодействовать. Если данная ViewModel уже была создана и присутствует в коллекции, то управление переходит к ней. Если же такой ViewModel еще не существует, то создается ее объект, помещается в коллекцию и управление переходит к ней. Такой же процесс происходит и с боковыми ViewModel.

### 5.2.3 Создание представлений

Все представления создаются как графические элементы на языке xaml. У каждого приложение есть два окна: для авторизации и остальное. После успешной авторизации первое окно закрывается и открывается основное. Проверка авторизации и переключение окон происходит в классе App (главный класс всего приложения). Код для проверки авторизации и открытия основного окна продемонстрирован на Рисунке 21.



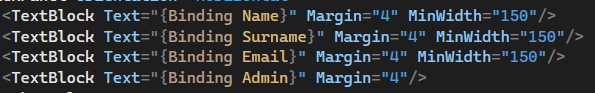
**Рисунок 21 – переключение окон**

Окно MainWindow имеет модульную структуру. Оно состоит из графических элементов (кнопки и текст), а также содержит два элемента ContentControl. В эти элементы встраиваются другие представления, которые имеют тип ContentControl. То есть, в данном случае, как и у моделей представлений, есть главное окно, а все остальные представления – это его часть. В зависимости от управляющей ViewModel меняется и представление.

### 5.2.4 Привязка данных

Как оговаривалось ранее, представление узнает об изменениях во ViewModel с помощью интерфейса INotifyPropertyChanged. Но как же визуальный элемент, например TextBox узнает, какую информацию ему необходимо отображать? Для этого и необходима привязка данных.

Для того, чтобы во View отображать содержимое ViewModel с помощью Data Binding, необходимо задать BindingContext представлению. Им будет являться желаемая ViewModel. Для уточнения, какое именно свойство для обращения к данным необходимо, используется ключевое слово Binding, как продемонстрированно на Рисунке 22.



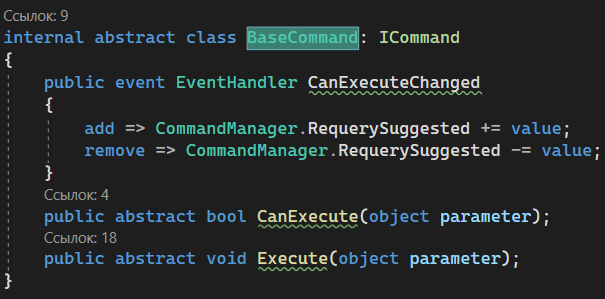
**Рисунок 22 – пример использования Data Binding**

Этот инструмент можно использовать не только для привязки текстов. Например, для ListBox можно в параметр ItemsSource привязать коллекцию объектов из ViewModel.

Для связи ViewModel с View можно не только настроить BindingContext, но и создать в ресурсах приложения DataTemplate. Таким образом все связи между View и ViewModel будут храниться в одно месте, что упрощает поиск.

### 5.2.5 Создание команд

Команды являются привязанными объектами, что позволяет разделить логику и пользовательский интерфейс друг от друга. В WPF команды представлены интерфейсом ICommand. Создадим базовый класс команды, реализующий этот интерфейс. Код класса BaseCommand продемонстрирован на рисунке 23.



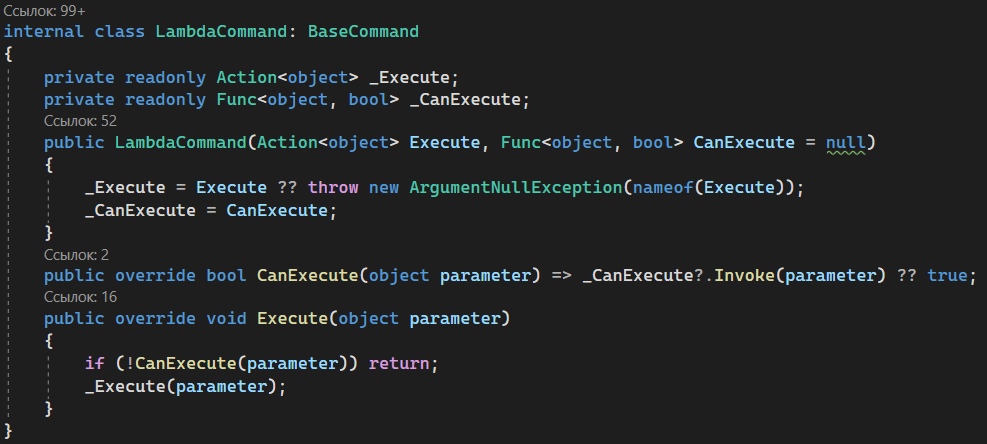
**Рисунок 23 – код класса BaseCommand**

Данный класс создает два виртуальных метода: CanExecute определяет, может ли команда выполняться и Execute, который, собственно, выполняет логику команды. Событие CanExecuteChanged вызывается при изменении условий, указывающий, может ли команда выполняться. Для этого используется событие CommandManager.RequerySuggested.

Методы Execute и CanExecute будут переопределяться наследниками этого базового класса.

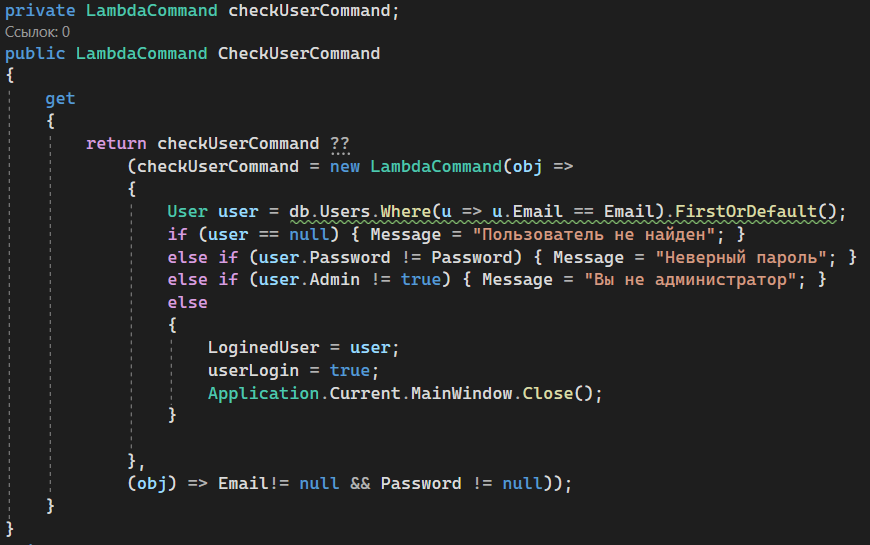
BaseCommand позволяет создать команду как отдельный класс. Этот инструмент подходит для тех операций, которые могут выполняться во многих местах, например, команда для закрытия окна. Но иногда команда используется всего раз во всем приложении, а таких команд большинство. В таком случае нецелесообразно создавать под каждую из них новый класс. Решением в данном случае становится определение команды с помощью лямбда-выражений. Для этого требуется создать класс новой команды, потому что необходимо куда-то записывать лямбда-выражения. Код класса LambdaCommand продемонстрирован на Рисунке 24.

Данный класс содержит поля с делегатами, первый не возвращает значение и отвечает за метод Execute, а второй возвращает логическое значение, отвечает за метод CanExecute. Таким образом, при создании объекта данного класса в конструкторе с помощью лямбда-выражения описать необходимые операции, а команда сама уже их обработает.



**Рисунок 24 – код класса LambdaCommand**

Когда стала понятна логика работы лямбда-команд, можно вернуться к описанию AuthViewModel. Разберем на примере команду CheckUserCommand. Код данной команды продемонстрирован на Рисунке 25.



**Рисунок 25 – код команды CheckUserCommand**

На примере этой команды видна структура оформления подобных элементов. Первым параметром в конструктор помещаются действия, которые должны быть выполнены. Второй параметр – условия выполнения.

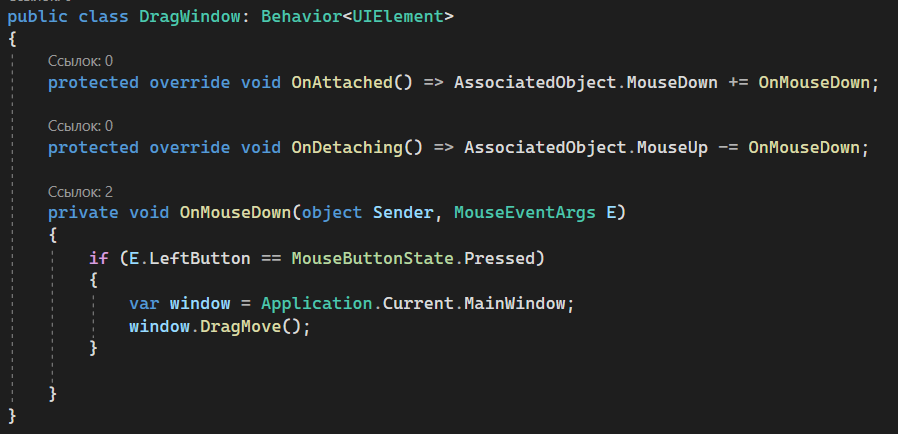
Команды можно назначать кнопкам с помощью упомянутого ранее механизма привязки данных. При этом, от возвращенного методом CanExecute значения будет зависеть, доступна кнопка или нет.

### 5.2.6 Создание поведений

Бывают такие действия, которые необходимо выполнять не только по нажатию кнопки, а всегда. Например, перетаскивание окна за центральную панель главного экрана. Так как спроектированный интерфейс предполагает, что стандартную верхнюю панель системы Windows необходимо убрать (для замены на собственную), то требуется реализовать поддержку данного системного действия.

В таких случаях используют Поведения.

Для их реализации необходим класс Behavior<T> из пакета Microsoft.Xaml.Behaviors. Пример реализации поведения для перетаскивания окна продемонстрирован на Рисунке 26.



**Рисунок 26 – реализация DragWindow**

Для применения этого поведения к элементу интерфейса (а данном случае к верхней панели) необходимо прописать следующее:

<in:Interaction.Behaviors>

<b:DragWindow/>

</in:Interaction.Behaviors>

## 5.3 Реализация интерфейса

Графические интерфейсы в WPF MVVM представлены в виде представлений. Ранее было описано, каким образом происходит реализация этих представлений. В данной части стоит отметить, что верстка всех визуальных элементов выполняется на языке XAML.

В оформлении разных страниц приложения присутствуют повторяющиеся элементы. Целесообразно прописать определенные стили оформления этих элементов. Это позволить сократить количество кода и упростить процесс создания представлений.

Таким образом было создано два ресурсных файла:

* colors.xaml;
* theme.xaml.

В первом содержатся описания основных цветов, используемых в приложении. А во втором – шаблоны визуальных элементов (кнопки, текстовые поля и другие).

## 5.4 Диалоговый сервис

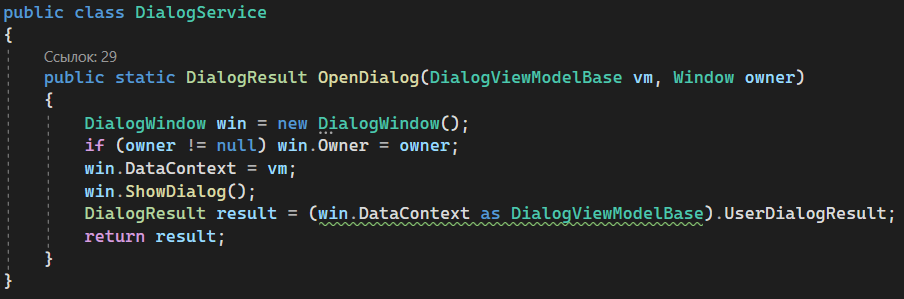
В рамках шаблона MVVM нельзя использовать встроенные в WPF способы взаимодействия с диалоговыми окнами, так как это нарушает принципы MVVM. Поэтому было принято решение создать собственный сервис диалоговых окон, который будет полностью соответствовать концепции выбранного архитектурного шаблона.

### 5.4.1 Dialog Service

В данной части описано создание диалогового сервиса, представление диалогового окна и базовая ViewModel для диалогов.

Для обработки данных после взаимодействия пользователя с диалоговым окном потребуется перечисляемый тип для результата. Перечисляемые типы относятся к классу enum. DialogResult имеет три значения: Undefined, Yes, No.

Код класса DialogService продемонстрирован на Рисунке 27.



**Рисунок 27 – код класса DialogService**

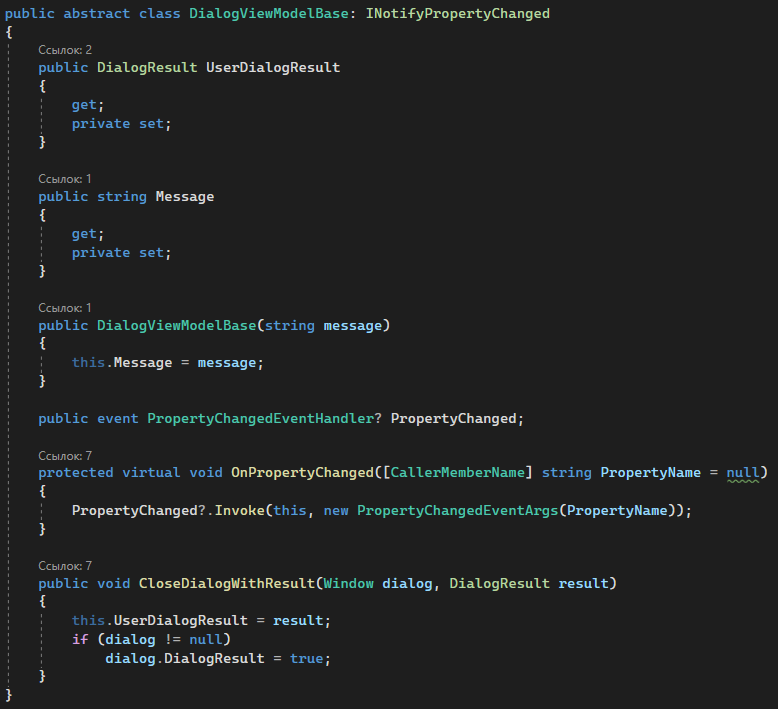
Метод OpenDialog создает базовый View диалога, DataContext которого – выбранная ViewModel. Как описывалось ранее, за каждой ViewModel закреплена ее собственная View в файле App.xaml. Метод возвращает перечислимый тип DialogResult описанный ранее.

### 5.4.2 Базовая ViewModel

Структура базовой ViewModel диалога продемонстрирована на рисунке 28.

Так как это ViewModel, то этот класс реализует методы интерфейса INotifyPropertyChanged. Кроме того, имеет один собственный метод, который отвечает за закрытие диалога при получении результата. Поле Message нужно для вывода пользователю сообщения.

Но DialogBaseViewModel и DialogWindow нельзя использовать сами по себе, потому что они не облают необходимой функциональностью. На основе этих



**Рисунок 28 – код класса DialogViewModelBase**

элементов можно создать разные типы диалоговых окон. Например, окно подтверждения.

### 5.4.2 DialogYesNoViewModel

В данных приложениях потребуется только диалоговое окно, которое возвращает результат Yes или No. Для этих целей был ранее описан DialogResult. Опишем новый тип диалогового окна DialogYesNo. Диалоговые окна данного типа могут как демонстрироваться отдельно, так и служить базой для других типов.

На представлении находится всего три элемента: текстовое поле message для вывода необходимого вопроса, кнопки утвердительного и отрицательного ответов.

DialogYesNoViewModel содержит две команды: для закрытия с положительным или отрицательным ответом. Классы, которые наследуются от DialogYesNoViewModel, могут добавлять свои функции в эти команды, либо использовать совершенно другие.

Для нужд приложений были сконструированы еще четыре других диалоговых окна:

* EditNameDialog;
* EditProductDialog;
* RecipeStepEditDialog;
* ChoiceDialog.

Принцип их реализации схож с таковой у DialogYesNo.

В данной части описана реализация диалогового сервиса, который позволяет осуществлять общения пользователя и приложения.

## 5.5 Реализация запросов к базе данных

Взаимодействие с базой данных происходит с помощью Entity Framework. Соответственно запросы можно осуществлять с помощью LINQ (Language-Integrated Query) to Entities, а не SQL-запросов. Это упростит разработку и понимание принципов работы. Кроме того, это помогает защитить систему от так называемых SQL-инъекций.

Рассмотрим пример составления запроса: db.Users.Where(u => u.Email == Email).FirstOrDefault().

Если переписать данный запрос на язык SQL, то получим следующее:

SELECT \* FROM Users WHERE Email = {Email}

Стоит упомянуть, что запросы LINQ можно переводить и в SQL-подобный синтаксис. Например:

(from user in db.Users where user.Email == Email select user).FirstOrDefault().

Таким образом будут реализованы все существующие запросы. Синтаксис для описания стоит выбирать исходя из функций, которые необходимо использовать. Так, например, SQL-подобный синтаксис не содержит функцию Include необходимую для загрузки связанных данных.

## 5.6 Результаты реализации

Разработаны два приложения (для посетителя и администратора) и база данных. Для пользования приложением и другими людьми можно перенести базу данных на сервер, тогда она будет общая у всех.

В ходе разработки были созданы: собственная реализация архитектурного шаблона MVVM, сервис диалоговых окон, шаблоны для оформления интерфейсов. Бали использованы различные технологии, упрощающие разработку. Произведено тестирование приложения по методу чёрного ящика, выявленные в ходе тестирования ошибки были устранены. Проведенное тестирование выявило наибольшее количество ошибок, чем поиск вручную от других пользователей, которые впоследствии были устранены. Интерфейс готового приложения удобный и понятный.

# Заключение

Результатом работы стали два приложения для пользователей и общая база данных. Для выполнения работы были изучены необходимые материалы по используемым технологиям и архитектурным шаблонам.

В результате анализа предметной области были выявлены бизнес-процессы, которые нуждались в автоматизации. На их основе были собраны функциональные требования, которые в последствии были реализованы с использованием изученных технологий. Были созданы прототипы интерфейсов для приложения, все формы управления окном (закрытие, скрытие и перетаскивание) были переписаны заново ради сохранения единого стиля оформления.

Таким образом разработанное приложения для администратора позволяет быстро и просто получать доступ к редактированию информации и просмотру ее. Разработанное приложение для посетителя позволяет хранить информацию о своих продуктах, подбирать рецепты на их основе, искать необходимые рецепты.

В будущем возможно дальнейшее развитие этих приложений. Например, в планах создать веб-приложение для просмотра рецептов. Кроме него вариантов дальнейшего расширения много, например, мобильные платформы. В будущих версиях .NET ожидается официальный релиз платформы MAUI для кроссплатформенной разработки, которая позволит создать приложения для Android и iOS.

# Приложение А. Репозитории GitHub (Код программ)

Приложение для клиента - <https://github.com/zaktsy/recipe_client.git>

Приложение для администратора - [https://github.com/zaktsy/recipe.git](https://github.com/zaktsy/recipe_client.git)

# Приложение B. Техническое задание

## Название проекта

Разработка приложения для автоматического подбора рецептов.

## Цель проекта

Целью является проектирование и создания приложений (-ий) для автоматического подбора рецептов

Задачи:

1. Проанализировать предметную область.
2. Сформулировать функциональные и программные требования.
3. Спроектировать приложение и выбрать средства реализации.
4. Реализовать спроектированное приложение и составленные требования.
5. Протестировать работу приложения и исправить ошибки.

## Сроки реализации проекта

Проектные цели необходимо реализовать в течение 3-4 модулей 2021-22 учебного года.

## Функциональные требования

В приложении (-иях) должны быть реализованы следующие функции:

1. Учет пользователей системы.
2. Учет продуктов.
3. Учет рецептов и их характеристик.
4. Взаимодействие посетителя с холодильников.
5. Взаимодействие посетителя со списком покупок.
6. Просмотр рецептов.
7. Взаимодействие посетителя со списком избранных рецептов.

## Этапы

Количество этапов и длительность проекта определяется итерационным характером модели жизненного цикла продукта и линейным характером жизненного цикла проекта.

С точки зрения проектного управления проект состоит из стадий:

* инициация,
* планирование,
* исполнение,
* закрытие.

С точки зрения управления жизненным циклом продукта проект состоит из трех итераций, каждая из которых состоит из пяти этапов:

* инициализация,
* проектирование,
* реализация,
* тестирование,
* развертывание.

## Критерии завершённости проекта

Критерием завершенности проекта является наличие работоспособного программного обеспечения, принятого комиссией.

## Критерии качества продукта

|  |  |  |
| --- | --- | --- |
| **Критерий** | **Подтверждение** | **Шкала** |
| Достижимость результата | Задача пользователя была решена | Да/нет |
| Интуитивность использования | При работе с Приложением обращение к разработчикам не потребовалось | Да/нет |
| Дополнительная функциональность | Продукт позволяет решить дополнительные задачи пользователя | Да/нет |
| Внешнее представление | Интерфейс пользователя удобен | 1-10 |

## Форма отчетности

К защите проекта Исполнитель должен предоставить – комплект программной документации, который в себя включает:

* техническое задание;
* отчет по проектной работе;
* исходный код программного обеспечения.

# Приложение C. Руководство пользователя

## Глоссарий

Холодильник – место для хранения всех продуктов пользователя.

Список покупок – место для хранения продуктов, которые пользователь хочет купить (необходимы для рецептов).

Избранное – рецепты, понравившиеся пользователю, которые он собирается приготовить.

## Как добавить продукт в Список покупок?

Чтобы добавить продукт в Список покупок, воспользуйтесь следующими шагами:

1. Откройте вкладку **Список покупок.**
2. Нажмите на кнопку **Добавить**.
3. Найдите продукт, введите количество и выберите меру измерения.

## Как перенести продукт из Списка покупок в Холодильник?

Для этого в списке покупок нужно выбрать необходимый элемент и нажать на кнопку **Куплено**. Приложение автоматически добавит продукт в холодильник.

## Как найти рецепт из моих продуктов?

Чтобы найти рецепт с продуктами из холодильника:

1. Перейдите на вкладку **Рецепты**.
2. В левой боковой панели нажмите на кнопку **Из моих продуктов**.
3. При необходимости примените **Фильтры** к рецептам.

## Как применить фильтры?

К любому списку рецептов можно применить фильтры. Для этого необходимо:

1. Перейдите на вкладку **Рецепты**.
2. В левой боковой панели выбрать необходимую кухню, категорию или прием пищи. Фильтры можно выбирать как по одному, так и сразу несколько.
3. Нажать кнопку **Применить.**

## Как сбросить все фильтры?

Для того, чтобы убрать все фильтры необходимо:

1. Нажать квадратную красную кнопку рядом с каждым фильтром. Эта кнопка сбрасывает фильтр.
2. Нажать на кнопку **Применить.**