МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1-98 01 03 “Программное обеспечение информационной

безопасности мобильных систем”

Специализация Инженер-программист

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Сборник рецептов»

Выполнил студент Александрова Александра Алексеевна

(Ф.И.О.)

Руководитель проекта ст. преподаватель Северинчик Н.А. (учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В .

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: ст. преподаватель Северинчик Н.А. (учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: ст. преподаватель Северинчик Н.А (учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2020

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc41229094)

[1 Обзор аналогов приложений «Сборников рецептов» 4](#_Toc41229095)

[1.1 Приложение «Подбери рецепт» 4](#_Toc41229096)

[1.2 Приложение «ChefTap» 5](#_Toc41229097)

[1.3 Приложение «365 советов про еду» 6](#_Toc41229098)

[1.4 Выводы по главе «Обзор аналогов приложений «Сборников рецептов»» 7](#_Toc41229099)

[2 Проектирование программного продукта 8](#_Toc41229100)

[2.1 Проектирование мобильного приложения 8](#_Toc41229101)

[2.2 Проектирование базы данных 10](#_Toc41229102)

[2.3 Проектирование сервера 12](#_Toc41229103)

[2.4 Выводы по главе «Проектирование программного продукта» 12](#_Toc41229104)

[3 Программная реализация приложений 13](#_Toc41229105)

[3.1 Технические средства для разработки 13](#_Toc41229106)

[3.2 Разработка баз данных 13](#_Toc41229107)

[3.3 Разработка мобильного приложения 14](#_Toc41229108)

[3.4 Разработка сервера 19](#_Toc41229109)

[3.5 Вывод по главе «Программная реализация приложений» 22](#_Toc41229110)

[4 Тестирование разработонного продукта 23](#_Toc41229111)

[4.1 Негативное тестирование 23](#_Toc41229112)

[4.2 Выводы по главе «Тестирование разработонного продукта» 24](#_Toc41229113)

[5 Руководство пользователя 25](#_Toc41229114)

[5.1 Выводы по главе «Руководство пользователя» 28](#_Toc41229115)

[ЗАКЛЮЧЕНИЕ 29](#_Toc41229116)

[Список использованных источников 30](#_Toc41229117)

[ПРИЛОЖЕНИЕ А 31](#_Toc41229118)

[ПРИЛОЖЕНИЕ Б 32](#_Toc41229119)

[ПРИЛОЖЕНИЕ В 33](#_Toc41229120)

[ПРИЛОЖЕНИЕ Г 34](#_Toc41229121)

[ПРИЛОЖЕНИЕ Д 35](#_Toc41229122)

# **ВВЕДЕНИЕ**

Цель курсового проекта: экономия времени пользователей путем хранения и использования рецептов в одном и доступном месте.

Задачи курсового проекта:

1. Изучить аналогичные программные средства по теме курсового проекта.

Данная задача позволит нам изучить похожие программные средства для анализа, где пользователям может быть неудобно в использовании приложения. Также определить, чем новое приложение будет отличатся от уже имеющихся аналогов.

1. Спроектировать приложение.

Проектирование приложения позволит нам увидеть прозрачную картину предстоящей работы целиком и разрабатывать приложение с четким пониманием что, зачем и для кого делается, а также сколько это займет времени.

1. Реализовать приложение.

В курсовом проекте будет реализовано мобильное приложение для хранения и использования рецептов. В век развития информационных технологий этот вариант будет более оптимальный и удобный, так как вся информация будет хранится в одном приложении на телефоне, и пользователь легко сможет им воспользоваться, где бы он не был.

1. Протестировать разработанный продукт.

Это один из важных этапов в процессе создания продукта. Не уделив ему должного внимания, нельзя добиться высокого качества на выходе. Проведется негативное тестирование использование приложения в рамках которого будут применятся сценарии, которые соответствуют внештатному поведению тестируемой системы. Например, исключительные ситуации или неверные данные.

1. Разработать руководство пользователя.

В рамках работы данного курсового проекта будет разработано руководство пользователя, призванное помочь пользователю разобраться с работой программы и её основными функциями. Важно разработать руководство пользователя, даже если вы уверены, что с вашим продуктом легко работать, а его интерфейс интуитивно понятен – всегда найдутся пользователи, которые любят читать инструкции, прежде чем начать освоение продукта.

# **1 Обзор аналогов приложений «Сборников рецептов»**

# **1.1 Приложение «Подбери рецепт»**

В приложении вы найдете простые рецепты и блюда разных стран мира с фотографиями. Все рецепты доступны без интернета. Основная функция приложения – предоставление рецептов. В приложении можно добавить в избранное понравившиеся рецепты и ингредиенты.

Приложение предоставлено скриншоте ниже (рис 1.1.1).

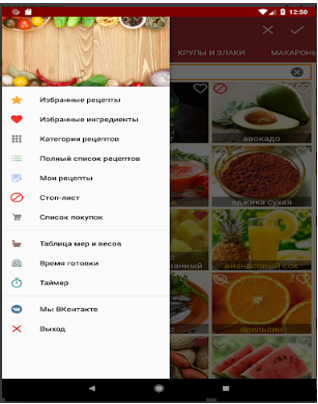


Рисунок 1.1.1 –Приложение «Подбор рецептов»

Понравившимися рецептами можно поделиться в социальных сетях, мессенджерах, по электронной почте и т.д.

Все доступные рецепты можно посмотреть по категориям или общим списком. Также можно использовать дополнительные функции: список покупок, таблица мер и весов, таблица времени приготовления.

# **1.2 Приложение «ChefTap»**

Удобное специализированное приложение, основной функцией которого является сохранение рецептов из интернета. Интуитивно понятное, простое, но в бесплатной версии мало функций.

Общее представление о приложении можно получить, посмотрев скриншот (рис 1.2.1).

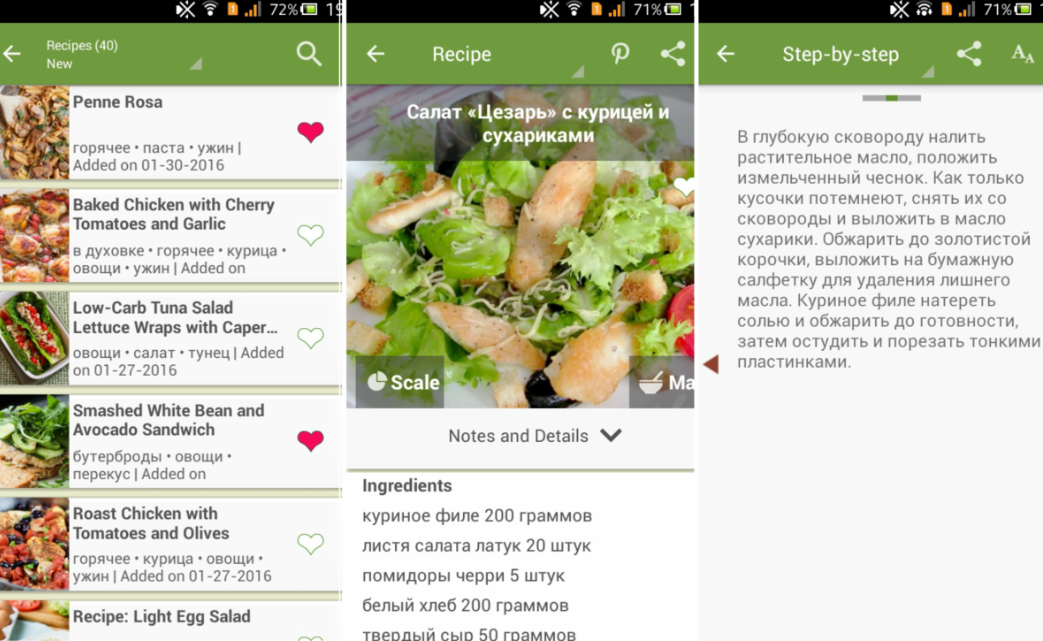


Рисунок 1.2.1 – Приложение «ChefTap»

Данное приложение позволяет поставить метку, если рецепт опробован и понравился, также можно добавить рецепт в Make List. Но в бесплатной версии нет автоматического "подтягивания" рецепта в веб-версии. Также бесплатная версия ограничена только 100 рецептами и нельзя создавать свои рецепты.

# **1.3 Приложение «365 советов про еду»**

Кулинарных рецептов в приложении нет, но вашему вниманию представится множество интересных фактов и советов о готовке. Основная функция приложения – знакомить с новой полезной информацией о еде.

Приложение показано на скриншоте (рис 1.3.1):



Рисунок 1.3.1 – Приложение «365 советов про еду»

Приложение является платным и пробного бесплатного периода у него нет. Из функции существует возможность поставить лайк, если совет был полезен, также поделиться каким-либо советом в Facebook и Twitter.

# **1.4 Выводы по главе «Обзор аналогов приложений «Сборников рецептов»»**

В данной главе были изучены три приложения, которые являются аналогами приложения «Сборник рецептов»: «Подбери рецепт», «ChefTap» и «365 советов про еду». В результате были подведены выводы:

Дизайн у приложений «ChefTap» и «365 советов про еду» достаточно приятный для глаз. Однако дизайн «Подбери рецепт» имеет нагромождение кнопок и текстового контента, который затрудняет восприятие информации. Из этого можно сделать вывод, что дизайн для будущего приложения необходимо делать простой и лаконичный.

Плюсы функционала приложения «Подбери рецепт»: добавление своих рецептов, просмотр рецептов по категориям или полным списком, предоставление таймера в самом приложении, также работа в автономном режиме.

Минусы функционала приложения «Подбери рецепт»: нет возможности изменить свой добавленный рецепт.

Плюсы функционала приложения «ChefTap»: сохранение рецептов из интернета, поиск рецептов по названию.

Минусы функционала приложения «ChefTap»: нельзя добавлять свои рецепты, ограниченный функционал в бесплатной версии.

Плюсы функционала приложения «365 советов про еду»: предоставление множества интересных фактов и советов о готовке.

Минусы функционала приложения «365 советов про еду»: рецептов в приложении нет.

Приложение «Подбери рецепт» является бесплатным и весь функционал приложения доступен.

Приложение «ChefTap» является с ограниченным функционалом, как отмечалось ранее, и за использование всех функций необходимо платить.

Приложение «365 советов про еду» является полностью платным.

# **2 Проектирование программного продукта**

# **2.1 Проектирование мобильного приложения**

В рамках курсового проекта необходимо разработать мобильное приложение позволяющего пользователю добавлять, изменять и просматривать рецепты, выбирать категорию рецептов, просматривать полезную информацию и иметь возможность воспользоваться таймером.

На главной странице приложение должно позволить пользователю выбрать: просмотреть или добавить рецепт. Также в меню главной страницы должен быть предоставлен доступ к таймеру, просмотра информации для готовки и возможности синхронизации с сервером.

Если же пользователь выбирает просмотреть рецепт, то должна быть отображена страница с выбором: посмотреть все рецепты или выбрать категорию.

Если выбирается просмотр всех рецептов, то должны быть предоставлены все рецепты из базы данных в списке и реализован поиск по названию рецепта.

Когда пользователь выбирает рецепты по категориям, должен быть предоставлен список категорий и при нажатии на одну из них, должны выводиться рецепты выбранной категории и реализован поиск по названию рецепта.

Можно ознакомится с UML-диаграммой последовательности для окна «Рецепты по категориям» на рисунке 2.1.1.

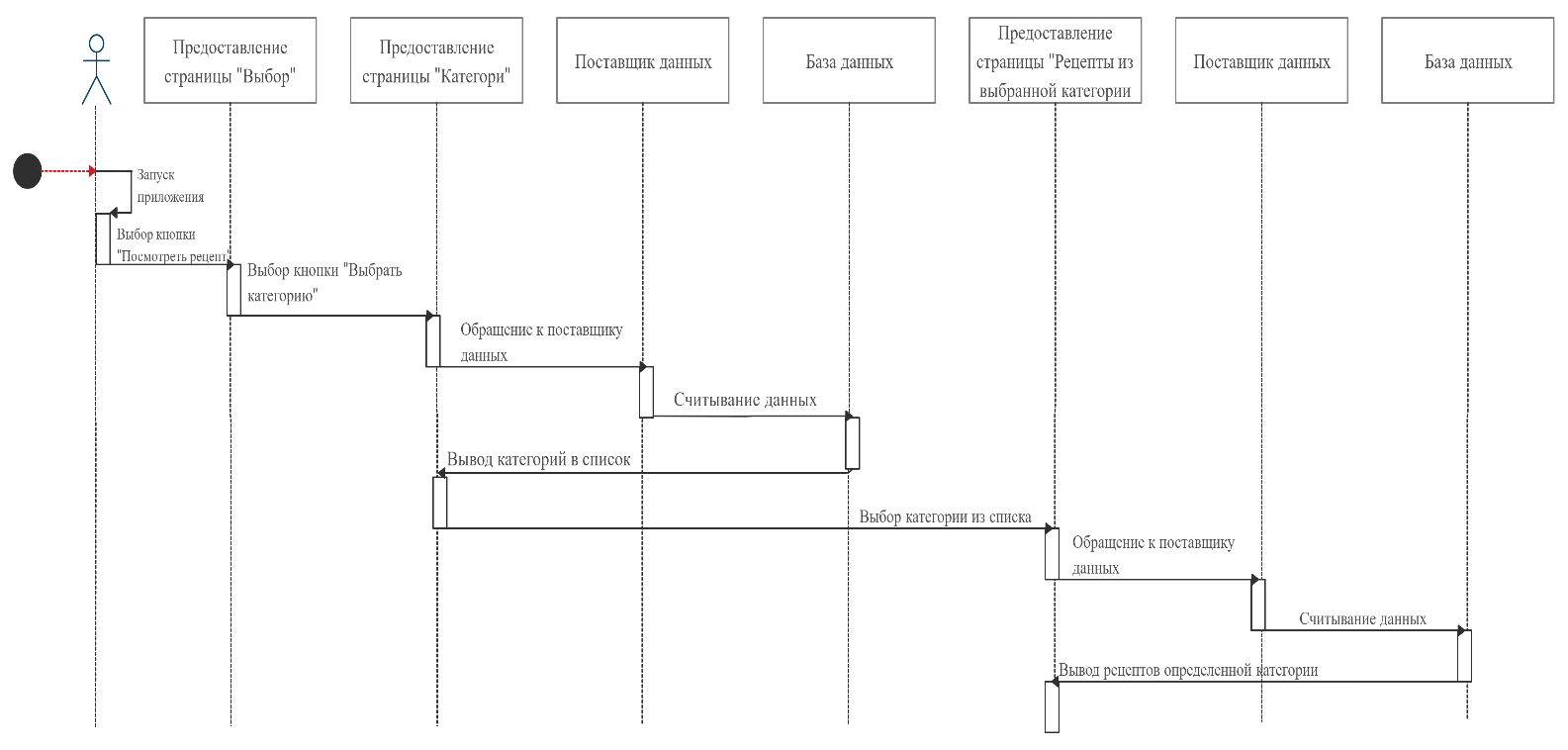


Рисунок 2.1.1 – Диаграмма последовательности

При простом нажатии на один из элементов списка необходимо реализовать страницу для отображения подробной информации о рецепте (название рецепта, ингредиенты и их количество, время приготовления, описание рецепта). При удержании пальца на элементе списка должно появляться меню с выбором: удалить и изменить. Для изменения рецепта должно быть реализована соответствующая страница, а для удаления выплывающее окно с подтверждением действия.

В случае выбора пользователем на главной странице добавления нового рецепта должна быть реализована активность для заполнения информации о рецепте, а именно название рецепта, описание приготовления, время приготовления, категория рецепта, выбор продуктов и количество выбранных продуктов. Также в меню данной страницы должна быть возможность добавления новых продуктов

При выборе таймера должна быть реализована установка часов, минут, секунд, а также кнопки «Старт», «Стоп», «Продолжить», «Сброс». В качестве страницы с полезной информации должна быть реализована галерея с картинками, которые меняются по «свайпу» влево или вправо.

Можно ознакомиться с UML-диаграммой, а именно диаграммой вариантов использования. Она отображает функциональность программного продукта с точки зрения получения значимого результата для пользователя. (рис. 2.1.2)

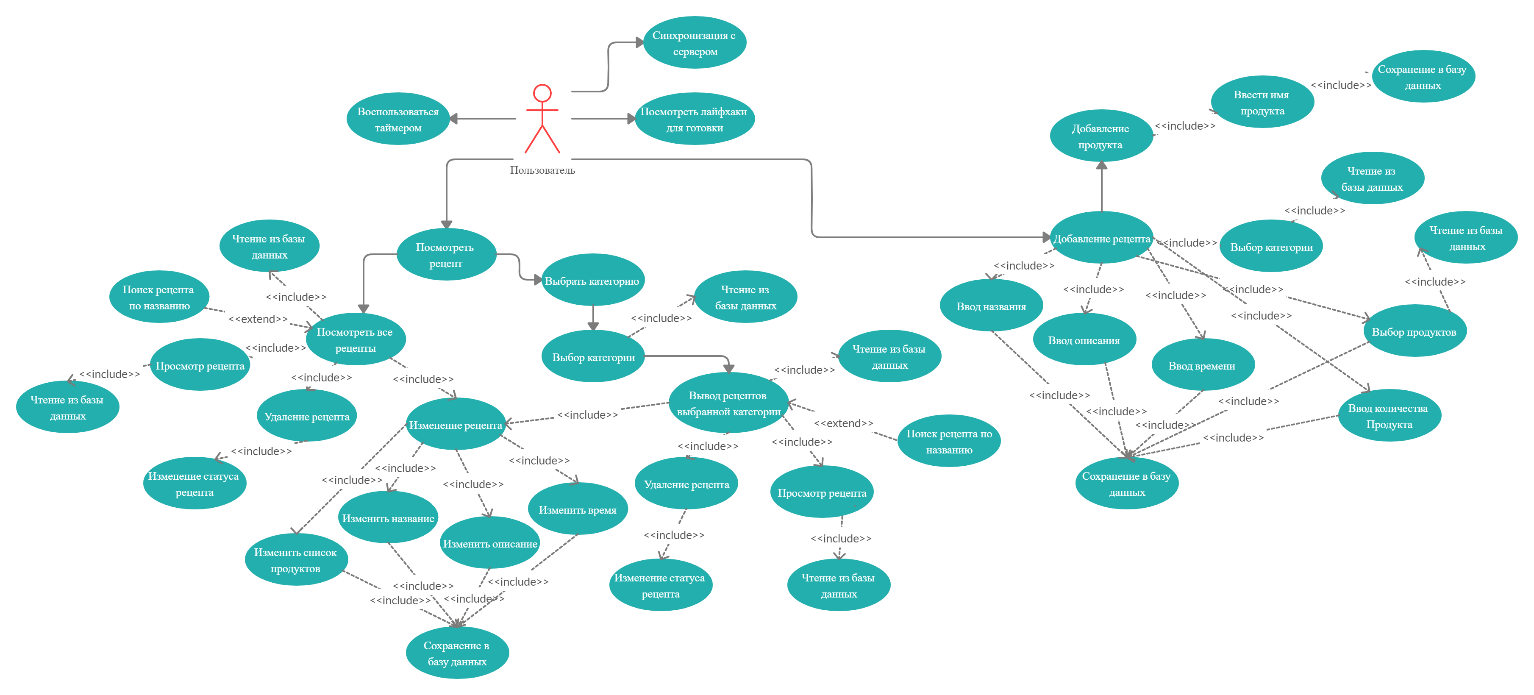


Рисунок 2.1.2 – Диаграмма вариантов использования

Таким образом приложение должно выполнять функции:

* предоставление списка продуктов;
* хранение рецептов;
* выбор рецептов по категориям;
* поиск рецептов по названию;
* добавление рецептов;
* синхронизация данных с сервером;
* предоставление полезной информации для готовки;
* предоставление таймера для готовки;

# **2.2 Проектирование базы данных**

В базе данных данного приложения должны быть реализованы 4 таблицы. В каждой из таблиц необходимо добавить сервисные поля для синхронизации.

В таблице «Product» содержится информация о продуктах. Столбцы данной таблицы:

* IDProduct – целочисленный тип данных, не должен содержать значений null, является первичным ключом;
* Name – строковый тип данных, максимальной длинной 50 символов;
* DateLastChange – целочисленный тип данных, не должен содержать значений null;
* TimeLastChange – целочисленный тип данных, не должен содержать значений null;
* Status – символьный тип данных, максимальной длинной 1 символ.

В таблице «Categorie» хранится информация об ID категории и названии категории и поля для синхронизации с сервером

* IdCategorie – целочисленный тип данных, не должен содержать значений null, является первичным ключом;
* Name – строковый тип данных, максимальной длинной 50 символов;
* DateLastChange – целочисленный тип данных, не должен содержать значений null;
* TimeLastChange – целочисленный тип данных, не должен содержать значений null;
* Status – символьный тип данных, максимальной длинной 1 символ.

В таблице «Recipe» содержится информация о рецептах. Столбцы данной таблицы:

* IdResipe – целочисленный тип данных, не должен содержать значений null, является первичным ключом;
* Name – строковый тип данных, максимальной длинной 50 символов;
* Description – текстовый тип данных;
* TimeCook – строковый тип данных, максимальной длинной 50 символов;
* IdCategorie – целочисленный тип данных, внешний ключ;
* DateLastChange – целочисленный тип данных, не должен содержать значений null;
* TimeLastChange – целочисленный тип данных, не должен содержать значений null;
* Status – символьный тип данных, максимальной длинной 1 символ.

В таблице «Product\_in\_Recipe» хранится информация о ID продукта и ID рецепта для связки таблиц «Product» и «Recipe»

* IdProductInResipe – целочисленный тип данных, не должен содержать значений null, является первичным ключом;
* IdResipe – целочисленный тип данных, внешний ключ;
* IdProduct – целочисленный тип данных, внешний ключ;
* Quantity – строковый тип данных, максимальной длинной 50 символов;
* DateLastChange – целочисленный тип данных, не должен содержать значений null;
* TimeLastChange – целочисленный тип данных, не должен содержать значений null;
* Status – символьный тип данных, максимальной длинной 1 символ.

Схема базы данных представлена на рисунке 2.2.1.

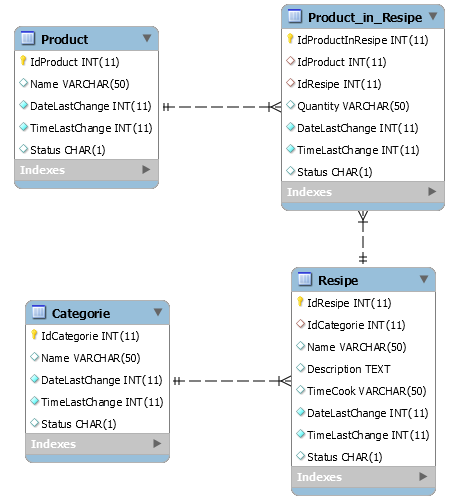


Рисунок 2.2.1 – Схема базы данных

Из данной схемы видно, что таблицы Product и Resipe соединены связью многий-ко-многим посредствам дополнительной таблицы Product\_In\_Recipe, а также таблица Categorie имеет связь с таблицей Resipe один-ко-многим.

# **2.3 Проектирование сервера**

Для синхронизации двух и более клиентов необходимо серверное приложение. Серверная часть приложения должна предоставлять API для доступа к каждой сущности в базе данных. Передаваемые или получаемые данные от сервера должны передаваться в формате JSON.

Серверное приложение должно быть построено на основе архитектурного стиля REST и будет использоваться для взаимодействия компонентов распределённого приложения в сети.

При обращении к серверу по заданному маршруту и определенному HTTP методу сервер будет определять с какой сущностью базы данных необходимо взаимодействовать и какие действия над данными необходимо будет произвести.

При помощи HTTP метода GET сервер будет возвращать данные клиенту, ссылаясь на аргументы в запросе, как на условие для выборки данных. Метод POST в свою очередь позволяет передать данные на сервер для последующей обработки.

Механизм синхронизации между серверным приложением и клиентским приложением представлено схемой в Приложении A.

# **2.4 Выводы по главе «Проектирование программного продукта»**

В данной главе было рассмотрено проектирование каждого отдельного компонента приложения.

По части мобильного приложения был определен основной функционал приложения, также были разработаны диаграмма вариантов использования приложения и диаграмма последовательности одного из вариантов использования, а именно, отображение рецептов из выбранных категорий.

Была также спроектирована база данных, где были определены необходимые таблицы, поля для каждой из них, определены типы данных и ограничения целостности. Также были установлены связи между таблицами, которые описаны на схеме базы данных.

При проектировании серверного приложения был определен архитектурный стиль взаимодействия между клиентом и сервером, формат передаваемых данных, обозначили HTTP методы которые будут использованы в работе с серверной частью.

# **3 Программная реализация приложений**

# **3.1 Технические средства для разработки**

Для разработки приложений курсового проекта были использованы следующие технические средства:

SQLite – компактная встраиваемая СУБД представляющая собой библиотеку.

MySQL – свободная реляционная система управления базами данных.

Android Studio – интегрированная среда разработки для работы с платформой Android.

Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств.

Java – строго типизированный объектно-ориентированный язык программирования.

C# — объектно-ориентированный язык программирования.

.NET Core – это модульная платформа для разработки программного обеспечения с открытым исходным кодом.

Entity Framework – специальная объектно-ориентированная технология на базе фреймворка .NET для работы с данными.

SQLCipher – библиотека с открытым исходным кодом, созданная компанией Zetetic, для прозрачного 256-битного AES шифрования баз данных SQLite.

OkHttp – библиотека предоставляющая собой простой и удобный клиент для работы с HTTP запросами.

Newtonsoft.Json – библиотека позволяющую преобразовывать объекты в JSON формат с большим количеством настроек.

# **3.2 Разработка баз данных**

Для хранения данных на телефоне будет использоваться встраиваемая СУБД SQLite, а для обеспечения безопасности будет использована библиотека SQLCipher. А для сервера будет использована СУБД MySQL.

Код создания структуры мобильной базы данных представлен в Приложении Б, серверной в Приложении В.

Мобильное и серверное приложение имеют общие структурные объекты:

* Product – таблица продуктов;
* Categorie – таблица категорий;
* Recipe – таблица рецептов;
* Product\_In\_Resipe – таблица хранящая продукты относящиеся к рецептам.

Также со стороны мобильного приложения необходимо было добавить триггер «afterSetRemovedStatus\_Remove», который при удалении рецепта меняет статус строк таблицы «ProductInResipe», относящихся к данному рецепту, на «R».

# **3.3 Разработка мобильного приложения**

Мобильное приложение было реализовано в среде разработки – Android Studio 3.0. Для написания приложения был использован язык программирования Java.

Графическая структура проекта представлена на рисунке 3.3.1.

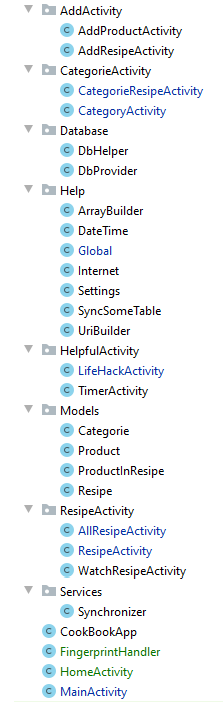


Рисунок 3.3.1 – Графическая структура проекта

Пакет AddActivity хранит в себе классы активностей, которые позволяют добавить новых объекты в базу данных. вариантом (см. таблицу 3.3.1).

**Таблица 3.3.1 – Содержание пакета AddActivity**

|  |  |
| --- | --- |
| Класс AddProductActivity – добавление нового продукта | |
| addProduct\_onClick | Метод привязанный к событию нажатия на кнопку «Добавить продукт» |
| Класс AddRecipeActivity – добавление и изменение рецепта | |
| fillSpinner | Метод выполняющий заполнение выпадающих списков отображаемых в активности |
| setData | Метод заполнения полей ввода и выбора значения из выпадающих списков при изменении рецепта |
| addResipe\_onClick | Метод привязанный к событию нажатия на кнопку «Добавить рецепт» |
| updateResipe\_onClick | Метод привязанный к событию нажатия на кнопку «Изменить рецепт» |

Пакет CategorieActivity хранит в себе классы активностей, которые позволяют отобразить категории и рецепты из выбранной категории (см. таблицу 3.3.2).

**Таблица 3.3.2 – Содержание пакета CategorieActivity**

|  |  |
| --- | --- |
| Класс CategorieResipeActivity – отображение списка категорий | |
| loadLists | Метод загрузки списка категории. |
| Класс CategorieActivity – отображение рецептов из выбранной категории | |
| defaultItemClickOfList | Метод привязанный к событию нажатия на элемент списка |
| defaultLongClickListener | Метод привязанный к событию долго удержания пальца на элементе списка |
| loadLists | Метод загрузки рецептов выбранной категории |

Пакет Database хранит в себе классы для работы приложения с базой данных (см. таблицу 3.3.3).

**Таблица 3.3.3 – Содержание пакета Database**

|  |  |
| --- | --- |
| Класс DbHelper – управления базой данных | |
| onCreate | Метод создания базы данных |
| onUpgrade | Метод вызываемый при смени версии базы данных |
| onConfigure | Метод вызываемый для конфигурации базы данных |
| getReadableDatabase | Метод открывающий соединение с базой данных с помощью пароля, только для чтения |
| getWritableDatabase | Метод открывающий соединение с базой данных с помощью пароля, только для чтения и записи |
| Класс DbProvider – представляет собой поставщик контента | |

Пакет Help хранит в себе вспомогательные классы для разработки приложения (см. таблицу 3.3.4).

**Таблица 3.3.4 – Содержание пакета Help**

|  |  |
| --- | --- |
| Класс ArrayBuilder – преобразование курсора в списки объектов | |
| getCategories | Метод возвращающий список категорий |
| getProducts | Метод возвращающий список продуктов |
| getResipes | Метод возвращающий список рецептов |
| getProductsInResipes | Метод возвращающий список продуктов в рецепте |
| Класс DateTime – работа с датой и временем | |
| getDateSpecialFormat | Метод возвращающий дату в формате "yyMMdd" |
| getTimeSpecialFormat | Метод возвращающий дату в формате "HHmmss" |
| addMinutes | Метод прибавляющий к переданной дате в аргументе определённое количество минут |
| Класс Global – хранит поля доступные всему приложению | |
| Класс Internet – работа с сетью | |
| checkInternetConenction | Метод проверяющий соединение с интернетом |
| serverIsReachable | Метод проверяющий доступен ли сервер |
| Класс Settings – доступ к настройкам приложения | |
| setValue | Метод задающий значения |
| getValue | Метод получающий значения |
| Класс SyncSomeTable – позволяет синхронизировать таблицы с сервером | |
| Update | Метод возвращающий данные с сервера |
| Upgrade | Метод для загрузки данных на сервер |
| Класс UriBuilder – составляет URI для DbProvider | |
| getUri | Метод возвращающий URI составленной на основе аргументов. |

Пакет HelpfulActivity хранит в себе классы активностей, которые позволяют отобразить таймер и советы о готовке. (см. таблицу 3.3.5). Код в классе TimerActivity был написан с помощью источника указанного в списке литературы [3].

**Таблица 3.3.5 – Содержание пакета HelpfulActivity**

|  |  |
| --- | --- |
| Класс LifeHackActivity – отображение картинок | |
| setPositionNext | Метод возвращающий следующее положение картинки |
| setPositionPrev | Метод возвращающий предыдущее положение картинки |
| makeView | Метод создающий картинку в ImageView |
| onTouchEvent | Метод возвращающий событие касания на картинку |
| onFling | Метод возвращающий событие «свайпа» картинки влево и вправо |
| Класс TimerActivity – управление таймером | |
| setTime | Метод устанавливающий время |
| startTimer | Метод запускающий таймер |

**Продолжение таблицы 3.3.5– Содержание пакета HelpfulActivity**

|  |  |
| --- | --- |
| pauseTimer | Метод ставящий таймер на паузу |
| resetTimer | Метод сбрасывающий таймер |
| updateCountDownText | Метод обновляющий текст в TextView |
| updateWatchInterface | Метод обновляющий интерфейс страницы при старте/конце/паузе таймера |

Пакет Models хранит в себе классы, которые позволяют представить данные базы данных в виде объектов (см. таблицу 3.3.6).

**Таблица 3.3.6 – Содержание пакета Models**

|  |  |  |
| --- | --- | --- |
| Класс Categorie – описание объекта категория | | |
| idCategorie | Свойство содержащее идентификатор категории | |
| Name | Свойство содержащее имя категории | |
| dateLastChange | Свойство содержащее дату последнего изменения категории | |
| timeLastChange | | Свойство содержащее время последнего изменения категории |
| Status | | Свойство содержащее статус категории |
| Класс Product – описание объекта продукт | | |
| idProduct | | Свойство содержащее идентификатор продукта |
| Name | | Свойство содержащее имя продукта |
| dateLastChange | | Свойство содержащее дату последнего изменения продукта |
| timeLastChange | | Свойство содержащее время последнего изменения продукта |
| Status | | Свойство содержащее статус продукта |
| Класс ProductInResipe – описание объекта продукт в рецепте | | |
| idProductInResipe | | Свойство содержащее идентификатор продукта в рецепте |
| idProduct | | Свойство содержащее идентификатор продукта |
| idResipe | | Свойство содержащее идентификатор рецепта |
| Quantity | | Свойство содержащее количество продукта |
| dateLastChange | | Свойство содержащее дату последнего изменения продукта в рецепте |
| timeLastChange | | Свойство содержащее время последнего изменения продукта в рецепте |
| Status | | Свойство содержащее статус продукта в рецепте |
| Класс Resipe – описание объекта рецепт | | |
| idResipe | | Свойство содержащее идентификатор рецепта |
| idCategorie | | Свойство содержащее идентификатор категории |
| Name | | Свойство содержащее имя рецепта |
| Description | | Свойство содержащее описание рецепта |

**Продолжение таблицы 3.3.6– Содержание пакета Models**

|  |  |
| --- | --- |
| timeCook | Свойство содержащее время приготовления рецепта |
| dateLastChange | Свойство содержащее дату последнего изменения рецепта |
| timeLastChange | Свойство содержащее время последнего изменения рецепта |
| Status | Свойство содержащее статус рецепта |

Пакет ResipeActivity хранит в себе классы активностей, которые позволяют отобразить все рецепты, описание отдельного рецепта и активность выбора посмотреть все рецепты или выбрать категорию. (см. таблицу 3.3.7). Ознакомится с кодом класса ResipeActivity можно в Приложении Г.

**Таблица 3.3.7 – Содержание пакета ResipeActivity**

|  |  |
| --- | --- |
| Класс AllResipeActivity – отображение списка рецептов | |
| defaultItemClickOfList | Метод привязанный к событию нажатия на элемент списка |
| defaultLongClickListener | Метод привязанный к событию долго удержания пальца на элементе списка |
| loadLists | Метод загрузки всех рецептов |
| Класс ResipeActivity – отображение описания конкретного рецепта | |
| Класс WatchResipeActivity – отображение кнопок «Все рецепты» и «Выбрать категорию» | |
| allResipe\_onClick | Метод привязанный к событию нажатия на кнопку «Все рецепты» |
| categorie\_onClick | Метод привязанный к событию нажатия на кнопку «Выбрать категорию» |

Пакет Services хранит в себе класс синхронизации (см. таблицу 3.3.8).

**Таблица 3.3.8 – Содержание пакета Services**

|  |  |
| --- | --- |
| Класс Synchronizer– сервис для синхронизации | |
| onHandleIntent | Метод отрабатывающий при старте сервиса |
| SyncCategorieTable | Метод синхронизации таблицы Categorie |
| SyncProductTable | Метод синхронизации таблицы Product |
| SyncResipeTable | Метод синхронизации таблицы Resipe |
| SyncProductInResipeTable | Метод синхронизации таблицы ProductInResipe |

Также можно ознакомиться с основными классами, которые не входят ни в один из пакетов на таблице 3.3.9.

**Таблица 3.3.9 – Основные классы**

|  |  |
| --- | --- |
| Класс CookBookApp – стартовая точка приложения | |
| Класс HomeActivity – главная активность приложения | |
| add\_onClick | Метод привязанный к событию нажатия на кнопку «Добавить рецепт» |
| watch\_onClick | Метод привязанный к событию нажатия на кнопку «Посмотреть рецепт» |
| Класс MainActivity – активность с аутентификацией FingerPrint | |
| generateKey | Метод генерирующий ключ для шифрования |
| cipherInit | Метод шифрующий данные |
| start\_onClick | Метод привязанный к событию нажатия на кнопку «Добро пожаловать» |
| Класс FingerprintHandler – класс обрабатывающий результаты сенсора | |
| onAuthenticationHelp | Метод вызывающийся в случае грязных пальцев, недостаточно сильного зажима пальца |
| onAuthenticationFailed | Метод вызывающийся в случае считывания отпечатка пальца, но он не верен |
| onAuthenticationError | Метод вызывающийся в случае 5 неудачных считываний отпечатка пальца |
| onAuthenticationSucceeded | Метод вызывающийся в случае, если все прошло успешно |

Ознакомится с диаграммой классов разработанного приложения можно на рисунке 3.3.2.

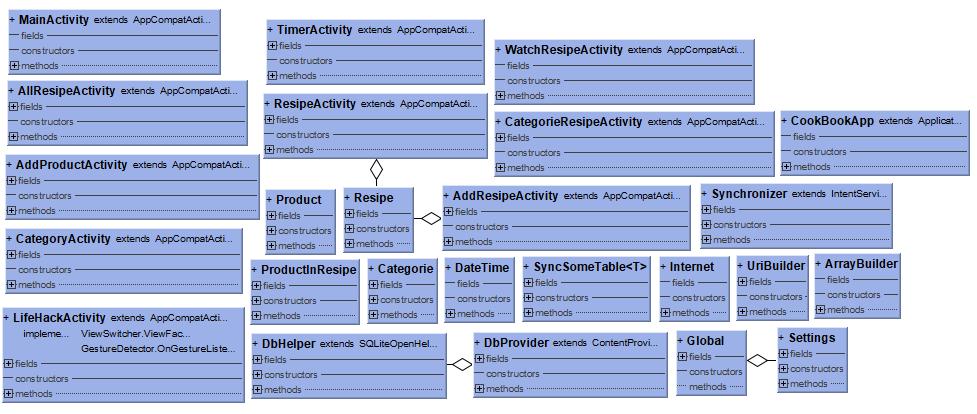


Рисунок 3.3.2 – Диаграмма классов

Из этой диаграммы видно, что есть классы Resipe, DbHelper и Global, которые имеют между классами ResipeActivity, AddResipeActivity, DbProvider и Settings отношение – агрегация, то есть ассоциация при отношении между целым и его частями.

# **3.4 Разработка сервера**

Сервер был реализован в среде разработки – Microsoft Visual Studio 2019. Для написания приложения был использован язык программирования C#.

Графическая структура проекта представлена на рисунке 3.4.1.

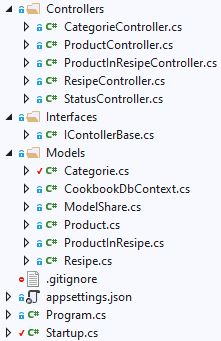


Рисунок 3.4.1 – Графическая структура проекта

Папка Controllers хранит в себе классы контроллеров обрабатывающие поступающие запросы к серверу(см. таблицу 3.4.1). ). Ознакомится с кодом класса CategorieController можно в Приложении Д.

**Таблица 3.4.1 – Содержание папки Controllers**

|  |  |
| --- | --- |
| Класс CategorieController – контроллер таблицы категорий | |
| Update | Метод отправляющий данные клиенту для обновления |
| Upgrade | Метод обрабатывает полученные данные для обновления |
| AddNewRecord | Метод добавляющий новую запись, если та отсутствует в базе данных |
| Класс ProductController – контроллер таблицы продукты | |
| Update | Метод отправляющий данные клиенту для обновления |
| Upgrade | Метод обрабатывает полученные данные для обновления |
| AddNewRecord | Метод добавляющий новую запись в таблицу продукты, если та отсутствует в базе данных |

**Продолжение таблицы 3.4.1– Содержание папки Controllers**

|  |  |
| --- | --- |
| Класс ProductInResipeController – контроллер таблицы продукт в рецепте | |
| Update | Метод отправляющий данные клиенту для обновления |
| Upgrade | Метод обрабатывает полученные данные для обновления |
| AddNewRecord | Метод добавляющий новую запись в таблицу продукты, если та отсутствует в базе данных |
| Класс ResipeController – контроллер таблицы рецепт | |
| Update | Метод отправляющий данные клиенту для обновления |
| Upgrade | Метод обрабатывает полученные данные для обновления |
| AddNewRecord | Метод добавляющий новую запись в таблицу продукты, если та отсутствует в базе данных |
| Класс StatusController – контроллер определяющий работу сервера | |
| GetInfo | Метод возвращающий информацию о текущем времни |

Папка Interfaces содержит класс IControllerBase интерфейс в котором описаны методы, используемые в контроллерах.

Папка Models хранит в себе классы, которые позволяют представить данные базы данных в виде объектов (см. таблицу 3.4.2).

**Таблица 3.4.2 – Содержание пакета Models**

|  |  |
| --- | --- |
| Класс Categorie – описание объекта категории | |
| IdCategorie | Свойство содержащее идентификатор категории |
| Name | Свойство содержащее имя категории |
|  |  |
| DateLastChange | Свойство содержащее дату последнего изменения категории |
| TimeLastChange | Свойство содержащее время последнего изменения категории |
| Status | Свойство содержащее статус категории |
| Класс CookbookDbContext – определяет контекст данных, используемый для взаимодействия с базой данных | |
| Класс ModelShare – содержит общий код для каждой модели | |
| Класс Product – описание объекта продукт | |
| IdProduct | Свойство содержащее идентификатор продукта |
| Name | Свойство содержащее имя продукта |
| DateLastChange | Свойство содержащее дату последнего изменения продукта |
| TimeLastChange | Свойство содержащее время последнего изменения продукта |
| Status | Свойство содержащее статус продукта |

**Продолжение таблицы 3.4.2– Содержание пакета Models**

|  |  |
| --- | --- |
| Класс ProductInResipe – описание объекта продукт в рецепте | |
| IdProductInResipe | Свойство содержащее идентификатор продукта в рецепте |
| IdProduct | Свойство содержащее идентификатор продукта |
| IdResipe | Свойство содержащее идентификатор рецепта |
| Quantity | Свойство содержащее количество продукта |
| DateLastChange | Свойство содержащее дату последнего изменения продукта в рецепте |
| TimeLastChange | Свойство содержащее время последнего изменения продукта в рецепте |
| Status | Свойство содержащее статус продукта в рецепте |
| Класс Resipe – описание объекта рецепт | |
| IdResipe | Свойство содержащее идентификатор рецепта |
| IdCategorie | Свойство содержащее идентификатор категории |
| Name | Свойство содержащее имя рецепта |
| Description | Свойство содержащее описание рецепта |
| TimeCook | Свойство содержащее время приготовления рецепта |
| DateLastChange | Свойство содержащее дату последнего изменения рецепта |
| TimeLastChange | Свойство содержащее время последнего изменения рецепта |
| Status | Свойство содержащее статус рецепта |

Также можно ознакомиться с основными классами, которые не входят ни в одну из папок на таблице 3.4.3.

**Таблица 3.4.3 – Основные классы**

|  |
| --- |
| Класс Program – определяет начальную точку входа в приложение |
| Класс Startup – определяет конфигурацию приложения |

# **3.5 Вывод по главе «Программная реализация приложений»**

В данной главе было рассмотрена разработка каждого отдельного компонента приложения.

На первоначальном этапе были рассмотрены технологии, которые применены для разработки курсового проекта.

Вторым этапом была разработка структуры базы данных на стороне сервера и клиента.

Третий этапом стала разработка мобильного приложения, где была представлена графическая структура приложения, рассмотрен каждый класс и их методы, в частности.

Четвертый этап – это разработка серверного приложения, где была представлена графическая структура приложения, рассмотрен каждый класс и их методы.

# **4 Тестирование разработонного продукта**

Процесс тестирования продукта представляет собой столь же неотъемлемую часть процесса разработки, как и проектирование. Тестирование позволяет оценить качество разрабатываемого продукта.

# **4.1 Негативное тестирование**

В рамках курсового проекта будет выполнено негативное тестирование. Негативное тестирование – это один из видов тестирования, который позволяет проверить систему на некорректное поведение. В ходе такого тестирования мы можем узнать, что система справится с непредвиденными ситуациями.

Предусмотрена следующая обработка исключительных ситуаций:

* в случае ввода неверного опечатка пальца, не верного отпечатка пальца 5 раз, отмены ввода отпечатка пальца предусмотрена обработка данных ситуаций (рис 4.1.1)

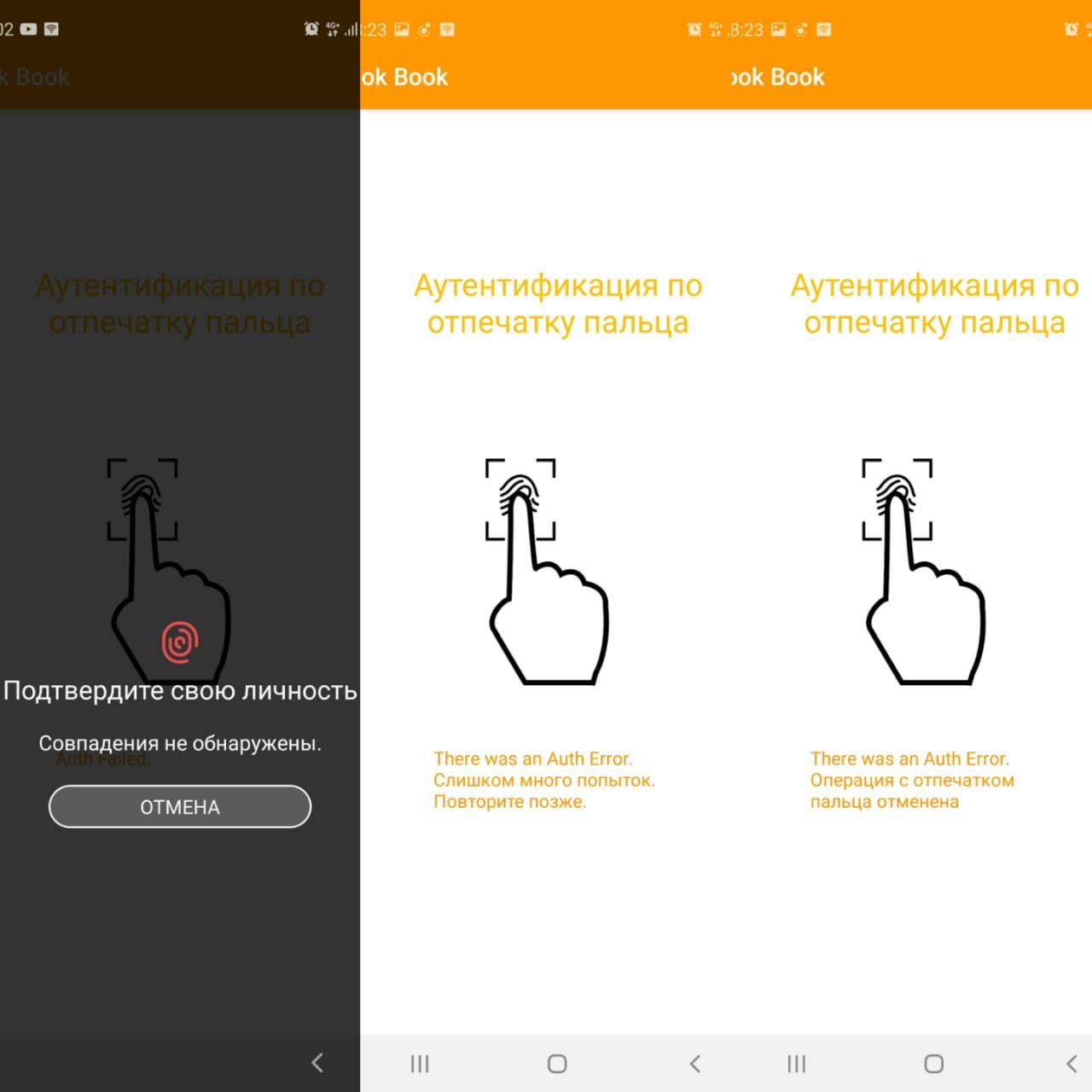


Рисунок 4.1.1 – обработка ошибок в аутентификации

* в случае ввода в поле времени «0» на странице таймера будут всплывать предупреждающее сообщение (рис. 4.1.2).

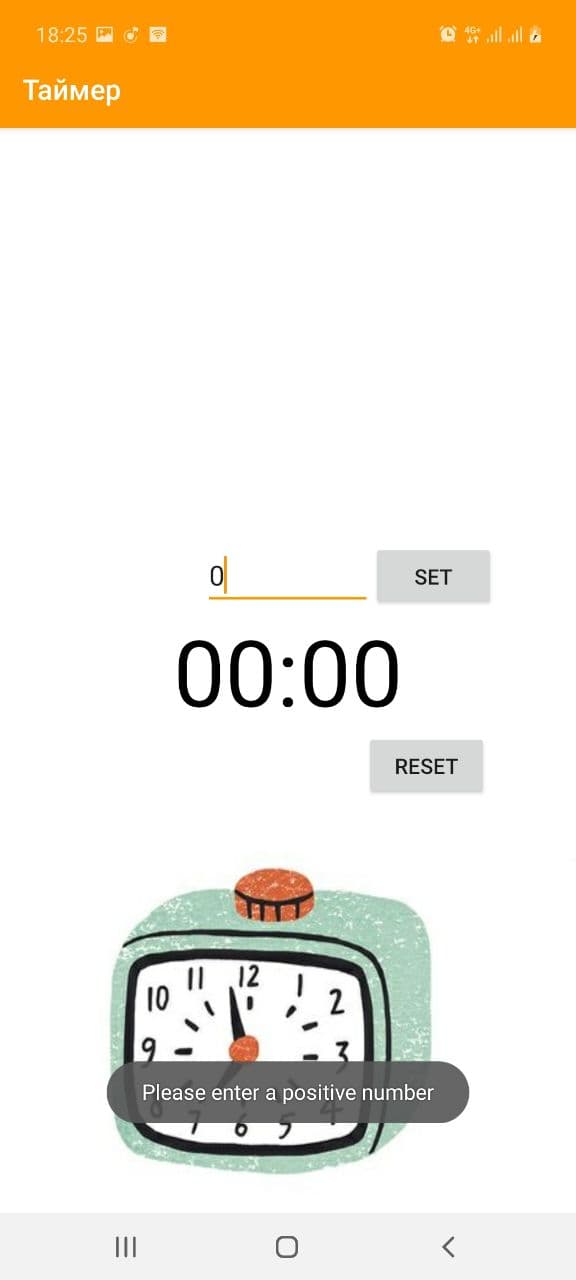


Рисунок 4.1.2 – Предупреждающее сообщение в таймере

* будет всплывать предупреждающее сообщение, если хотя бы одно из полей или все поля не заполнены при добавлении рецепта и продукта. (рис. 4.1.3)

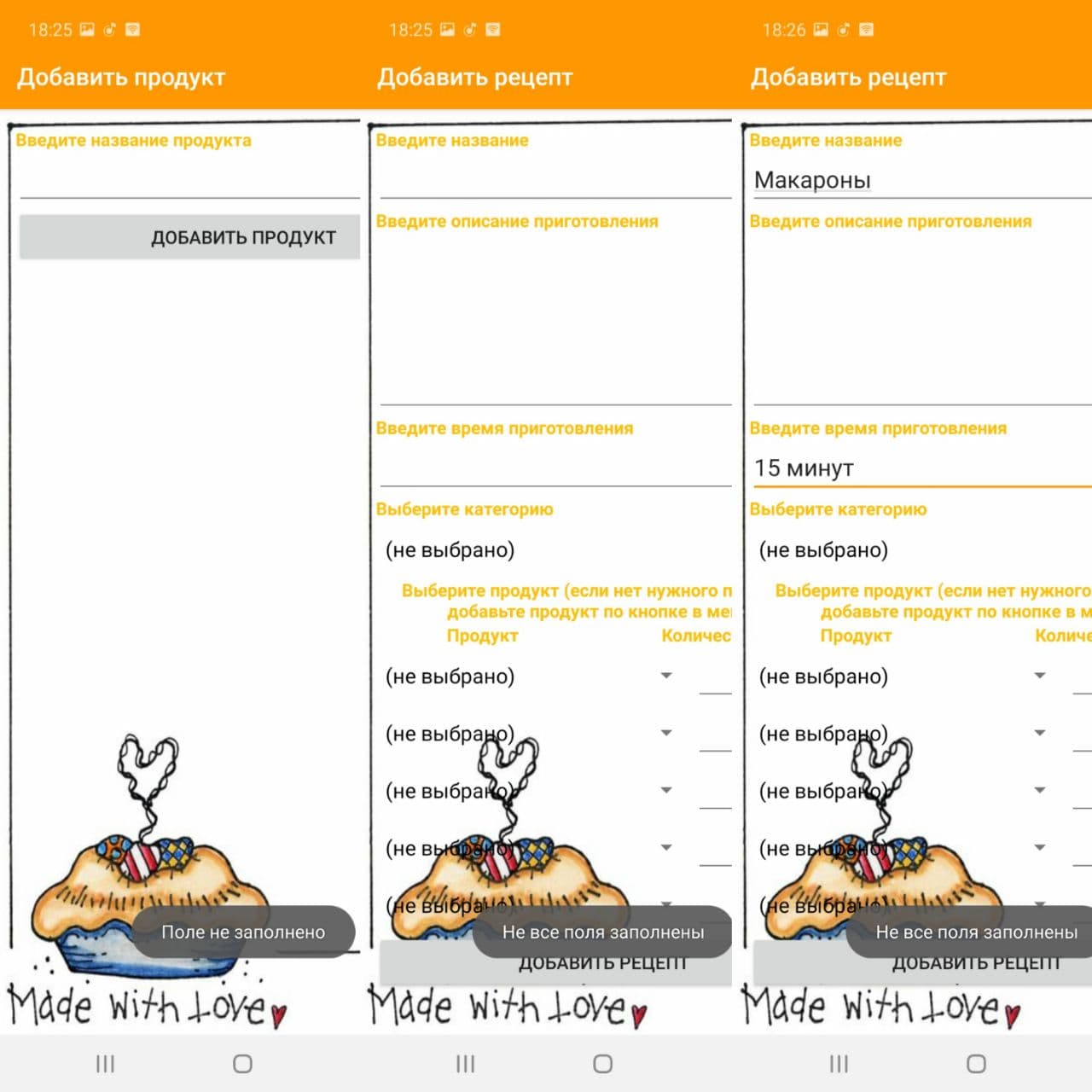


Рисунок 4.1.3 – Предупреждающие сообщения в добавлении рецепта

* появится всплывающее сообщение «Not found», если в поиске по рецептам на странице все рецепты или странице с рецептами из выбранной категории нужного рецепта не будет найдено;
* пользователь не может сделать скриншот при просмотре рецепта.

# **4.2 Выводы по главе «Тестирование разработонного продукта»**

В данной главе было рассмотрено негативное тестирование наиболее уязвимых компонентов мобильного приложения.

Были произведена проверка советующих предупреждений о том, что пользователь не сможет добавить пустые строки в таблицы базы данных при добавлении рецепта и продукта.

Также были проверены соответствующие ограничения при установке времени в таймере, при поиске рецептов на страницах с отображением всех рецептов и рецептов из определенной категории, таким образом при неверно введенных данных ошибок в приложении не будет.

# **5 Руководство пользователя**

Для того чтобы установить приложения на телефон необходимо иметь версию Android минимум 8.0 и версию API не ниже 26. Версию Android и API вашего телефона можно узнать в настройках телефона.

Если версия Android подходит, тогда можно устанавливать приложение на мобильный телефон. В меню ваших приложений или на главном экране смартфона появится новая иконка с именем «Сборник рецептов». Запустим приложение нажатием на иконку приложения. Откроется главная страница с аутентификацией по отпечатку пальца. Если ваш отпечаток пальца успешно подошел, тогда появится кнопка «Добро пожаловать» (рис. 5.1).

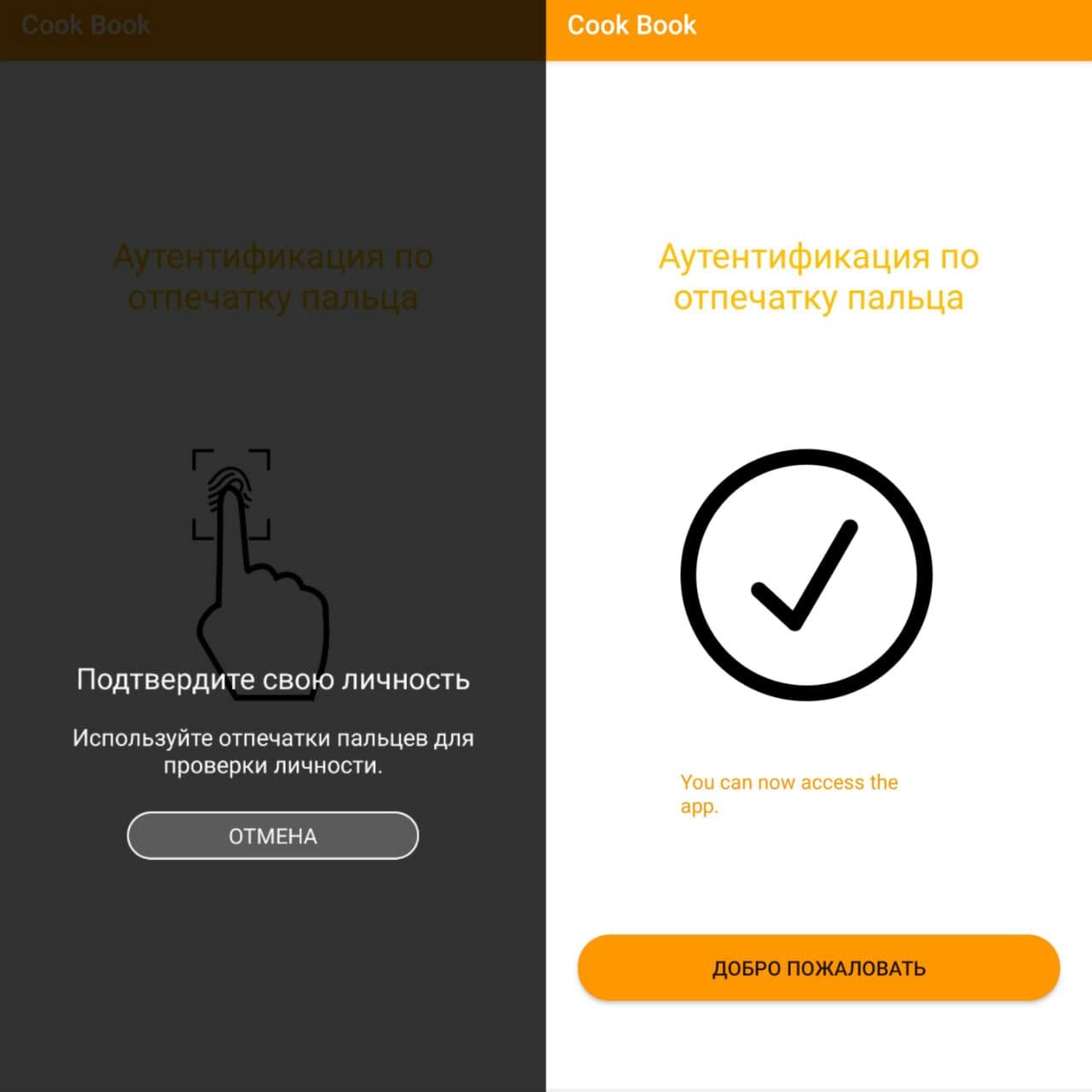


Рисунок 5.1 – Главная страница

После нажатия на кнопку «Добро пожаловать» откроется домашняя страница приложения. Для работы с приложением, вам необходимо выбрать «Синхронизация» в верхнем меню данной страницы. (рис. 5.2). После нажатия на эту кнопку появится всплывающее окно с надписью «Синхронизация».



Рисунок 5.2 – Домашняя страница

Для добавления нового рецепта необходимо нажать на кнопку «Добавить рецепт». Отобразиться страница «Добавить рецепт». Необходимо ввести название рецепта, описание приготовления и время приготовления. Также выберите категорию вашего рецепта из предложенного списка и используемые продукты в вашем рецепте, а напротив выбранных продуктов введите их количество.

Пример добавления рецепта представлен на рисунке 5.3.

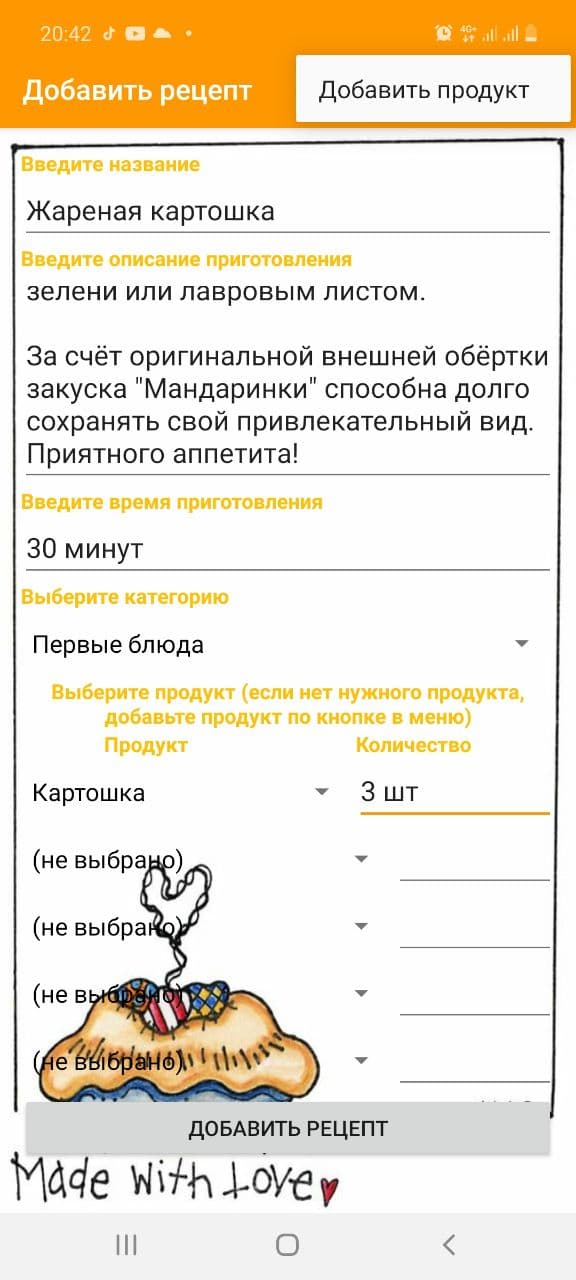


Рисунок 5.3 – Пример добавления рецепта

Если же в предложенном списке продуктов нет нужного вам, тогда необходимо выбрать «Добавить продукт» в верхнем меню данной страницы. Отобразиться страница «Добавить продукт», где в поле вам необходимо ввести название продукта и нажать на кнопку «Добавить продукт». После этого вы вернетесь на страницу «Добавить рецепт» для продолжения заполнения информации о рецепте.

Когда вся информация заполнена, нажимаем на кнопку «Добавить рецепт». Появиться всплывающее сообщение «Добавлено» и откроется главная страница приложения.

Для того чтобы посмотреть рецепты, вам необходимо на главной странице выберите кнопку «Посмотреть рецепт». Вам будет предоставлена страница «Выбор» с кнопками «Посмотреть все рецепты» и «Выбрать категорию».

Если вы выбираете «Посмотреть все рецепты» вам будет предоставлена страница «Все рецепты» со списком всех рецептов.

Для поиска рецепта необходимо нажать на иконку лупы и ввести название нужного рецепта.

При нажатии на название рецепта вам будет предоставлена страница «Рецепт» со всей информацией о выбранном рецепте, а именно, название, время приготовления, продукты, которые используются, и описание приготовление рецепта. (рис. 5.3).



Рисунок 5.3 – Страница с описанием рецепта

При необходимости изменить некоторую информацию в рецепте удерживайте пальцем название необходимого рецепта из списка до появления, всплывающего меню и нажмите на «Изменить». Появиться страница «Изменить рецепт» в которой вы можете изменить необходимые вам поля и после этого нажмите на кнопку «Изменить рецепт» (рис. 5.4). На это странице в верхнем меню, также можно добавить продукт, если в списке продуктов нет необходимого вам.

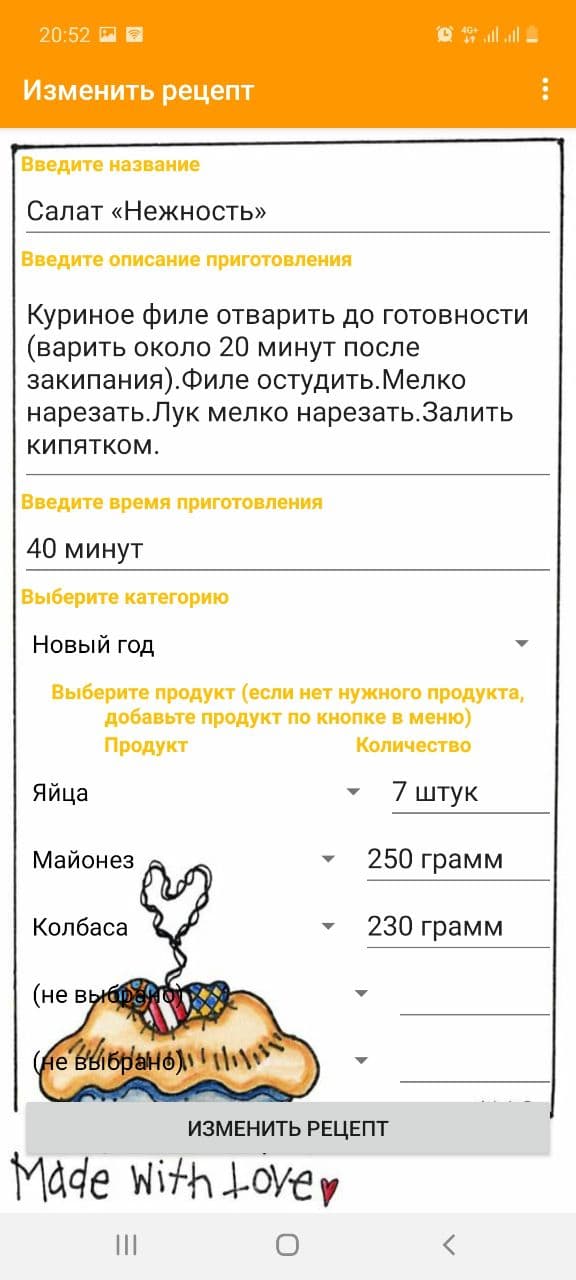


Рисунок 5.4 – Страница изменения рецепта

После этого вы вернетесь на страницу со всеми рецептами.

При необходимости удалить рецепт удерживайте пальцем название необходимого рецепта из списка до появления, всплывающего меню и нажмите на «Удалить». Появиться окно подтверждения действий «Уверены?». Нажмите на «Да», если вы уверены или на «Нет», если передумали удалять рецепт.

Когда вам необходимы рецепты определенной категории на странице «Выбор» нажмите на «Выбрать категорию». Вам будет предоставлена страница «Категории». При нажатии на необходимую вам категорию из списка, будет предоставлена страница «Рецепты из выбранной категории» (рис. 5.5).



Рисунок 5.5 – Страница с категориями

Все действия над рецептами производятся точно также, как и на странице с всеми рецептами.

Также на главной странице в верхнем меню можно выбрать «Таймер». Вам представится страница «Таймер» в которой можно установить необходимое вам время в минутах. По истечению этого времени будет играть короткая мелодия, которая свидетельствует об окончании установленного времени.

Если вам необходимы некоторые интересные факты и советы о готовке, тогда на главной странице в верхнем меню выберите «Лайфхаки для готовки». Будет отображена страница «Лайфхаки для готовки» с картинкой. Для просмотра всех картинок с советами необходимо «свайпать» вправо или влево.

Если вы добавляли, изменяли или удаляли информацию во время работы с приложением, тогда вам необходимо, повторно, на главной странице в верхнем меню выбрать «Синхронизация».

# **5.1 Выводы по главе «Руководство пользователя»**

В данной главе было описано руководство пользователя.

Данное руководство позволяет пользователю узнать требования для установки приложения на телефон, а также как использовать все функции приложения. Были предоставлены скриншоты самого приложения для большей наглядности использования.

# **ЗАКЛЮЧЕНИЕ**

Целью моего курсового проекта было экономия времени пользователей путем хранения и использования рецептов в одном и доступном месте.

Были решены следующие поставленные задачи:

* обзор аналогов;

В результате было рассмотрено 3 приложения и были выявлены их положительные и отрицательные стороны. Эта информация была использована для улучшения функционала приложения и пользовательского интерфейса.

* проектирование приложения;

Был спроектирован каждый отдельный компонент приложения. В мобильном приложении был определен основной функционал, также были разработаны необходимые диаграммы. При проектировании базы данных, были определены необходимые таблицы и разработана схема базы данных. В серверном приложении был определен архитектурный стиль взаимодействия между клиентом и сервером, формат передаваемых данных и HTTP методы для работы с серверной частью.

* реализация приложения;

Данная задача была реализована с помощью следующих технических средств: СУБД SQLite и MySQL, сред разработки Android Studio и Microsoft Visual Studio, языков программирования Java и C#, модульной платформы .NET Core, библиотек Entity Framework, SQLCipher, OkHttp, Newtonsoft.Json.

Были реализованы базы данных на стороне сервера и клиента.

При решении данной задачи была достигнута поставленная цель, и был создан проект «Сборник рецептов», который включает в себя android-клиент и сервер.

* тестирование разработанного продукта.

Для реализации данной задачи проведено негативное тестирование наиболее уязвимых компонентов мобильного приложения.

* разработка руководства пользователя.

Разработано руководство, позволяющее пользователю узнать требования для установки приложения на телефон, а также как использовать все функции мобильного приложения.

Мобильное приложение имеет простой, интуитивно понятный пользователю интерфейс и обладает следующими особенностями:

– аутентификация PrintFinger;

* добавление новых рецептов;
* просмотр всех рецептов и рецептов по категориям;
* изменение существующих рецептов;
* удаление рецепта;
* поиск необходимого рецепта по названию;
* предоставление полезной информации для готовки и таймер;
* синхронизация с сервером.

В соответствии с поставленными задачами и полученным результатом можно сделать вывод, что задачи были выполнены в полном объеме.

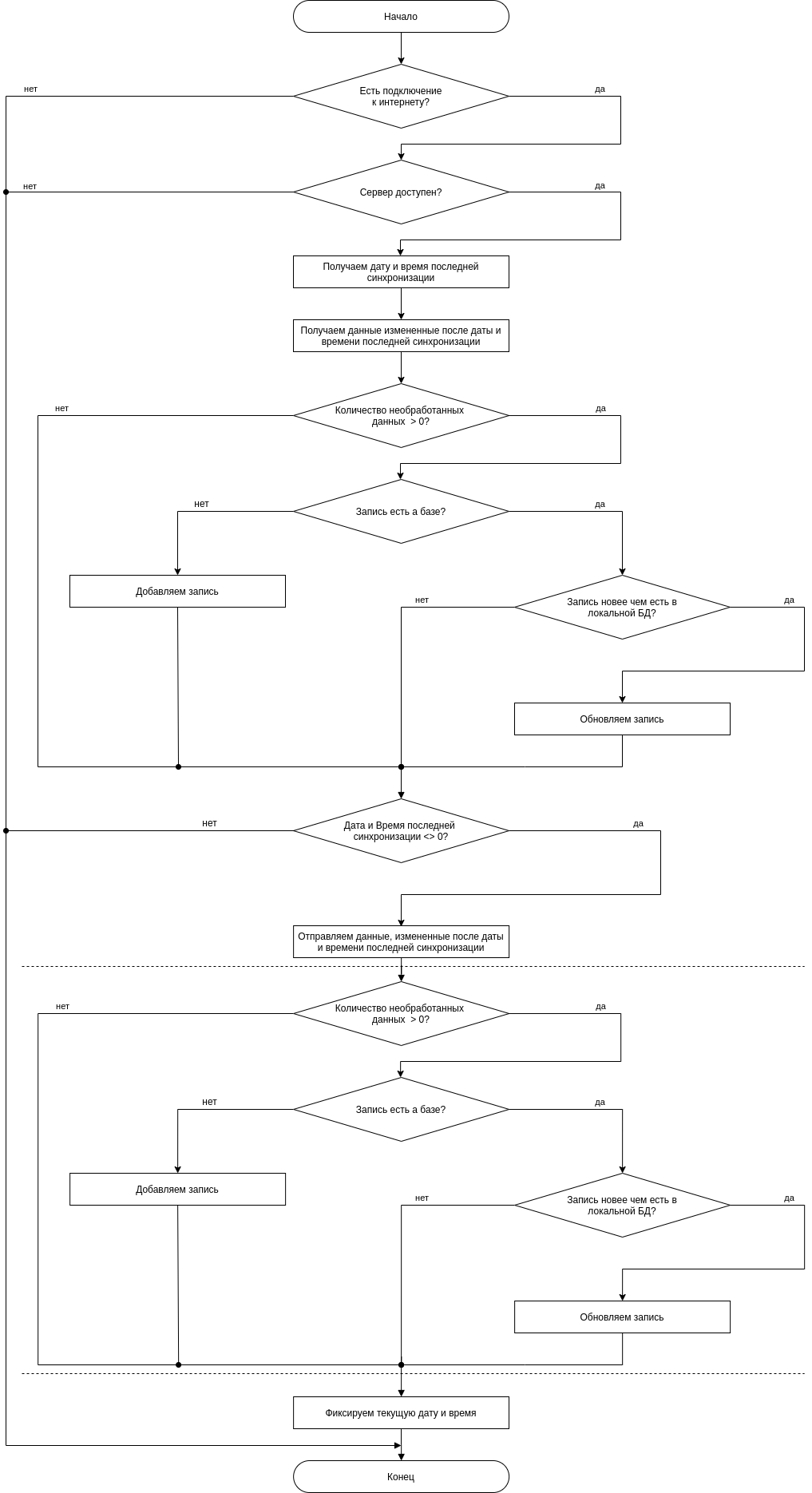
# **Список использованных источников**

1 Архитектура REST [Электронный ресурс]. / habrahabr.ru – Режим доступа: https://habrahabr.ru/post/38730 – Дата доступа: 29.04.2019.

1. Программирование под ОС Андроид [Электронный ресурс] / Metanit.com. – Режим доступа: https://metanit.com/java/android. Дата доступа: 06.04.2018
2. СountDownTimer Tutorial [Электронный ресурс] – codinginflow.com – Режим доступа: https://codinginflow.com/tutorials/android/countdowntimer/part-4-time-input. – Дата доступа: 23.02.2020
3. Руководство по ASP.NET Core 3 [Электронный ресурс] / Metanit.com. – Режим доступа: https://metanit.com/sharp/aspnet5. Дата доступа: 23.01.2020
4. Руководство по MySQL [Электронный ресурс] / Metanit.com. – Режим доступа: https://metanit.com/sql/mysql. Дата доступа: 26.05.2018

**ПРИЛОЖЕНИЕ А**

Блок схема синхронизации

****

**ПРИЛОЖЕНИЕ Б**

Код создания структуры мобильной базы данных

create table Product(

IdProduct integer primary key autoincrement,

Name integer,

DateLastChange integer,

TimeLastChange integer,

Status char(1));

create table Categorie(

IdCategorie integer primary key autoincrement,

Name integer,

DateLastChange integer,

TimeLastChange integer,

Status char(1));

create table Resipe(

IdResipe integer primary key autoincrement,

Name text,

Description text,

TimeCook text,

IdCategorie text,

DateLastChange integer,

TimeLastChange integer,

Status char(1),

foreign key(IdCategorie) references Categorie(IdCategorie));

create table Product\_in\_Resipe(

IdProduct integer,

IdResipe integer,

IdProductInResipe integer primary key autoincrement,

Quantity text,

DateLastChange integer,

TimeLastChange integer,

Status char(1),

foreign key(IdProduct) references Product(IdProduct),

foreign key(IdResipe) references Resipe(IdResipe));

create TRIGGER afterSetRemovedStatus\_Remove

after UPDATE on Resipe

when new.Status = 'R'

BEGIN

update Product\_in\_Resipe set Status = 'R',

DateLastChange = cast(substr(strftime('%Y','now'),3,2) || strftime('%m%d','now') as INTEGER),

TimeLastChange = cast(strftime('%H%M%S',time('now','+3 hour')) as INTEGER)

where IdResipe = new.IdResipe;

end;

**ПРИЛОЖЕНИЕ В**

Код создания структуры серверной базы данных

use cookbook\_db;

CREATE TABLE Categorie (

IdCategorie int NOT NULL,

Name nvarchar(50),

DateLastChange int NOT NULL,

TimeLastChange int NOT NULL,

Status char(1),

PRIMARY KEY (IdCategorie));

CREATE TABLE Product (

IdProduct int NOT NULL,

Name nvarchar(50),

DateLastChange int NOT NULL,

TimeLastChange int NOT NULL,

Status char(1),

PRIMARY KEY (IdProduct));

CREATE TABLE Resipe (

IdResipe int NOT NULL,

IdCategorie int,

Name nvarchar(50),

Description text,

TimeCook nvarchar(50),

DateLastChange int NOT NULL,

TimeLastChange int NOT NULL,

Status char(1),

PRIMARY KEY (IdResipe),

FOREIGN KEY(IdCategorie) REFERENCES Categorie(IdCategorie));

CREATE TABLE Product\_in\_Resipe (

IdProductInResipe int NOT NULL,

IdProduct int,

IdResipe int,

Quantity nvarchar(50),

DateLastChange int NOT NULL,

TimeLastChange int NOT NULL,

Status char(1),

PRIMARY KEY (IdProductInResipe),

FOREIGN KEY(IdProduct) REFERENCES Product(IdProduct),

FOREIGN KEY(IdResipe) REFERENCES Resipe(IdResipe)

);

**ПРИЛОЖЕНИЕ Г**

Код ResipeActivity.java

**import** androidx.appcompat.app.AppCompatActivity;  
**import** android.database.Cursor;  
**import** net.sqlcipher.database.SQLiteDatabase;  
**import** android.os.Bundle;  
**import** android.widget.TextView;  
**import** by.bstu.fit.alexsandrova.projectbd.Help.UriBuilder;  
**import** by.bstu.fit.alexsandrova.projectbd.Database.DbHelper;  
**import** by.bstu.fit.alexsandrova.projectbd.Models.Resipe;  
**import** by.bstu.fit.alexsandrova.projectbd.R;  
  
**public class** ResipeActivity **extends** AppCompatActivity {  
 TextView **nameResipeTextView**;  
 TextView **descriptionTextView**;  
 TextView **timeCookTextView**;  
 TextView **productTextView**;  
 Resipe **resipe**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_resipe***);  
 *//инициализируем TextView* **nameResipeTextView** = findViewById(R.id.***nameResipeTextView***);  
 **descriptionTextView** = findViewById(R.id.***descriptionTextView***);  
 **timeCookTextView** = findViewById(R.id.***timeCookTextView***);  
 **productTextView** = findViewById(R.id.***productTextView***);  
 setTitle(**"Рецепт"**);  
 *//получаем id из списка на странице, где был выбран рецепт* String id = String.*valueOf*(getIntent().getIntExtra(**"id"**, 0));  
 *//получаем информацию о рецепте используя метод query и id рецепта* Cursor cursor = getContentResolver().query(UriBuilder.*getUri*(UriBuilder.***RESIPE***, id), **null**, **null**, **null**);  
 **if** (cursor.moveToFirst()) {  
 **resipe** = **new** Resipe(cursor);  
 }  
 *//устанавливаем полученные значения в TextView* **nameResipeTextView**.setText(**resipe**.getName());  
 **descriptionTextView**.setText(**resipe**.getDescription());  
 **timeCookTextView**.setText(**resipe**.getTimeCook());  
 cursor.close();  
 *//создаем запрос для получения продуктов в рецепте и их количества* cursor = **new** DbHelper(**this**).getReadableDatabase().rawQuery(**"select t2.name, t1.Quantity from Product\_in\_Resipe t1\n"** +  
 **"inner join Product t2\n"** +  
 **"on t1.IdProduct = t2.IdProduct\n"** +  
 **"where t1.IdResipe = ?"**, **new** String[]{id});  
 StringBuilder sb = **new** StringBuilder();  
 **if** (cursor.moveToFirst()) {  
 **do**{  
 sb.append(cursor.getString(0) + **" - "** + cursor.getString(1) + **"\n"**);  
 }  
 **while** (cursor.moveToNext());}  
 cursor.close();  
 *//устанавливаем полученные значения в TextView* **productTextView**.setText(sb.toString());  
 }}

**ПРИЛОЖЕНИЕ Д**

Код CategorieController.cs

using System.Collections.Generic;

using System.Linq;

using CookBookAPI.Interfaces;

using CookBookAPI.Models;

using Microsoft.AspNetCore.Mvc;

using Newtonsoft.Json;

namespace CookBookAPI.Controllers

{

[ApiController]

[Route("[controller]")]

public class CategorieController : ControllerBase, IControllerBase<Categorie>

{

private readonly CookbookDbContext \_db;

public CategorieController(CookbookDbContext db)

{

\_db = db;

}

[HttpGet]

public string Update(int date, int time)

{

// получаем данные, что больше указанной даты

var list1 = \_db.Categorie.Where(x => x.DateLastChange > date);

// получаем данные за указанную дату, но больше указанного времени

var list2 = \_db.Categorie.Where(x => x.DateLastChange == date && x.TimeLastChange > time);

//сериализуем список объектов в JSON

Return JsonConvert.SerializeObject(list2.Union(list1).ToList(),Formatting.Indented);}

[HttpPost]

public void Upgrade(List<Categorie> list)

{

foreach (var item in list)

{

//поучаем объект

List<Categorie> listOfCategories = \_db.Categorie.Where(x => x.IdCategorie == item.IdCategorie).Select(x => x).ToList();

// если не нашли данные в БД значит добавляем новую запись с пришедшими данными

if (listOfCategories.Count == 0){

AddNewRecord(item);

continue;}

Categorie categorie = listOfCategories[0];

//проверка даты и времени, чтобы пришедший объект был более новым

if(categorie.DateLastChange < item.DateLastChange ||

(categorie.DateLastChange == item.DateLastChange && categorie.TimeLastChange < item.TimeLastChange)){

//обновление записей

categorie.IdCategorie = item.IdCategorie;

categorie.Name = item.Name;

categorie.DateLastChange = item.DateLastChange;

categorie.TimeLastChange = item.TimeLastChange;

if (item.Status != null)

categorie.Status = "R";

//вызываем метод обновления и сохранения

\_db.Categorie.Update(categorie);

\_db.SaveChanges();}}}

public void AddNewRecord(Categorie newRecord){

//добавить и сохранить

\_db.Categorie.Add(newRecord);

\_db.SaveChanges();}}}