МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования

**«Южно-Уральский государственный университет»**

**(национальный исследовательский университет)**

**Высшая школа электроники и компьютерных наук**

**Кафедра системного программирования**

|  |  |
| --- | --- |
| РАБОТА ПРОВЕРЕНА  Рецензент  <ученая степень, ученое звание>  \_\_\_\_\_\_\_\_\_\_ <И.О. Фамилия рецензента>  “\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 2018 г. | ДОПУСТИТЬ К ЗАЩИТЕ  Заведующий кафедрой, д.ф.-м.н., профессор  \_\_\_\_\_\_\_\_\_\_ Л.Б. Соколинский  “\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 2018 г. |

**Разработка мобильной кулинарной книги для ОС iOS с функцией подбора сочетаемых ингредиентов**

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ЮУрГУ – 02.03.02.2018.115-007.ВКР

|  |  |
| --- | --- |
|  | Научный руководитель  к.ф.-м.н., доцент  \_\_\_\_\_\_\_\_\_\_Г.И. Радченко    Автор работы, студент группы КЭ-401  \_\_\_\_\_\_\_\_\_\_Г.В. Волкова  Ученый секретарь  (нормоконтролер)  \_\_\_\_\_\_\_\_\_\_\_\_\_ О.Н. Иванова  “\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 2018 г. |

Челябинск-2018

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 3](#_Toc514278952)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И СУЩЕСТВУЮЩИХ РАБОТ ПО ТЕМАТИКЕ ДИПЛОМА 5](#_Toc514278953)

[1.1. Проектирование UI мобильных приложений 5](#_Toc514278954)

[1.2. Анализ схожих проектов 6](#_Toc514278955)

[1.3. Анализ подходов к реализации мобильных приложений 8](#_Toc514278956)

[2. ТРЕБОВАНИЯ К СИСТЕМЕ 10](#_Toc514278957)

[2.1. Функциональные требования к системе 10](#_Toc514278958)

[2.2. Нефункциональные требования к системе 11](#_Toc514278959)

[2.3. Варианты использования системы 11](#_Toc514278960)

[3.1. Компоненты системы 13](#_Toc514278961)

[3.2. Структура базы данных 15](#_Toc514278962)

[3.3. Диаграмма деятельности 16](#_Toc514278963)

[4.1. Наполнение базы данных 20](#_Toc514278964)

[4.2. Реализация моделей 22](#_Toc514278965)

[4.3. Реализация контроллеров 23](#_Toc514278966)

[4.4. Реализация отображений 24](#_Toc514278967)

[ЛИТЕРАТУРА 30](#_Toc514278968)

# 

# ВВЕДЕНИЕ

**Актуальность темы**

Мобильные приложения входят в жизнь людей с огромной скоростью. По данным [1] за 2016 год, в магазине приложений App Store располагалось более 2 миллионов приложений. Все они относятся к различным категориям: игры, здоровье, музыка, кулинария, образование и пр. Мобильные приложения облегчают повседневные задачи, помогают в их выполнении, разнообразят рутинные дела, вовлекают в процесс выполнения, казалось бы совершенно обычных и даже скучных дел, и даже могут помочь улучшить некоторые навыки и способности.

В наше время, практически у каждого человека есть смартфон, который всегда под рукой. Люди стали быстрее получать различную информацию, и сами данные стали доступнее с появлением смартфонов. По статистике, человек, пользующийся смартфоном, в среднем проверяет свой телефон раз в 6,5 минут [2]. Поэтому, актуальность темы мобильных приложений достаточно высока.

Данная работа направлена на создание мобильного приложения, которое облегчит такую повседневную обязанность, как приготовление еды. Отличительной особенностью системы будет подбор сочетающихся ингредиентов к выбранному пользователем продукту, для улучшения вкусовых качеств повседневных блюд.

**Цели и задачи работы**

Целью данной работы является разработка мобильной кулинарной книги для ОС iOS с функцией подбора сочетаемых ингредиентов. Для достижения цели работы, необходимо решить следующие задачи:

* Анализ литературы и смежных проектов, связанных с мобильной кулинарией;
* Определение требований к мобильному приложению для мобильной кулинарной книги;
* Разработка архитектуры мобильного приложения для мобильной кулинарной книги;
* Разработка схемы взаимодействия пользователя с интерфейсом приложения;
* Создание базы данных с сочетаемыми ингредиентами для мобильной кулинарной книги;
* Разработка архитектуры базы данных для хранения данных, вводимых пользователем.

**Объем и структура работы**

**Краткое содержание работы**

## АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И СУЩЕСТВУЮЩИХ РАБОТ ПО ТЕМАТИКЕ ДИПЛОМА

## Проектирование UI мобильных приложений

Разработка мобильных приложений на сегодняшний день – долгий и трудоемкий процесс. Чтобы приложение было успешным на рынке, необходимо учесть и проработать многие аспекты, такие как дизайн, функционал, хранение и обработка данных, база данных.

Важной частью в создании мобильных приложений является создание удобного к применению интерфейса. В статье [3] представлены основные проблемы проектирования мобильных приложений, такие как:

* разные размеры экранов мобильных телефонов и других похожих устройств;
* трудности ввода данных в мобильное приложение;
* легкость в использовании мобильного приложения;
* безопасность хранения данных.

Так, опираясь на статью [3], для создания мобильного приложения необходимо разработать простой в использовании и приятный для пользователя интерфейс.

В книге В. Головача описаны основные аспекты, по которым разрабатывается пользовательский интерфейс [8]. Согласно его книге, интерфейс должен быть интуитивно понятным, не броским, и должен быть скорее незаметным, чем ярко бросаться в глаза пользователю. «Любая красота со временем надоедает и в лучшем случае перестает восприниматься. Именно поэтому в интерфейсах обычно не место красоте» - говорит В. Головач.

Слова В. Головача также подтверждает другой дизайнер пользовательского интерфейса – Г. Кришна [9]. В своей книге «Хороший интерфейс – невидимый интерфейс» автор также хочет донести до читателей, что интерфейс любой вещи, в том числе и мобильного приложения, должен быть как бы невидимым, ненавязчивым, понятным. Ко всему можно добавить интерфейс, даже к холодильнику. Однако, «Лучший интерфейс - невидимый интерфейс», вот что на протяжении всей книги доносит нам автор.

## Анализ схожих проектов

Для получения и хранения рецептов различных блюд существует множество мобильных приложений и вeб-сервисов. Но не все они выполняют функцию подбора сочетаемых ингредиентов. Рассмотрим функционал некоторых из них, доступных на территории РФ (рис. 1). Приложение «Eat this much» (рис. 1-а) обеспечивает пользователю следующие возможности:

* Подбор персонального рациона на неделю;
* Рецепты подобранных блюд;
* Подсчет калорий и БЖУ блюд;
* Добавление продуктов в корзину.

В приложении отсутствует функция подбора сочетаемых ингредиентов, также оно доступно только на английском языке. Приложение «Твои рецепты» (рис. 1-б) реализует следующий функционал:

* Предоставление информации о рецептах по категориям, коллекциям или авторам;
* Поиск рецептов по категориям;
* Добавление понравившихся рецептов во вкладку «Избранное»;
* Добавление ингредиентов в корзину.
* Приложение не реализует функцию подбора сочетаемых ингредиентов. Также, многие рецепты доступны только в платной версии приложения.

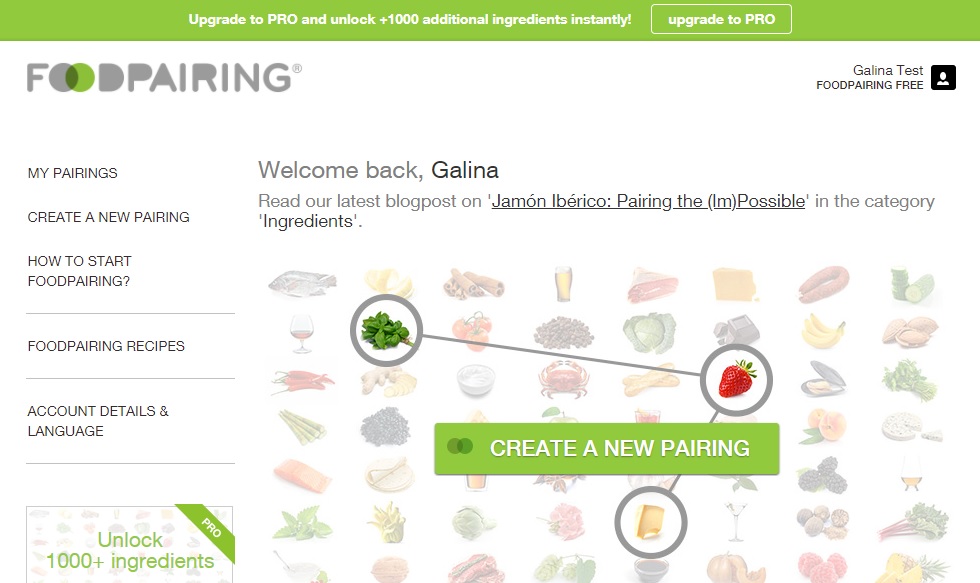
Приложение «Что с чем» (рис. 1-в) реализует функцию подбора сочетаемых ингредиентов. Однако, продукты объединены в категории (например, нельзя отдельно найти ингредиенты для свинины, только для мяса). Соответственно, подбор ингредиентов происходит так же по категориям, а не по отдельным продуктам.

|  |  |  |
| --- | --- | --- |
| C:\Users\Иван\Desktop\Диплом\diplom\скрины\eat this much.jpg  а) | б) | в) |

**Рис. 1.** Скриншоты мобильных приложений по кулинарии:

а) Eat this much; б) Твои рецепты; в) Что с чем

Отдельного внимания заслуживает вeб-приложение «FoodPairing» (рис. 2). При помощи данного сервиса, пользователь может самостоятельно подобрать продукты разной степени сочетаемости. Количество выбранных продуктов не ограничивается, и можно подобрать сочетания как с одним, так и с несколькими продуктами. Сервис предоставляется по премиум модели: основной функционал и некоторые продукты доступны бесплатно, дополнительные функции и продукты – по подписке. Также, в данном приложении нет поддержки русского языка.



**Рис. 2.** Вeб-приложение «FoodPairing»

## Анализ подходов к реализации мобильных приложений

Для реализации приложения можно выделить два типа решений:

1. Кроссплатформенные – средства, которые позволяют вести разработку приложения под несколько различных платформ;
2. Нативные – средства, предоставляемые компанией-разработчиком платформы.

Так как в рамках данной работы разрабатывается мобильное приложения для операционной системы iOS, рассмотрим инструментарий для разработки, предоставленный компанией Apple – iOS SDK. iOS SDK [4] – это пакет средств для разработки, которые могут понадобиться для создания мобильного приложения на смартфоны или планшеты компании Apple. Основными средствами для создания проекта будут следующие компоненты:

1. Xcode – интегрированная среда разработки со встроенным эмулятором iPhone;
2. Interface Builder [5] – компонент Xcode, необходимый для создания интерфейса программ и связей между событиями интерфейса и кодом программы.

Для создания мобильных приложений существует множество шаблонных решений – паттернов. Однако, наиболее распространенным является такой паттерн проектирования, как MVC (Model View Controller) [6]. MVC классифицирует объекты в приложении по ролям. Существует три вида ролей:

1. Модель (model) содержит данные приложения и определяет механизмы манипуляций над ними;
2. Представление (view) отвечает за визуальное представление модели, а также элементов управления, с которыми пользователь может взаимодействовать;
3. Контроллер (controller) управляет всей работой. Он имеет до-ступ к данным модели, отображает эти данные в представлениях, подписывается на события и манипулирует данными

Локальное хранилище данных [13] возможно организовать на основе таких программных платформ, как:

1. FMDB, предоставляющий хранение на основе sqlite;
2. Realm – хранение при помощи данного фреймворка организовано на основе собственной кроссплатформенной мобильной базы данных;
3. Core data – интегрированная платформа для работы с данными.

**Вывод**

Анализ существующих аналогов по теме работы показывает, что качественного и доступного мобильного приложения по подбору сочетаний продуктов на данный момент нет. Это говорит о том, что решение данной задачи является возможным и актуальным.

## ТРЕБОВАНИЯ К СИСТЕМЕ

Проект Appetizer будет представлять собой мобильное приложение, помогающее пользователю в подборе сочетаемых ингредиентов (продуктов) для выбранной категории блюд, создании и хранении собственных рецептов, формировании списка покупок. С помощью данного мобильного приложения, пользователь сможет быстро подбирать ингредиенты (продукты) для выбранного блюда, записывать и хранить понравившиеся рецепты в одном месте и иметь к ним быстрый доступ. Исходя из выбранной пользователем категории пищи, приложение будет выводить на экран ингредиенты, сочетаемые с данным блюдом, которые будут храниться в базе данных продуктов. Система должна облегчить пользователю выбор между различными специями и значительно улучшить вкусовые качества готового продукта. Также, в системе есть возможность создать список покупок, в который можно добавлять как собственные пункты, так и предложенные системой ингредиенты.

Для использования мобильного приложения пользователю понадобится смартфон с операционной системой iOS. Доступ к интернету для использования приложения не требуется.

## Функциональные требования к системе

Можно определить следующий набор функциональных требований к системе:

* 1. Система Appetizer должна отображать все имеющиеся продукты, сочетающиеся с выбранным пользователем продуктом.
  2. Система Appetizer должна позволять пользователю сохранять подобранные сочетания продуктов и просматривать их.
  3. Система Appetizer должна позволять пользователю создавать списки покупок с возможностью добавлять туда как свои пункты, так и элементы, предлагаемые системой при подборе сочетаемых продуктов.

## Нефункциональные требования к системе

Выделяются следующие нефункциональные требования к системе:

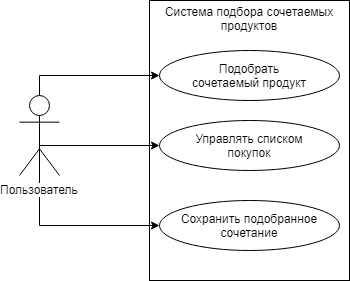
1. Система Appetizer должна быть написана на языке Swift.
2. Система Appetizer должна быть доступна на смартфонах с операционной системой iOS v 10.0 и выше.
3. Система Appetizer должна брать данные о сочетаемых продуктах из готовой базы данных.
4. Хранение данных системы Appetizer должно быть реализовано при помощи Realm.

## Варианты использования системы

Мы выделяем одного актера, взаимодействующего с системой (см. рис. 3).

Пользователь – это человек, взаимодействующий с приложением Appetizer, который может управлять вводом новых сочетаний продуктов, просматривать все имеющиеся данные в приложении, управлять списком покупок.

Пользователь может реализовать следующие варианты использования системы:



**Рис.3.** Диаграмма вариантов использования

1. Пользователь может выбрать продукт, предложенный из готового списка, и система предложит ему сочетаемые продукты.

2. Пользователь может создавать и управлять списками покупок. Пользователь может добавить собственные элементы в список, а может добавить продукты из подобранного программой списка сочетаний.

3. Пользователь может сохранять подобранные им сочетания в системе.

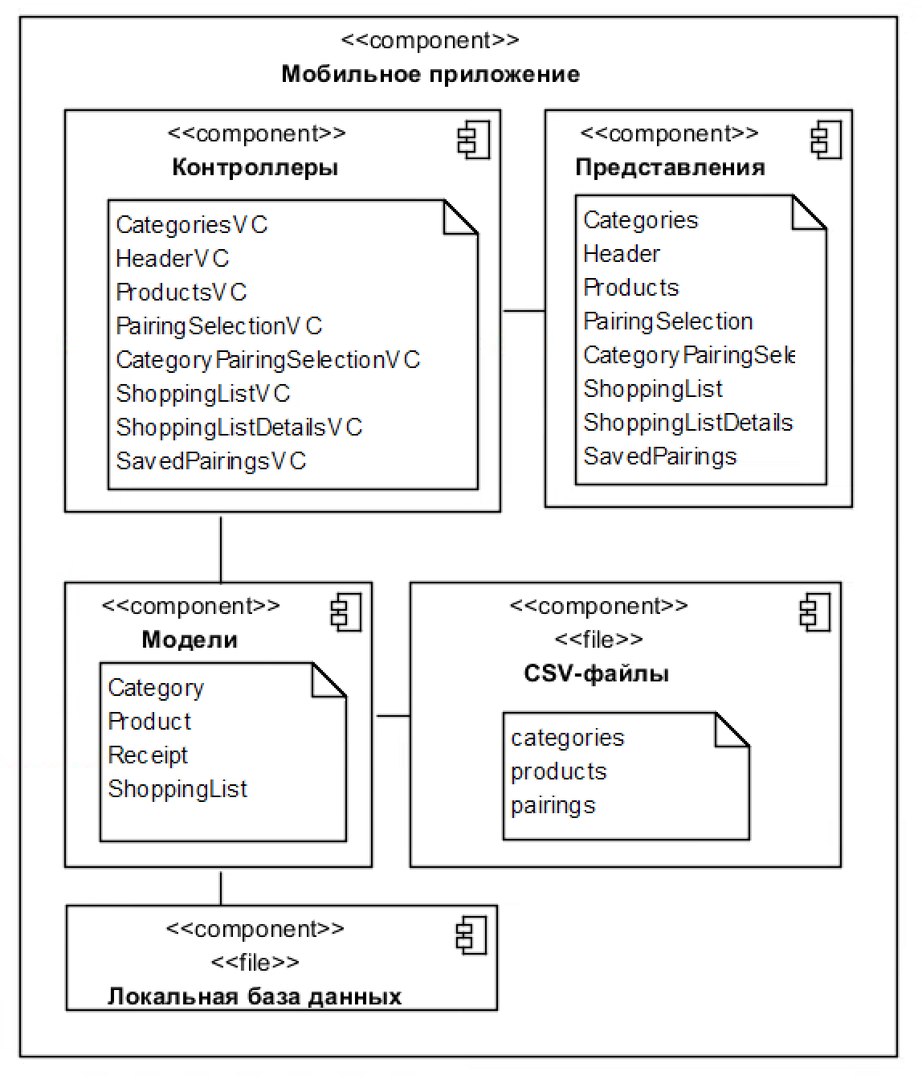
1. **АРХИТЕКТУРА СИСТЕМЫ**

Архитектура мобильного приложения Appetizer построена на основе паттерна проектирования MVC. Используя данный паттерн, мы разбиваем архитектуру мобильного приложения на три компонента: модель, отображение и контроллер. Они взаимодействуют друг с другом посредством интерфейсов, так что каждый компонент можно изменять независимо друг от друга.

Основываясь на книгу [10] мной были разработаны диаграммы UML, описанные в данной главе.

## Компоненты системы

Мобильное приложение Appetizer состоит из следующих компонентов (рис. 4):

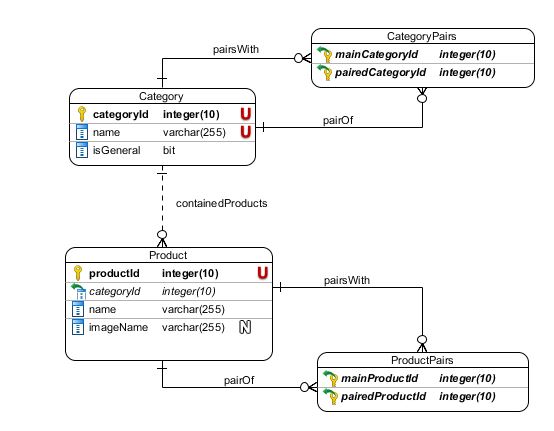


**Рис. 4.** Диаграмма компонентов системы

1. Локальная база данных и таблицы в формате CSV, хранящие в себе информацию о продуктах и их сочетаемости(о базе данных подробнее в пунктах 3.2 и 4.1);
2. Контроллеры и представления:
   1. Экран выбора категории главного продукта (*CategoriesVS* и *Categories*), который отвечает за отображение пользователю категорий продуктов, для которых возможно подобрать сочетания;
   2. Экран выбора главного продукта (*ProductsVS* и *Products*), который отвечает за отображение пользователю продуктов, для которых возможно подобрать сочетания;
   3. Экран подбора сочетаемых продуктов (*PairingSelectionVC* и *PairingSelection*), на котором пользователю предлагаются разделенные по категориям продукты доступные для сочетания с выбранным продуктом;
   4. Экран вывода подобранных сочетаний (*CategoriyPairingSelectionVC* и *CategoryPairingSelection*), на котором отображаются все выбранные пользователем подобранные продукты. На этом экране также пользователь может сохранить подобранное сочетание;
   5. Экран списков покупок (*ShoppingListVC* и *ShoppingList*), который отвечает за отображения списков с продуктами, добавленных пользователем;
   6. Экран списка продуктов (*ShoppingListDetailsVC* и *ShoppingListDetails*), который отвечает за отображение содержания списка покупок;
   7. Экран сохраненных подобранных сочетаний (*SavedPairingsVC* и *SavedPairingsVC*), который отображает сохраненные сочетания продуктов, выбранные пользователем;
3. Модели, реализующие логику связи контроллеров с базой данных:
   1. *Category* – модель, позволяющая управлять объектами категорий и их параметрами;
   2. *Product* – модель, позволяющая управлять объектамипродуктов и их параметрами;
   3. *Receipt* – модель, позволяющая управлять подобранными сочетаниями продуктов и их параметрами;
   4. *ShоppingList* – модель, позволяющая управлять объектами списков продуктов и их параметрами;

## Структура базы данных

Для работы системы была разработана схема базы данных (см. рис 4), содержащая информацию о продуктах и об их сочетаемости между собой.



**Рис. 5.** Диаграмма отношений сущностей базы данных

Рассмотрим подробнее таблицы, представленные в базе данных:

**Category** – категория, к которой принадлежит продукт. Имеет следующие поля:

* *CategoryID* – уникальный идентификатор категории. Является первичным ключом;
* *Name* – имя категории;
* *IsGeneral* – поле, в котором отмечается, является ли выбранная категория продукта главной (то есть, к ней ли мы ищем сочетания).

**Product –** таблица, хранящая в себе информацию о продукте. Здесь присутствуют следующие поля:

* *ProductID* – уникальный идентификатор продукта. Является первичным ключом;
* *CategoryID* – уникальный идентификатор категории, к которой принадлежит продукт. Значение берется из таблицы Category;
* *Name* – имя продукта;
* *ImageName –* название иконки, привязанной к продукту.

**CategoryPairs** – категория, сочетаемая с главной категорией, к кото-рой привязан выбранный продукт. У данной таблицы есть такие поля, как:

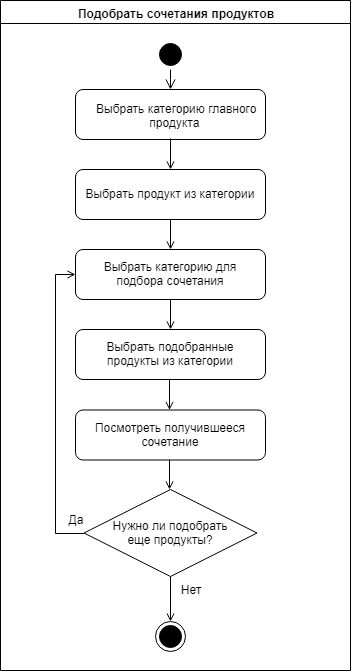
* *MainCategoryID* – уникальный идентификатор категории выбранного продукта. Значение берется из таблицы Category;
* *PairedCategoryId* – уникальный идентификатор категории, которая сочетается с категорией выбранного продукта. Значение берется из табли-цы Category.

**ProductPairs** **–** таблица, хранящая информацию о сочетаемых продуктах. В таблице есть следующие поля:

* *MainProductID* – уникальный идентификатор продукта. Значение берется из таблицы Product.
* *PairedProductID* – уникальный идентификатор продукта. Значение берется из таблицы Product.

## Диаграмма деятельности

На рис. 6 представлена диаграмма деятельности по прецеденту «Подобрать сочетания продуктов».



**Рис. 6.** Диаграмма деятельности прецедента «Подобрать сочетания продуктов»

Описание реализации прецедента «Подобрать сочетания продуктов»:

1. Пользователь выбирает категорию продукта, для которого он хочет подобрать сочетание;
2. Далее, система переводит пользователя на экран продуктов выбранной категории. Из предложенного списка, пользователь выбирает свой главный продукт;
3. Открывается окно, в котором нужно выбрать категорию продуктов для подбора сочетания. После выбора нужной категории, пользователь переходит к следующему шагу;
4. Пользователь выбирает понравившийся продукт из предложенного списка;
5. Открывается окно с выбранным сочетанием;
6. Если пользователю нужно подобрать еще сочетания, система переводи пользователя на шаг 3, иначе – работа с данным прецедентом закончена.



**Рис. 7.** Диаграмма деятельности прецедента «Добавить продукты в список покупок»

На рис. 7 представлена диаграмма деятельности по прецеденту «Добавить продукты в список покупок».

1. Описание реализации прецедента «Добавить продукты в список покупок»: На экране подобранного сочетания пользователь нажимает на кнопку «Добавить в список покупок»;
2. Если пользователю нужно добавить все продукты из подобранного сочетания в список покупок, то работа с прецедентом заканчивается;
3. Иначе, пользователь выбирает необходимые продукты из списка.
4. **РЕАЛИЗАЦИЯ СИСТЕМЫ**

## Наполнение базы данных

Отдельного внимания в разработке системы Appetizer заслуживает создание базы данных сочетаемых ингредиентов. Для решения данной задачи существует несколько путей:

1. Интеграция существующих баз данных о сочетаниях;
2. Автоматическое наполнение базы данных с помощью обхода и парсинга кулинарных сайтов;
3. Создание базы данных вручную с помощью статей, сайтов и книг.

Лучшим решением была бы интеграция существующей базы данных, однако, такой базы в открытом доступе не существует. Поэтому, этот способ не подходит для решения поставленной задачи.

Автоматическое наполнение базы данных на основе кулинарных статей и книг достаточно быстрый способ создания базы данных, однако он не является надежным. В базе могут появляться ненужные объекты, они могут дублироваться и создавать проблемы при использовании приложения.

Наполнение базы данных вручную – самый долгий и трудозатратный процесс, однако при этом варианте наполнение будет наиболее эффективным и достоверным, так как будет контролироваться непосредственно разработчиком. Поэтому, в рамках данной работы, было принято решение о создании базы данных вручную, используя различные кулинарные сайты, книги, статьи.

Мною было решено сделать базу данных в формате CSV. Внутри приложения таблицы созданной базы данных обрабатываются в соответствии со схемой базы данных, описанной в главе 3.2. В базе данных присутствуют три таблицы – Categories, Products, Pairs. Таблица Categories соответствует таблицам Category и CategoryPairs, описанных в главе 3.2. Она отвечает за хранение данных о категориях продуктов. В данной таблице содержится 9 строк с записями. Таблица создана для разделения продуктов по их категориям и удобной работы с данными. Пример заполнения таблицы Categories отображен в табл. 1.

**Табл. 1.** Пример заполнения таблицы Categories

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Id | isGeneral | pairedCategories |
| Мясо | 1 | True | 3,4,5,6,8,9 |
| Рыба | 2 | True | 3,4,5,6,8,9 |
| Гарнир | 3 | True | 1,2,4,5,6,7,8,9 |
| Специи | 4 | False | – |
| Десерт | 5 | False | – |

Таблица Products соответствует таблице Product, описанной в главе 3.2. Она хранит в себе данные о продуктах, которые используются в системе Appetizer. В таблице присутствуют 56 записей. Пример заполнения отображен в табл. 2.

**Табл. 2.** Пример заполнения таблицы Products

|  |  |  |
| --- | --- | --- |
| Name | Id | CategoryId |
| Свинина | 1 | 1 |
| Карп | 6 | 2 |
| Базилик | 19 | 4 |
| Тирамису | 32 | 5 |
| Чай | 38 | 6 |

Таблица Pairs хранит информацию о сочетании продуктов. Эта таблица соответствует таблице ProductPairs, описанной в главе 3.2. Эта таблица является основной для работы системы Appetizer. Для ее создания, я просмотрела различные статьи и кулинарные рецепты, выделив основные продукты и их сочетаемость [11] [12]. В таблице Pairs хранятся идентификатор главного продукта (к которому подбирается сочетание) и массив из идентификаторов продуктов, которые сочетаются с главным продуктом. В таблице 28 строк записей. Пример заполнения таблицы Pairs приведен в табл. 3.

**Табл. 3.** Пример заполнения таблицы Pairs

|  |  |
| --- | --- |
| MainId | PairedId |
| 1 | 10,11,12,13,14,15,16,22,23,24,25,35,37,38,39,48,49,50,51 |
| 2 | 10,11,12,13,18,19,21,23,33,34,38,39,44,49,51,52 |
| 3 | 11,12,13,15,16,18,20,21,27,33,35,36,38,40,41,48,49,50,51 |
| 4 | 11,19,21,26,30,33,36,39,40,49,51,56 |
| 5 | 11,12,23,26,27,33,34,37,38,40,43,49,50,51 |

## Реализация моделей

Модель представляет данные внутреннего хранилища приложения и методы работы с этими данными. В результате создания приложения Appetizer было разработано 4 класса моделей.

На рис. 8 приведен листинг модели списка покупок.

|  |
| --- |
| class CategoryManager: NSObject {  static func initData() {  let filepath = Bundle.main.path(forResource: "categories", ofType: "csv")!  let contents = try! String(contentsOfFile: filepath)  let csv = try! CSVReader(string: contents, hasHeaderRow: true, delimiter: ";")  var pairedCategoriesDict = [Int: [Int]]()  try! Realm().write {  while csv.next() != nil {  let category = Category()  category.id = Int(csv["id"]!)!  category.name = csv["name"]!  category.isGeneral = csv["isGeneral"]! == "true"  let pairsArray = csv["pairedCategories"]!.split(separator: ",").map({ (substring) -> Int in  return Int(substring)!  })  try! Realm().add(category, update: true)  pairedCategoriesDict[category.id] = pairsArray  }  for pairedCategories in pairedCategoriesDict {  if let category = try! Realm().object(ofType: Category.self, forPrimaryKey: pairedCategories.key) {  for pairedCategoryId in pairedCategories.value {  if let pairedCategory = try! Realm().object(ofType: Category.self, forPrimaryKey: pairedCategoryId){  category.pairedCategories.append(pairedCategory)  }  }  }  }  }  }  } |

**Рис. 8.** Листинг модели списка покупок

Помимо класса *ShoppingList* также были реализованы классы моделей *Category*, *Product*, *Receipt*.

## Реализация контроллеров

Контроллер представляет собой набор классов, которые отвечают за реакцию приложения на те или иные действия пользователя: обрабатывают нажатия на кнопки, сохраняют данные в локальное хранилище, формируют представления на основе данных из моделей.

Реализация контроллеров выполнялась на языке программирования Swift в среде программирования Xcode. В ходе реализации, было разработано 7 классов контроллеров.

На рис. 9 представлен пример кода, используемого для настройки списка покупок.

|  |
| --- |
| import UIKit  import RealmSwift  class ShoppingListSelectionVC: UIViewController, UITableViewDelegate, UITableViewDataSource {    @IBOutlet var tableView: UITableView!  @IBOutlet var header: HeaderView!  @IBOutlet weak var noDataView: UIView!    var shoppingLists: Results<ShoppingList>!    override func viewDidLoad() {  super.viewDidLoad()  navigationController?.navigationBar.isHidden = true  header.setup(withTitle: "Покупки", withSubtitle: "Выберите нужный вам список", withImage: #imageLiteral(resourceName: "cart\_bg"), withBackButtonEnabled: false)  }    override func viewWillAppear(\_ animated: Bool) {  super.viewWillAppear(animated)  self.shoppingLists = try! Realm().objects(ShoppingList.self)  tableView.reloadData()  noDataView.isHidden = self.shoppingLists.count != 0  tableView.isHidden = !noDataView.isHidden  }    func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {  return shoppingLists.count  }    func numberOfSections(in tableView: UITableView) -> Int {  return 1  }    func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {  let cell = tableView.dequeueReusableCell(withIdentifier: "shoppingListCell", for: indexPath) as! ShoppingListCell  cell.setup(withList: shoppingLists[indexPath.row])  return cell  }    func tableView(\_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {  tableView.deselectRow(at: indexPath, animated: true)  }    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {  if let vc = segue.destination as? ShoppingListVC, let shoppingListCell = sender as? ShoppingListCell {  vc.setup(withShoppingList: shoppingListCell.shoppingList)  }  }  } |

**Рис. 9.** Листинг контроллера списка покупок

## Реализация отображений

Стандартный инструмент создания интерфейса для мобильных приложений на операционной системе iOS – *InterfaceBuilder*. Он предоставляет для разработчиков коллекции объектов пользовательского интерфейса, такие как: текстовые поля, форма для ввода даты, таблицы данных, кнопки.

На рис. 10 представлены реализованные экраны блока подбора сочетаемых продуктов, на рис. 11 – блока списка покупок, на рис. 12 – блока сохраненных сочетаний.

|  |  |
| --- | --- |
| а) | б) |

**Рис. 10.** Блок подбора сочетаний:

а) *Экран выбора главного продукта*; б) *Экран подбора сочетаний*

|  |  |
| --- | --- |
| а) | б) |

**Рис. 11.** Блок списка покупок:

а) *Экран списков покупок*; б) *Экран списка продуктов*

|  |  |
| --- | --- |
| а) | б) |

**Рис. 12.** Блок сохраненных сочетаний:

а) *Экран сохранения*; б) *Экран списка сохраненных сочетаний*

1. **ТЕСТИРОВАНИЕ**
   1. **Выбор способов тестирования**

Выделяют несколько основных видов тестирования мобильных приложений:

1. Функциональное тестирование – тестирование программного обеспечения в целях проверки реализуемости функциональных требований;
2. Лабораторное тестирование – тестирование влияния внешних факторов, таких как качество соединения с Интернет;
3. Тестирование производительности;
4. Тестирование утечек памяти;
5. Юзабилити-тестирование – тестирование на удобство работы с интерфейсом;
6. Тестирование на установку;
7. Тестирование на соответствие стандартам – тестирование на соответствие общепринятым стандартам разрабатываемой платформы [14].

Для тестирования реализованного мобильного приложения было выбрано функциональное тестирование.

* 1. **Описание тестов**

Функциональные требования определяют, что именно делает программное обеспечение, какие задачи оно решает. Набор тестов на функциональность представлен в табл. 4.

**Табл. 4.** Тестирование системы

| **№** | **Название теста** | **Шаги** | **Ожидаемый результат** | **Тест пройден?** |
| --- | --- | --- | --- | --- |
| 1 | Подбор сочетания | 1. На экране выбора категории главного продукта выбрать одну из категорий; 2. На экране выбора продукта, принадлежащего к выбранной категории, выбрать продукт; 3. На открывшемся экране выбрать один или несколько продуктов, предложенных системой. | На экране должно отобразиться подобранное пользователем сочетание. | Да |
| 2 | Сохранение подобранного сочетания | 1. На экране с подобранным сочетанием нажать на кнопку с изображением сердечка; 2. Во всплывающем окне написать название для подобранного сочетания; 3. Нажать кнопку «Сохранить»; 4. Перейти на вкладку «Сохраненное»; 5. В списке сохраненных сочетаний найти только что созданное и нажать на него. | Система должна сохранить подобранное сочетание с выбранным названием и при нажатии открывать его. | Да |
| 3 | Удаление подобранного сочетания | 1. На вкладке «Сохраненное» выбрать одно из сочетаний; 2. В открывшемся окне нажать на кнопку с сердечком; 3. Перейти на вкладку «Сохраненное» и убедиться в отсутствии только что удаленного сочетания. | Выбранное сочетание не должно отображаться во вкладке «Сохраненное» | Да |
| 4 | Добавление продуктов в список покупок | 1. На экране с подобранным сочетанием нажать на кнопку с изображением списка; 2. Во всплывающем окне написать название для списка покупок; 3. Нажать на кнопку «Сохранить»; 4. Перейти во вкладку «Покупки» и найти только что созданный список; 5. Нажать на список и убедиться в правильном отображении списка продуктов. | Система должна автоматически добавить все выбранные продукты в список покупок с возможностью отмечать уже имеющиеся продукты. | Да |

# ЛИТЕРАТУРА

1. Статистика мобильных приложений в AppStore. [Электронный ресурс] URL: https://www.theverge.com/2016/6/13/11922926/apple-apps-2-million-wwdc-2016 (дата обращения 19.04.2018).
2. Suhailah Mohd Yusof, Sharifah Fateen Syuhada Syed Lokman. Personal Financial Planner: A Mobile Application that Implementing Forward Chaining Technique for Notification Mechanism // IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), April, 2014, pp. 65-69.
3. Peter Tarasewich. Designing Mobile Commerce Applications // Communications of the acm. Vol. 46, Issue 12, December 2003, pp. 57-60.
4. Документация Apple iOS. [Электронный ресурс] URL: https://developer.apple.com/library/ios/documentation (дата обращения: 19.04.2018).
5. Interface Builder. [Электронный ресурс] URL: https://developer.apple.com/xcode/interface-builder/ (дата обращения: 20.04.2018).
6. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д. Приемы объектно-ориентированного проектирования. Паттерны проектирования; [пер. с англ.: А. Слинкин науч. ред.: Н. Шалаев]. Санкт-Петербург [и др.]: Питер, 2014. 366 с.
7. Марк Д., Наттинг Д., Ламарш Д., Олссон Ф. iOS 6 SDK. Разработка приложений для iPhone, iPad и iPod touch на Objective-C в Xcode. СПб.:Вильямс, 2013. 672 с
8. Влад В. Головач. Дизайн пользовательского интерфейса V 1.2. –
9. Голден Кришна. Хороший интерфейс – невидимый интерфейс. Питер, 2016. 256 с.
10. Арлоу Д., Нейштадт А. UML 2 и Унифицированный процесс:.Практический объектно -–ориентированный анализ и проектирование (пер. с англ. Шатохиной Н.). – 2-е изд., М.: Символ Плюс, 2008. 622 с.
11. Применение специй в кулинарии. [Электронный ресурс] URL: https://vsadu.ru/post/kakie-specii-ispolzovat-v-prigotovlenii.html (дата обращения 20.04.2018).
12. Поваренок.ру. [Электронный ресурс] URL: http://www.povarenok.ru/ (дата обращения 20.04.2018).
13. iOS Databases. [Электронный ресурс] URL: https://rollout.io/blog/ios-databases-sqllite-core-data-realm/ (дата обращения: 20.04.2018).
14. Nimbalkar R. R.: Mobile Application Testing And Challеnges. // International Journal of Science and Research, 2013. – Vol. 2. – P. 56-58.