Санкт-Петербургский государственный университет

Математическое обеспечение и
администрирование информационных систем

Мамро Никита Владимирович

Разработка кулинарного приложения на Android

Отчёт по учебной практике

Научный руководитель:
к.т.н., доцент Литвинов Ю.В.

Санкт-Петербург
2019

Оглавление

Введение

В современном мире человек стремится максимально упростить такие свои бытовые обязанности, как приготовление пищи, в пользу других более важных задач. Одновременно с этим человеческая натура постоянно желает чего-то нового и необычного в гастрономической жизни. В связи с этим у многих любителей разнообразно поесть появляется необходимость иметь доступ к множеству уже составленных и протестированных рецептов.

Раньше для этих целей использовались бумажные кулинарные книги, в которых хранились рецепты любимых блюд. Такой подход был не без недостатков: книги занимали лишнее место, были подвержены порче и, что самое важное, были сильно ограничены в количестве хранимой информации.

Сегодня же более чем у половины населения планеты есть смартфоны[1], на которые можно установить программное обеспечение, способное не только заменить бумажные книги рецептов, но и значительно улучшить и расширить доступную функциональность, например, давая пользователю возможность гораздо быстрее искать нужные рецепты.

В силу популярности обозначенной проблемы, неудивительно, что существует ряд готовых решений, которые позволяют использовать телефон как кулинарную книгу, однако они обладают определёнными недостатками. Желание создать продукт, которым будет удобно пользоваться и к которому нельзя будет применить те же упреки, что и к существующим решениям, вдохновило на реализацию собственного приложения.

[1] Сайт со статистикой по количеству смартфонов — Режим доступа: https://www.bankmycell.com/blog/how-many-phones-are-in-the-world

(Дата обращения: 25.12.2019)

Постановка задачи

Цель работы — создание удобного кулинарного приложения, которое можно будет использовать как книгу рецептов, а именно: просматривать все доступные рецепты, добавлять понравившиеся в избранные, искать рецепты по названию, искать рецепты, которые бы удовлетворяли выбранным параметрам (время приготовления, доступные ингредиенты, тип и кухня, к которым относится блюдо), а также просматривать новые рекомендуемые рецепты.

Над проектом работала команда из 3 человек.

Непосредственно перед автором стояли следующие задачи.

- Разработать и реализовать интерфейс и логику работы экрана с поиском по параметрам.
- Разработать и реализовать представление ленты с рецептами, которые можно просматривать, фильтровать по названию и перемешивать
- Реализовать сохранение данных для использования разными экранами и передачу данных между экранами
- Реализовать тёмную тему для интерфейса приложения и переключение между тёмной и светлой темой

1. Обзор существующих решений

Перед началом разработки приложения были рассмотрены существующие предложения в Google Play. Среди них только два приложения предоставляли пользователю достаточно гибкий поиск по параметрам: «Что готовим?»[2] и «Подбери рецепт»[3]

1.1 «Что готовим?»

В этом приложении поиск по названию осуществляется на том же экране, где и происходит выбор типа блюда, в меню для перехода на этот экран нужно нажать на значок лупы. Однако если пользователь хочет сформировать запрос, исходя из необходимых ингредиентов, ему необходимо будет нажать на иконку холодильника, что вряд ли является интуитивно понятным. После этого пользователь может выбрать список ингредиентов и получить необходимые рецепты, и только после этого полученный список можно отфильтровать по типу блюда и сложности приготовления. Никакого экрана с рекомендациями не предусмотрено. Элементы интерфейса рябят перед глазами, а цветовая гамма не может быть заменена на более тёмную или более светлую, что делает приложение менее удобным для использование в тёмное время суток или при ярком освещении.

1.2 «Подбери рецепт»

Приложение встречает пользователя возможностью выбрать интересующие ингредиенты. Каждая позиция сопровождается изображением ингредиента. После нажатия на элемент интерфейса с ингредиентом, цвет названия ингредиента подсвечивается жёлтым и добавляется в формирующийся запрос. Это не очень удобно, так как кроме цветового индикатора выбранные ингредиенты никак не отображаются, из-за чего можно запутаться при выборе. После предоставления пользователю результатов поиска доступна только фильтрация по типу блюда.

[2] Страница приложения «Что готовим?» на Google Play — Режим доступа: https://play.google.com/store/apps/details?id=ru.gamespace.myfridge

(Дата обращения: 25.12.2019)

[3] Страница приложения «Подбери рецепт» на Google Play — Режим доступа: https://play.google.com/store/apps/details?id=com.ggl.jr.cookbooksearchbyingredients

(Дата обращения: 25.12.2019)

2. Обзор архитектуры приложения

Перед тем, как приступать непосредственно к выполнению задач, была обдумана общая структура приложения, а также продуманы все элементы управления, с которыми будет взаимодействовать пользователь.

Было решено, что для необходимо разработать 4 основных экрана, а также 3 дополнительных экрана, на которые будет переходить пользователь при взаимодействии с интерфейсом основных экранов. 4 основных — лента, поиск по параметрам, настройки, избранное. 3 дополнительных — экран с результатами поиска, экран с рецептом и экран с выбором ингредиентов для добавления их в поисковой запрос.

Другим участником команды разработки приложения было решено использовать стандартные компоненты Android API — фрагменты — для представления каждого из экранов.

Так как на каждом из экранов выводится полезная для пользователя информация, было принято решение, что необходимо поддерживать текущее состояние экрана для каждого раздела приложения, чтобы не было лишних потерь введённых и выведенных данных.

Помимо сохранения состояния каждого из экранов, реализована возможность передачи данных между экранами, которая позволяет разделять экраны, отвечающие формированию поискового запроса и обработки ответа.

3. Поиск по параметрам

Главным экраном приложения является экран расширенного поиска рецептов. По этой причине было необходимо тщательно продумать структуру соответствующего фрагмента и элементов интерфейса.

3.1 Внешний вид экрана поиска

На экране поиска необходимо было разместить все элементы управления, отвечающие за выбор того или иного параметра, при этом нельзя засорять интерфейс, чтобы пользователь мог фокусироваться на составлении запроса. Поэтому было решено сделать крупные кнопки основного цвета приложения для выбора новых параметров, а выбранные позиции добавлять в виде тэгов, которые не занимают много места, не отвлекают пользователя, но при этом представляют возможность проверить, в каком состоянии находится запрос на данный момент, а также возможность удалить неактуальные позиции нажатием на иконку крестика. (Приложение – рисунок 1)

Выбор параметров реализован по средствам открытия диалогового окна с несколькими вариантами выбора (Приложение – рисунок 2)

3.2 Логика работы фрагмента поиска

Группы тэгов, в которых хранились выбранные пользователем параметры поиска, являются одним из ключевых элементов интерфейса, поэтому необходимо было продумать, как именно они будут изменяться при взаимодействии пользователя с приложением.

Первым вариантом решения поставленной задачи было обновление тэгов напрямую из обработчиков нажатий на кнопки «Ок» во всплывающих диалогах с выбором. Однако это решение было оценено, как чересчур некрасивое с точки зрения архитектуры, а также как чересчур медленно и нестабильно работающее. В связи с этим было решено использовать тот факт, что в приложении в любом случае надо использовать объект, в котором бы хранилось состояние параметров поиска.

Как итог, было использовано решение проблемы хранения и передачи данных, описанное в пункте 5.2, которое предоставляет возможность отслеживание текущего состояния данных и динамического обновления тэгов.

В финальной версии приложения обновление элементов интерфейса экрана поиска осуществляется именно через наблюдение за объектом, хранящим данные о текущем сформированным запросом.

4. Лента с рецептами

При анализе существующих решений было замечено, что нет доступных кулинарных приложений, которые бы сами предлагали пользователю новые рецепты, улучшая опыт использования. По этой причине было решено разработать экран с лентой, который бы разнообразил взаимодействие пользователя с приложением. (Приложение – рисунок 3)

4.1 RecyclerView и Adapter

Из-за того, что лента представляет собой список всех доступных рецептов, нужно было понять, как правильнее подойти к процессу вывода её содержимого на экран.

Было обращено внимание на RecyclerView[3], как на наиболее подходящий инструмент для вывода списка, содержащего большое количество элементов. RecyclerView принимает позволяет рендерить в один момент времени ровно столько элементов, сколько может уместиться на экран.

Чтобы заполнять данными RecyclerView, нужно было сделать свой адаптер для RecyclerView, который наследуется от класса Adapter. Для этого необходимо было определить методы, которые ставят в соответствие элементам переданной RecyclerView коллекции элементы самого RecyclerView, создают элементы RecyclerView, а также обрабатывают изменение коллекции, ассоциированной с адаптером.

Более того, существует расширение класса Adapter RealmRecyclerViewAdapter, которое позволяет работать в адаптере с коллекциями формата результатов запросов в базу данных Realm. Так как приложение использовало именно эту базу данных, было решено наследоваться именно от RealmRecyclerViewAdapter для реализации адаптера.

4.2 Поиск по названию

Как можно было заметить, в описании окна с расширенным поиском не был описан поиск по названию ингредиента. Это было сделано во избежание перегрузки интерфейса. Тем не менее очевидно, что такая возможность необходима в подобном приложении, поэтому было решено поиск по названию именно на экран лентой рецептов.

Для обеспечения удобства взаимодействия с поиском по названию, было реализовано обновление содержимого RecyclerView при изменении текста в поле поиска. Таким образом, лента динамически подстраивается под текущий запрос пользователя.

(Приложение – рисунок 4)

4.3 Перемешивание содержимого ленты

Чтобы расширить возможности взаимодействия пользователя с лентой, было решено добавить перемешивание элементов RecyclerView. Для этого был реализован обработчик свайпа сверху низ по ленте. Изначально планировалось просто передать в адаптер перемешанную коллекцию рецептов, однако выяснилось, что коллекцию результатов, полученных из базы данных после исполнения запроса, нельзя было перемешивать. По этой причине для реализации перемешивания содержимого ленты была создана отдельная модель, в которой хранился массив текущих индексов, по которым сопоставляются

5 Работа с данными

При изучении документации Android API было обращено внимание на класс ViewModel[2], который принято использовать с целью обработки данных для активностей и фрагментов. При создании модели ViewModel всегда указывается активность или фрагмент, внутри чьего жизненного цикла и будет существовать модель. Благодаря этому появляется возможность хранить данные вне зависимости от жизненных циклов фрагментов. Также это позволяет фрагментам оперировать с данными, не обращаясь напрямую друг к другу.

5.1 Сохранение данных

Необходимо было сохранять данные в следующей ситуации: пользователь ввёл параметры для поиска рецепта / ввёл название рецепта в строку поиска, переключился на другой экран, вернулся на экран поиска, в этот момент не должны пропадать введённые данные.

Для решения задачи были созданы модели RecipeSearchViewModel и FeedViewModel, унаследованные от класса ViewModel.

В RecipeSearchViewModel хранятся данные о состоянии экрана поиска по параметрам, а именно перечень выбранных типов блюд, кухонь, доступных ингредиентов и ограничения по времени приготовления.

В FeedViewModel хранятся данные о состоянии экрана ленты, а именно о текущем поисковом запросе по имени и о перестановке рецептов в ленте.

Модель создается в ассоциации с главной активностью, что позволяет сохранить данные даже при пересоздании фрагмента. Благодаря этому появилась возможность хранения данных, следующими задачами стали обработка и передача данных, сохраняемых в модели.

5.2 Обработка и передача сохраняемых данных

Учитывая то, что фрагмент с формированием поискового запроса был отделён от фрагмента с выводом результатов поиска, необходимо было иметь возможность передавать данные из одного фрагмента в другой, а также уведомлять все фрагменты, которые используют

Для обработки и передачи данных, сохраняемых в классах RecipeViewModel и FeedViewModel, логичным было иметь возможность наблюдать за изменениями данных, хранящихся в модели. Изучив доступные инструменты, было решено для обёртки хранимых в модели данных использовать класс MutableLiveData[5], который позволяет устанавливать обработчик изменений обёрнутых данных.

Благодаря тому, что стало возможным отслеживать изменения, происходящие внутри модели, был реализован процесс передачи данных между фрагментами.

6. Тёмная тема

Основной мотивацией к поддержке тёмной темы приложением стало отсутствие возможности поменять цветовую гамму интерфейса у существующих решений, что удивительно, учитывая популярность данной функции в современных приложениях. По этой причине, несмотря на несложность задания с технической точки зрения, данная задача была вынесена в отдельный подпункт, как важный аспект для формирования готового приложения.

Для решения задачи были изучены способы применения тем и стилей к приложению, а также возможность во время выполнения программы получать информацию о том, какая установлена в данный момент тема приложения, а также какая установлена в данный момент тема на смартфоне.

Для добавления тёмной темы были использованы средства Material Components[6], а именно тема типа DayNight[8]. Для динамического изменения темы, а также для получения текущей установленной темы, был использован класс AppCompatDelegate[7], в котором есть все необходимые методы.

Как итог, на экран настроек (Приложение – рисунок 5) был добавлен элемент управления, обеспечивающий возможность смены тему на темную (Приложение – рисунок 6) и обратно.

Заключение

В ходе работы были достигнуты данные результаты:

- Разработан интерфейс кулинарного приложения, добавлена возможность переключения между светлой и тёмной темами
- Настроена работа с данными внутри приложения
- Разработана логика работы экранов, отвечающих поиску рецепта по параметрам и ленте с рецептами

Код проекта доступен по ссылке[4]

[4] Репозиторий с проектом на Github — Режим доступа: https://github.com/mxprshn/coolky (Дата обращения: 25.12.2019)

Список литературы

[1] Разработка Android-приложений на Kotlin [Электронный ресурс]. — Режим доступа: https://stepik.org/course/4792/ (Дата обращения: 25.12.2019)

[2] ViewModel — Режим доступа: https://developer.android.com/reference/androidx/lifecycle/ViewModel

(Дата обращения: 25.12.2019)

[3] RecyclerView — Режим доступа: https://developer.android.com/guide/topics/ui/layout/recyclerview (Дата обращения: 25.12.2019)

[4] Adapter — Режим доступа: https://developer.android.com/reference/android/support/v7/widget/RecyclerView.Adapter

(Дата обращения: 25.12.2019)

[5] MutableLiveData — Режим доступа: https://developer.android.com/reference/androidx/lifecycle/MutableLiveData

(Дата обращения: 25.12.2019)

[6] Material components — Режим доступа: https://material.io (Дата обращения: 25.12.2019)

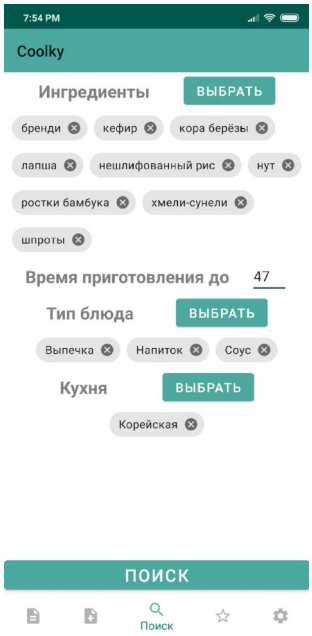
[7] AppCompatDelegate — Режим доступа: https://developer.android.com/reference/android/support/v7/app/AppCompatDelegate

(Дата обращения: 25.12.2019)

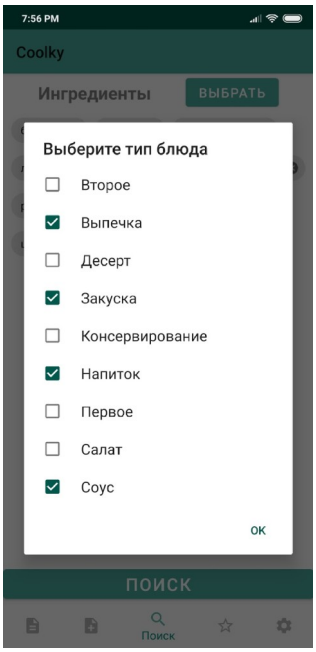
[8] DayNight — Режим доступа: https://developer.android.com/guide/topics/ui/look-and-feel/darktheme

(Дата обращения: 25.12.2019)

Приложение

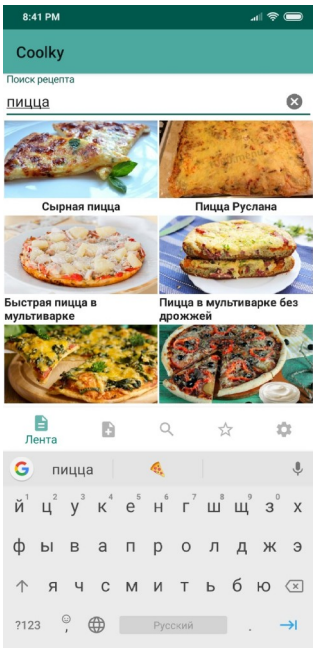


Приложение – рисунок 1



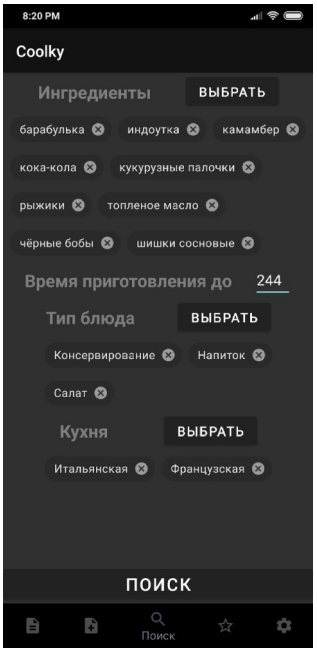
Приложение – рисунок 2

Приложение – рисунок 3



Приложение – рисунок 4

Приложение – рисунок 5



Приложение – рисунок 6