

Санкт-Петербургский государственный университет

Математико-механический факультет

Паршин Максим Алексеевич

Разработка кулинарного  
Android-приложения: база данных,  
отображение результатов поиска рецептов

Отчёт по учебной практике

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

Санкт-Петербург  
2019

# Оглавление

Введение	3
Постановка задачи	4
Обзор существующих решений	5
Обзор использованных инструментов	7
Реализация	9
Результаты	15
Список литературы	16

# Введение

Целью большого числа мобильных приложений ввиду их доступности и простоты часто является облегчение быта пользователя, в том числе и процесса приготовления пищи. Выбор блюда для приготовления, к примеру, становится проще при наличии доступа к базам данных с рецептами, который предоставляют многие приложения. При этом для эффективного поиска рецептов важно иметь возможность фильтрации по такому параметру как необходимые ингредиенты. На данный момент существует несколько русскоязычных Android-приложений, обладающих таким функционалом, но имеющих недостаток в виде недружественного пользователю интерфейса, затрудняющего работу с ними.

Цель проекта — разработать приложение для операционной системы Android, имеющее функцию поиска кулинарных рецептов по входящим в них ингредиентам, а также другим параметрам, и интуитивно-понятный интерфейс.

# Постановка задачи

Для достижения цели перед автором были поставлены следующие задачи:

- Выбрать СУБД для взаимодействия с базой данных рецептов
- Разработать архитектуру базы данных
- Реализовать инструмент для заполнения базы тестовыми данными
- Реализовать механизм доступа к данным
- Разработать элементы интерфейса для отображения результатов поиска

# Обзор существующих решений

При исследовании доступных в Google Play русскоязычных кулинарных приложений было найдено только две программы, обладающих функцией поиска рецепта по необходимым ингредиентам: «Что готовим?» [2] и «Подбери рецепт» [5].

## Приложение «Что готовим?»

Поиск рецептов по ингредиентам в данном приложении осуществляется путем заполнения пользователем раздела «Холодильник» элементами имеющихся у него продуктов и формирования по ним поискового запроса. В окне результатов поиска доступна фильтрация по сложности приготовления блюда и его типу, для каждого из рецептов отображается общее число необходимых и число совпавших с заданными ингредиентов, но сортировка по этому критерию не происходит. На странице отдельного рецепта содержится инструкция по приготовлению, а также информация о времени приготовления и количестве ингредиентов.

## Приложение «Подбери рецепт»

Интерфейс выбора необходимых ингредиентов в данном приложении доступен пользователю сразу при запуске. Имеются изображения для каждого ингредиента. В окне результатов поиска для каждого найденного рецепта отображается только число совпавших ингредиентов, но не общее их число в блюде. Доступна фильтрация по типу блюда. На странице рецепта можно найти инструкцию по приготовлению, информацию о количестве ингредиентов, времени приготовления и пищевой ценности. При этом страница оформлена достаточно броско, что может затруднять комфортное использование.

Подводя итог, можно сказать, что данные приложения, несмотря на наличие функциональности, достаточной для использования в быту, имеют недостатки в виде отсутствия информации о числе порций,

на которое рассчитан рецепт, и принадлежности к национальной кухне, а также относительно сложного в освоении интерфейса в случае приложения «Что готовим?» и затрудняющего восприятие оформления в «Подбери рецепт».

# Обзор использованных инструментов

## Realm

Требуемую функциональность приложения было бы невозможно реализовать без базы данных с рецептами, поэтому важной задачей стал выбор подходящей СУБД. При этом ввиду отсутствия возможности разместить базу данных на удаленном сервере было принято решение хранить информацию о рецептах в локальной базе данных на устройстве, предварительно заполненной на другой платформе и перенесенной в приложение.

С учетом современных тенденций в мобильной разработке, как возможные альтернативы были рассмотрены реляционная СУБД SQLite (и ORM-технологии) и появившаяся сравнительно недавно объектно-ориентированная NoSQL база данных Realm, которая в итоге и была выбрана для использования в проекте. Основным аргументом в пользу этой СУБД стала возможность быстрой интеграции базы данных, заполненной с помощью десктопного приложения, в Android-приложение без необходимости изучения двух ORM-технологий для разных платформ, что увеличило бы время работы над проектом. Realm имеет одинаковый механизм доступа к объектам данных на всех поддерживаемых платформах, и для интеграции базы в Android-приложение в его коде потребовалось лишь определить такие же классы-сущности, как при её заполнении, и затем обращаться к данным с помощью них.

Что касается скорости работы, согласно исследованию, проведенному пользователем GitHub с именем аккаунта AlexeyZatsepin [1], Realm показывает сравнимое с другими технологиями время чтения из базы данных и с большим отрывом (Рис. 1) опережает их по времени записи.

## AngleSharp

Для апробации разрабатываемого приложения требовалось заполнить базу тестовыми данными, поэтому было принято решение реализовать отдельный инструмент, задачей которого было бы получение ин-

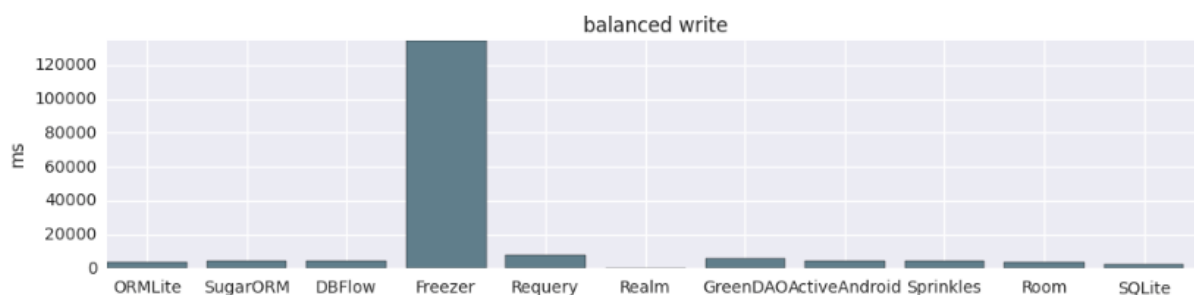


Рис. 1: Сравнение времени выполнения операции записи для популярных мобильных баз данных

формации о рецептах с некоторого веб-сайта. Так как эта вспомогательная программа сама по себе не использовалась бы при эксплуатации Android-приложения, написанного на Kotlin, она была вынесена в отдельный десктопный модуль. Языком разработки стал C#, опытом программирования на котором обладал автор. Чтобы извлечь определенные данные о рецепте из исходного кода веб-страницы, потребовалось воспользоваться одной из библиотек для парсинга HTML-разметки. После изучения исследования популярных инструментов, проведенного пользователем портала [habr.com](http://habr.com) Павлом Носовичем [6] и показавшего, что библиотека AngleSharp обладает рядом преимуществ, таких как возможность использования CSS-селекторов, было решено использовать именно её.



# Реализация

## Парсер

При разработке вспомогательного приложения-парсера HTML-разметки требовалось обеспечить возможность его расширения для работы с различными сайтами и организовать единый интерфейс доступа к информации, содержащейся на страницах с рецептами, для каждого из них. Чтобы решить данную задачу, была спроектирована и реализована следующая архитектура классов (Рис. 2):

- **Интерфейс IParsingLogic** Данный интерфейс наследуется для работы с отдельным веб-сайтом. Для каждого элемента данных о рецепте, например, времени приготовления, реализуется метод, с помощью библиотеки AngleSharp извлекающий эти данные из HTML-разметки переданной страницы.
- **Класс ParsingContext** При обработке страниц для некоторых сайтов может быть необходимо учитывать не только информацию о рецепте, содержащуюся непосредственно на них, но и информацию, определяемую по разделу сайта, где страница находится. К примеру, на странице с рецептом может быть не указано, что он относится к итальянской кухне, но по тому факту, что страница находится в разделе "Итальянская кухня", эту информацию можно выяснить. В таком случае можно унаследовать класс ParsingContext для раздела итальянской кухни и переопределить метод GetCuisine(), чтобы он возвращал название «итальянская». Остальные же методы будут просто вызывать соответствующие методы переданного им экземпляра IParsingLogic. Также класс ParsingContext содержит метод GetPages(), возвращающий коллекцию строк адресов страниц, относящихся к данному разделу.
- **Интерфейс IParserFactory** Реализуется для отдельного сайта и позволяет объединить всю логику работы с ним в одной сущности.

- **Класс RecipeParser** Используя переданный в конструкторе экземпляр IParserFactory осуществляет обход по страницам разделов определенного сайта, извлекает информацию о рецептах и обращается к классу, работающему с базой данных, для сохранения полученных рецептов.

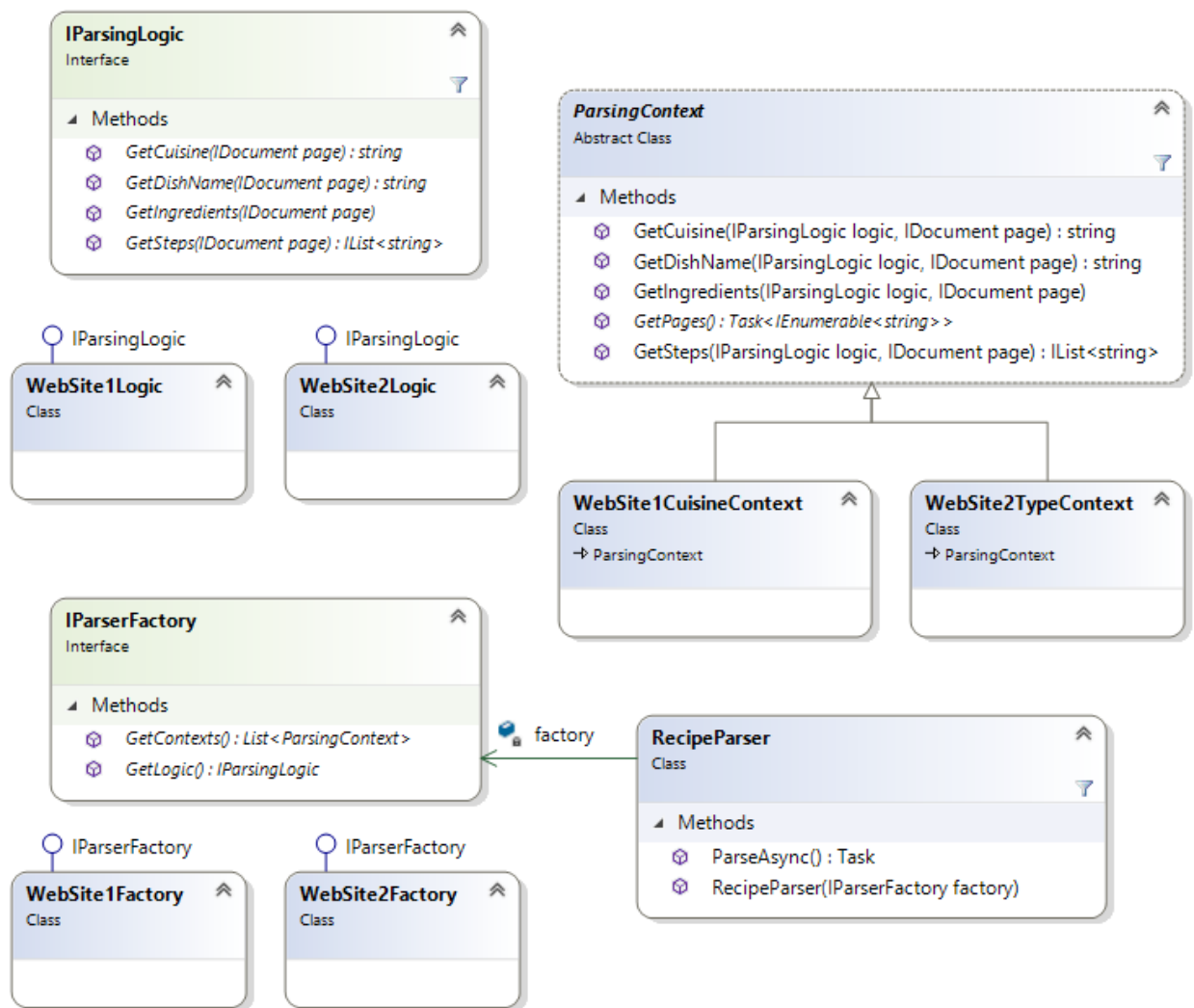


Рис. 2: Диаграмма классов парсера

## База данных

Для хранения информации о рецептах была реализована архитектура, состоящая из трех классов (Рис. 3), наследующих RealmObject

— базовый класс Realm, представляющий отдельную сущность в базе данных (аналогично таблице в реляционных СУБД):

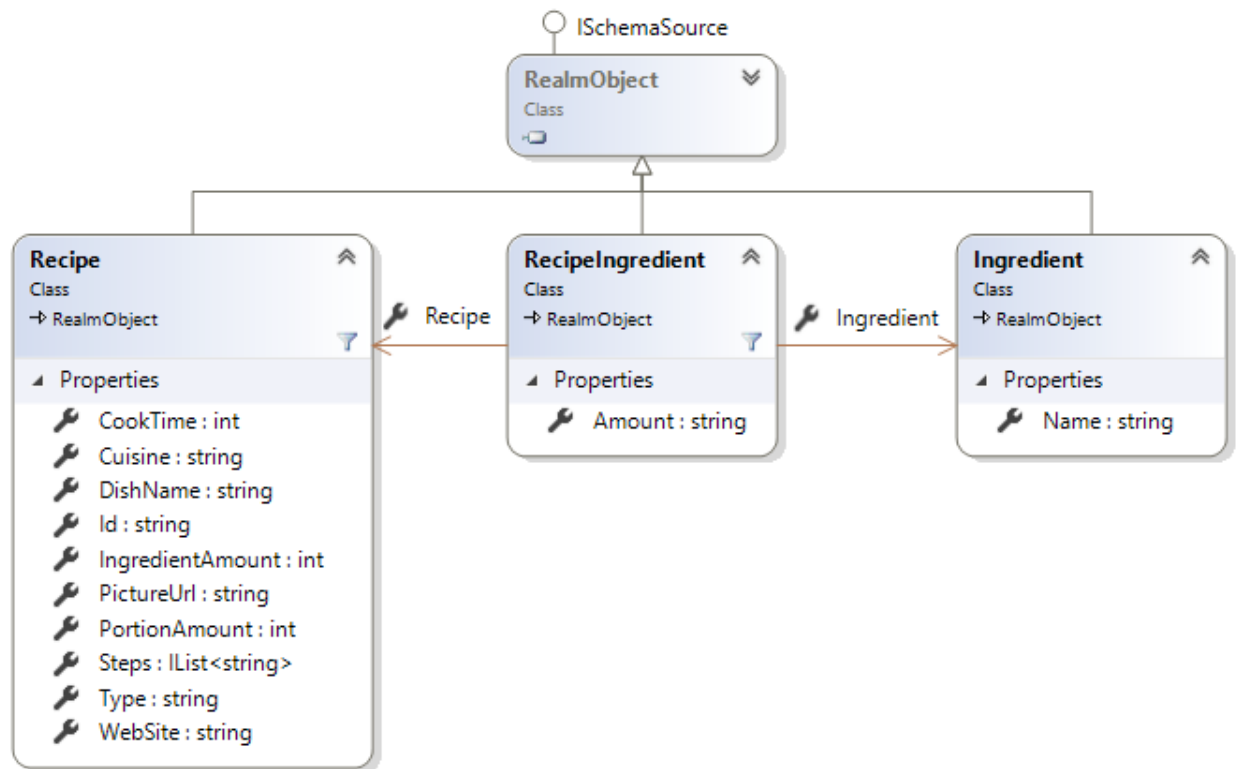


Рис. 3: Диаграмма классов базы данных

- **Recipe** Объекты данного класса содержат информацию об отдельном рецепте: название блюда, шаги приготовления, число порций, принадлежность к национальной кухне и т. д.
- **Ingredient** Класс, представляющий ингредиент, который может содержаться в блюде. Имеет свойство Name, возвращающее название ингредиента.
- **RecipeIngredient** Класс, используемый для представления связи между рецептами и необходимыми для приготовления ингредиентами. Имеет свойства Recipe и Ingredient, возвращающие ссылки соответственно на рецепт блюда и конкретный его ингредиент. Свойство Amount предоставляет информацию о количестве, в котором ингредиент должен использоваться.

DBProvider
+ findIngredients(input: String) + findRecipes(ingredients: Array<String>, types : Array<String>, cuisines: Array<String>, time: Int) + findRecipeById(recipeId: String) + findIngredientByName(ingredientName: String)

Рис. 4: Класс DBProvider, используемый в Android-приложении (Kotlin)

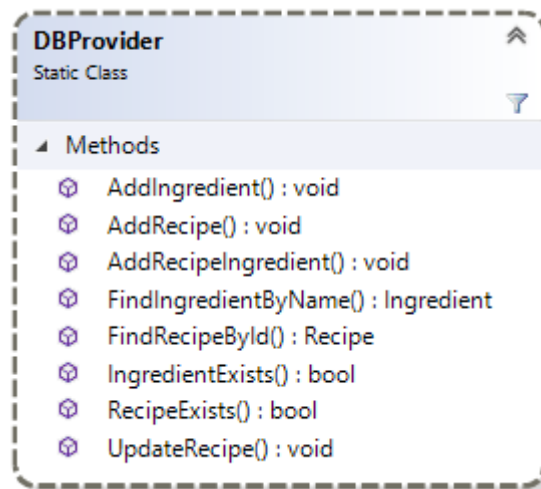


Рис. 5: Класс DBProvider, используемый в парсере (C#)

Описанная архитектура используется для взаимодействия с данными как в парсере, так и в Android-приложении.

Для осуществления операций с базой данных, таких как поиск рецептов по параметрам и запись новых рецептов в базу, в обоих модулях реализованы служебные классы DBProvider (Рис. 5 и 4).

## Отображение результатов поиска

Отображение найденных рецептов в мобильном приложении на Kotlin осуществляется с помощью фрагментов — стандартных компонентов Android API, представляющих собой модули, имеющие пользовательский интерфейс и содержащие логику взаимодействия с ним [3]. Реализация пользовательских фрагментов производилась путем наследования от класса Fragment и переопределения методов, связанных с «жиз-

ненным циклом» фрагмента, таких как `OnCreate()`, который вызывается при создании фрагмента приложением. С каждым фрагментом связан `.xml` файл, описывающий его графическое представление.



### Яблочный пирог Осень

80  
мин.

8  
порций

#### ИНГРЕДИЕНТЫ

#### ПРИГОТОВЛЕНИЕ

мука	• 250 гр
кефир	• 220 мл
сахар	• 120 гр
яйца	• 3 шт.

Рис. 6: Графический интерфейс фрагмента `RecipeFragment`

- **`RecipeFragment`** (Рис. 6) Фрагмент отдельного рецепта, имеет вложенные фрагменты `RecipeIngredientListFragment` и `RecipeStepListFragment`, отображающиеся в переключаемых вкладках и содержащие список ингредиентов и шаги приготовления соответственно. Так как приложение может инициировать перерисовку графического интерфейса фрагмента, данные о рецепте должны находиться в объекте, чьё состояние не зависит от таких изменений. Для этого существует библиотечный класс `ViewModel`, от которого в данном случае наследуется `RecipeInfoViewModel`, содержащий отображаемую информацию: название блюда, инструкцию и т. д.
- **`RecipeSearchResultsFragment`** (Рис. 7) Фрагмент со списком результатов поиска по запросу. Реализован при помощи компонента `RecyclerView` [4], использующего класс `SearchResultsListAdapter`







	<b>Бисквитный пирог с яблоками</b> <i>выпечка</i>	Нужно ингредиентов: 5
	<b>Шарлотка с карамелью</b> <i>выпечка</i>	Нужно ингредиентов: 5
	<b>Шарлотка в хлебопечке</b> <i>выпечка</i>	Нужно ингредиентов: 5
	<b>Слоеный пирог с яблоками и корицей</b> <i>выпечка</i>	Нужно ингредиентов: 5
	<b>Шарлотка классическая</b> <i>выпечка</i>	Нужно ингредиентов: 5
	<b>Шарлотка с яблоками и клюквой</b> <i>выпечка</i>	

Рис. 7: Графический интерфейс фрагмента RecipeSearchResultsFragment

для заполнения отдельного элемента списка соответствующими данными из базы.

# Результаты

- Спроектирована архитектура базы данных
- Реализован инструмент для заполнения базы тестовыми данными
- Реализован механизм доступа к данным
- Реализованы элементы интерфейса для отображения результатов поиска

## Список литературы

- [1] AlexeyZatsepin. Performance comparison of Android ORM Frameworks // GitHub. — Режим доступа: <https://github.com/AlexeyZatsepin/Android-ORM-benchmark> (дата обращения: 23.12.2019).
- [2] Destiny Sphere. Что готовим? // Google Play. — Режим доступа: <https://play.google.com/store/apps/details?id=ru.gamespace.myfridge&hl=ru> (дата обращения: 23.12.2019).
- [3] Fragments // Android API Reference. — Режим доступа: <https://developer.android.com/guide/components/fragments> (дата обращения: 25.12.2019).
- [4] RecyclerView // Android API Reference. — Режим доступа: <https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView.html> (дата обращения: 25.12.2019).
- [5] Y&M. Подбери рецепт // Google Play. — Режим доступа: <https://play.google.com/store/apps/details?id=com.ggl.jr.cookbooksearchbyingredients&hl=ru> (дата обращения: 23.12.2019).
- [6] Павел Носович. Распарсить HTML в .NET и выжить: анализ и сравнение библиотек // Хабр. — 2015. — Режим доступа: <https://habr.com/ru/post/273807/> (дата обращения: 24.12.2019).