Android приложение REST клиент

# Описание

Готовое решение будет представлять собой Android приложение, обладающее следующим функционалом: отправка и составление запросов к REST API, вывод ответа на запрос на экран, как Json строку, сохранение истории запросов пользователя в базу данных, чтобы он мог повторять сценарии, которые всегда будут под рукой. Так же будет сохраняться ответ на запрос и выводиться, чтобы в истории отображалось, какие запросы были успешны, а какие нет.  
У пользователя будет возможность создавать группы запросов, чтобы удобно переключаться между ними

## Наименование

REST Client

## Предметная область

Имитация клиента мобильного приложения, тестирование серверного продукта, разработка

# Данные

В приложении будут использоваться следующие модели данных:

**ResponseRequestModel** – основной контейнер для хранения данных. Он включает в себя айди, запрос и ответ. Таблица *ResponseRequest* в базе данных будет выглядеть как один набор данных (то есть все поля из классов *ResquestModel* и *ResponseModel* будут автоматически вынесены в эту таблицу, а при использовании *SELECT* – корректно распределены между классами)

**Листинг ResponseRequestModel**

data class ResponseRequestModel*(* var id: Int? = null,  
 var request: RequestModel,  
 var response: ResponseModel  
*)*

**RequestModel** – модель, используемая для хранения данных о запросе.  
Листинг **RequestModel**  
data class RequestModel*(* var method: RequestMethod,  
 var url: String = "",  
 var headers: MutableMap*<*String, String*>* = *mutableMapOf()*,  
 var query: MutableMap*<*String, String*>* = *mutableMapOf()*,  
 var body: String = ""  
*)*

Листинг **RequestMethod**  
enum class RequestMethod*{  
 GET*, *POST  
}*

**ResponseModel** – модель, используемая для хранения основных данных об ответе сервера

Листинг **ResponseModel**

data class ResponseModel*(* var code: Int,  
 var message: String,  
 var body: String  
*)*

**ScriptModel** – модель, используемая для хранения сценариев запросов. Она включает в себя свой айди, название сценария, а также список айди входящих в неё запросов  
Листинг **ScriptModel**

data class ScriptModel*(* var id: Int? = null,  
 var name: String = "",  
 var itemsId: List*<*Int*>* = *listOf()  
)*

Тем самым мы имеем минимальный набор данные для работы и хранения. Использование такого ограниченного набора объясняется тем, что приложение должно быть вспомогательным и легковесным инструментом, которое будет быстро реагировать на действия пользователя и быстро выполнять запросы к БД.

# Пользовательские роли

Существует единственная роль, для которого предназначено приложение – это тестировщик, либо разработчик. Так как приложение будет удобным, а самое главное мобильным инструментом, чтобы протестировать работу серверных методов, то можно использовать его повсеместно. Главное – иметь подключение к интернету

# UI

В приложении будет реализован пользовательский интерфейс. Он будет состоять из двух основных экранов и нижнего меню, для переключения между ними

## Экран быстрого создания запроса, а также истории запросов

Экран будет состоять из двух модулей. Модуль запрос-ответ и модуль история запросов

## Модуль запрос-ответ

В этом модуле будет находится текстовое поле с выпадающим списком, с возможностью выбора метода запроса (POST, GET), рядом будет находится поле ввода для URL сервера, куда будет выполняться запрос.

Ниже будет находиться список заголовков (Headers). Каждый элемент списка будет представлять из себя пару двух полей ввода (ключ, значение), а также иконка в виде крестика, чтобы удалить заголовок из списка. Прямо под списком будет находится кнопка добавить, чтобы добавить новый заголовок в список.

Под списком заголовков будет находится список параметров запроса (Query). Он будет реализован аналогично со списком заголовков.

Под списком параметров запроса будет находится большое текстовое поле для ввода тела запроса. Сюда нужно будет вводить тело, как JSON-строку.

Под телом запроса будет находится кнопка отправить, которая будет отправлять запрос на сервер.

Под кнопкой Отправить будет находится текстовое поле с отображением основных параметров ответа (CODE и MESSAGE), а также большое текстовое поле с телом ответа

## Модуль история запросов

В этом модуле будет располагаться список истории запросов, которые были совершены на устройстве пользователя из данного приложения, а также поиск по истории, который будет осуществляться по совпадению в url. Внешний вид элемента списка будет принимать следующий вид: текстовое поле с наименованием метода запроса (POST, GET), рядом с ним будет отображено текстовое поле с URL запроса. Под этими полями будет находиться CODE и MESSAGE ответа, чтобы видеть, какие запросы выполнились удачно, а какие с ошибкой. В правой части элемента будет крестик, чтобы удалить выбранный запрос из базы данных.

При нажатии на элемент списка данные запроса и ответа подставляются в соответствующие поля в Модуль запрос-ответ

## Сценарии

Этот модуль будет состоять из двух экранов: список сценариев и детальная информация о сценарии

## Список сценариев

На списке сценариев будет реализован поиск по имени сценариев, а также будет кнопка добавить. При нажатии на кнопку добавить будет создаваться временный сценарий и открываться его детальная информация. По нажатию на уже созданный сценарий из списка, будет открываться экран с детальной информацией о нём

## Детальная информация о сценарии

На этом экране будет находиться список модулей запрос-ответ, который будет реализован с помощью View Pager. Также можно будет по кнопке добавить новый запрос-ответ, и он будет прикреплен к этому сценарию.

# Технологии разработки

## Общая бизнес-логика

При нажатии кнопки отправить пользователем будут выполняться следующие действия:

1. Формирование запроса и отправление по указанному адресу с указанным методом
2. Отображение значка загрузки, пока запрос выполняется
3. Получение ответа от сервера
4. Сохранение данных запрос-ответ в базу данных и отображение его в общем списке запросов после сохранения\*
5. Отображение данных ответа от сервера в интерфейсе пользователя\*

Действия 4 и 5 будут выполняться параллельно в разных потоках

Логика сохранения запросов:

1. Пользователь ввёл новые данные (URL, Headers, Query, Body) и нажал отправить => При получении ответа запрос будет сохранён как новый
2. Пользователь не вводил новые данные и нажал отправить => При получении ответа будет перезаписан ответ уже существующего запроса

## Язык программирования

В проекте будут использованы следующие языки и технологии: Kotlin, Android SDK, OkHttp (для управления запросов к сети), Room (высокоуровневый интерфейс SQLite), Android Navigation Component (навигация между экранами), Kotlin Coroutines (работа с потоками). Архитектура проекта будет построена на основе Clean Architecture и MVVM.

## СУБД

SQLite. Это компактная встраиваемая СУБД, которая по рекомендуется и по умолчанию используется в Android