Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут”

Кафедра ОТ

**ЗВІТ**

про виконання проектної роботи

з дисципліни

“Технології створення ПП для моб. платформ”

|  |  |
| --- | --- |
|  | Виконав: |
|  | Студент 3 курсу  гр. ІП-53 ФІОТ  Бойко Сергій Сергійович |

Київ – 2018

**ЗМІСТ:**

[1 Мета роботи 3](#_Toc483954463)

[2 ВИкористані бібліотеки 4](#_Toc483954464)

[3 Діаграма класів та структура проекту 13](#_Toc483954465)

[4 Код програми 20](#_Toc483954466)

[5 Приклади виконання програми 21](#_Toc483954467)

[Висновок 24](#_Toc483954468)

# Мета роботи

Цель работы – реализовать мобильное приложение, которое позволит пользователю увидеть возможности реализации мобильного приложения с использованием базы данных, акселерометра и загрузка с отображением изображений.

# Використані бібліотеки

**1. Библиотека Butter Knife**

Библиотека для оптимизация кода

Butter Knife – популярная библиотека для «инжектирования» Views, разработанная Джейком Уортоном (Square Inc.). Данная библиотека использует аннотации, на основе которых генерируется тот самый boilerplate code. Это никак не влияет на производительность приложения, так как кодогенерация происходит во время компиляции.

При разработке мобильных приложений сталкиваемся с проблемой «boilerplate code», сильно надоедающим из-за присутствия в любом приложении под Android.

boilerplate code (шаблонный код) – это часть кода, которая должна быть включена во многие места с небольшими отличиями или без них. Boilerplate – это повторение, то, чего необходимо избегать для хорошего стиля программирования.

Пример:

*EditText nameView = (EditText) findViewById(R.id.nameView);*

Если View понадобится объявить в поле класса, тогда количество строк кода увеличивается минимум в два раза.

*private EditText nameView;*

*@Override*

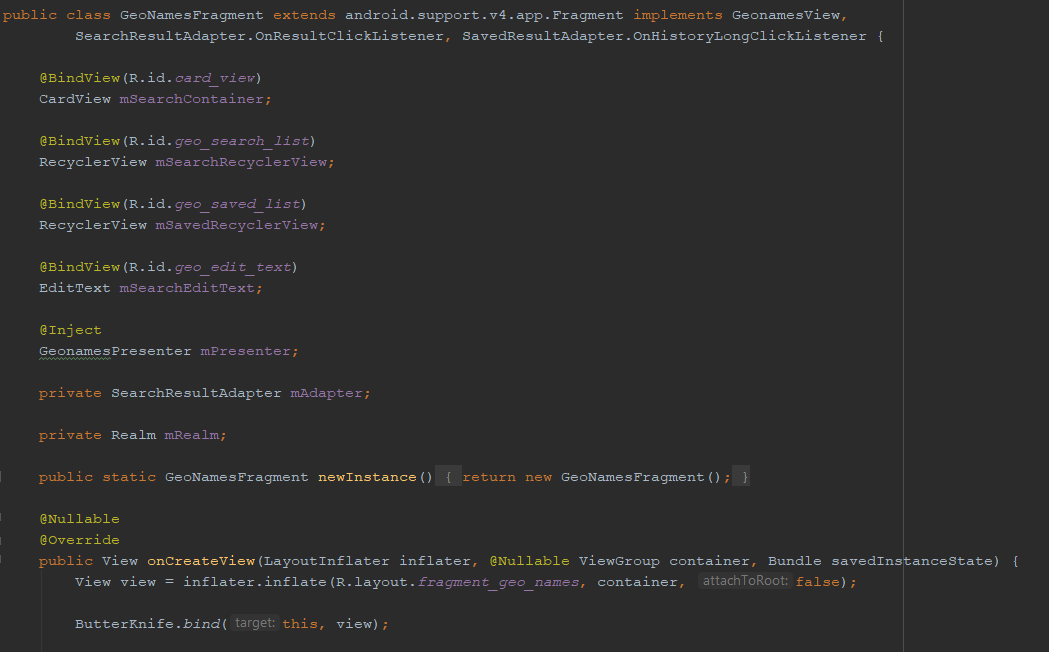
*protected void onCreate(Bundle savedInstanceState) {*

*super.onCreate(savedInstanceState);*

*setContentView(R.layout.activity\_main\_old);*

*nameView = (EditText) findViewById(R.id.nameView);*

*}*



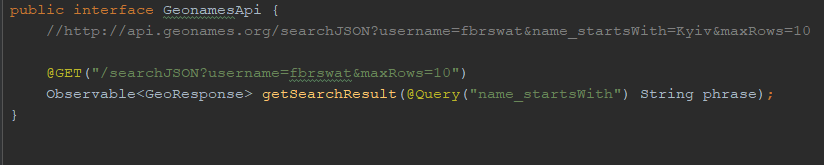
**2. Библиотека retrofit**

Retrofit (согласно официальному сайту) — типобезопасный HTTP-клиент для Android и Java. Он является незаменимым инструментом для работы с API в клиент-серверных приложениях.

Библиотеку Retrofit можно подключить тремя способами: Gradle, Maven, Jar. Мы использовали Gradle

Для подключения в файл build.gradle модуля приложения в раздел dependencies вставляем строчку:

compile 'com.squareup.retrofit2:retrofit:2.1.0'



**3. Библиотека Dagger**

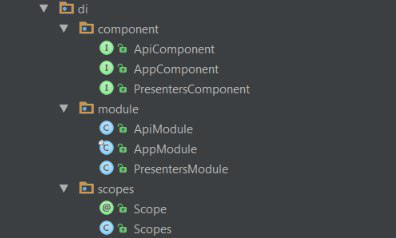
Dagger 2 представляет собой библиотеку, которая помогает реализовать паттерн Внедрение зависимости (Dependency Injection), который в свою очередь является "специфичной формой инверсии управления (Inversion of control)".

Внедрение зависимостей (Dependency injection, DI) — техника, упрощающая покрытие приложения тестами, а Dagger — один из самых популярных Java/Android фреймворков, предназначенных для этой цели.

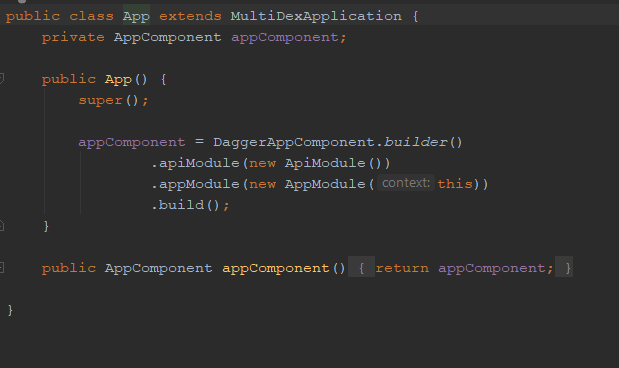
Принципы инверсии управления

1.Модули верхних уровней не должны зависеть от модулей нижних уровней. Модули обоих уровней должны зависеть от абстракций.

2.Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.







**5. Библиотека rxJava**

RxJava - библиотека для функционального реактивного программирования .

Главный принцип RxJava основан на паттерне проектирования Observer (наблюдатель). У нас есть Observable — это источник данных, который выдает нам определенный поток с данными и также у нас есть Subscriber, который получает эти данные.

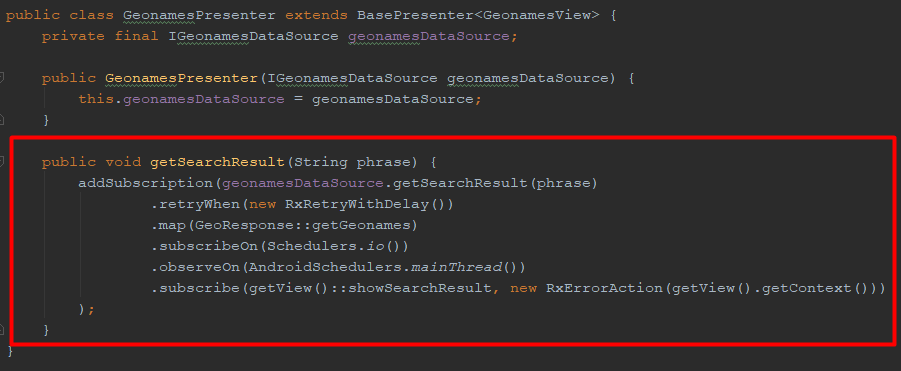
Проблемы, которые решает RxJava:

1. Позволяет уйти от кучи вложенных колбэков

2. Делает работу с потоками удобнее и проще

3. Содержит гибкие механизмы для обработки ошибок

4. Предоставляет обширный функционал за счет большого числа операторов



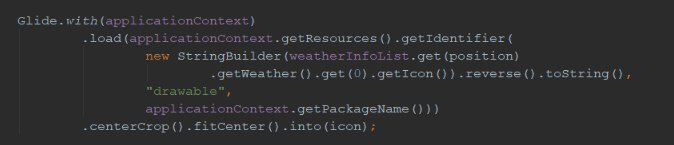
**6. Библиотека Glide**

Glide может загружать и отображать изображения из многих источников, а также заботится о кешировании и использование небольшого количества памяти при выполнении манипуляций с изображением

Библиотека Glide использует fluent interface. Билдер запросов Glide требует как минимум три параметра:

1. with(Context context) — сюда просто передаем Context. Это будет строка, представляющая из себя URL-адрес картинки.

2. into(ImageView targetImageView) — здесь указывается ImageView, в котором должна отобразиться картинка.



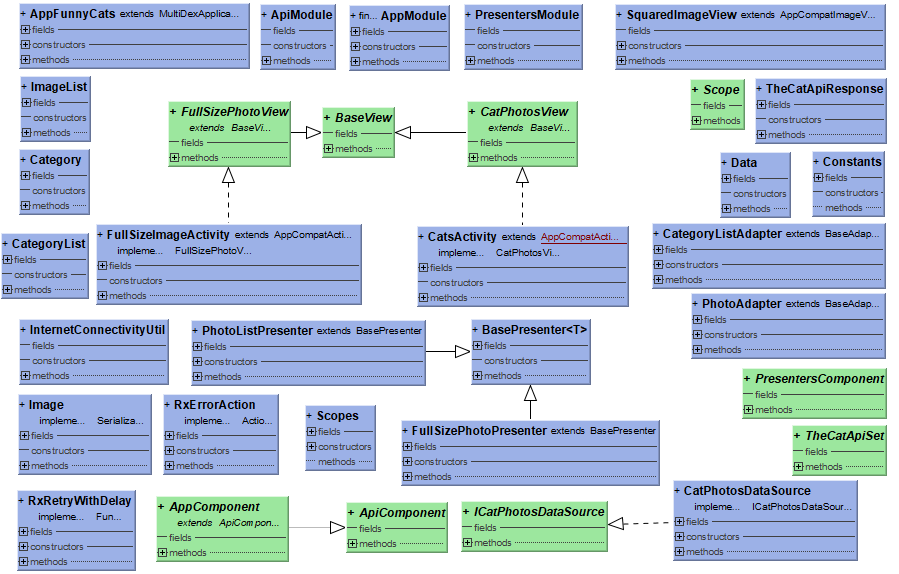
**7. Realm**

Realm — кроссплатформенная мобильная база данных.

База данных Realm быстрее альтернативных ORM на Android примерно в 100 раз (по сравнению с SQLite — в 10 раз). А скорость взаимодействия с БД напрямую влияет на время отклика приложения. Если вы хотите, чтобы оно быстро реагировало на запросы пользователя, не допуская лагов и зависаний, стоит работать именно с Realm.

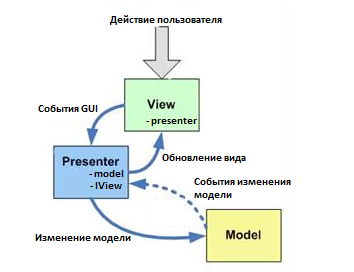
# Діаграма класів та структура проекту

UML-диаграмма:

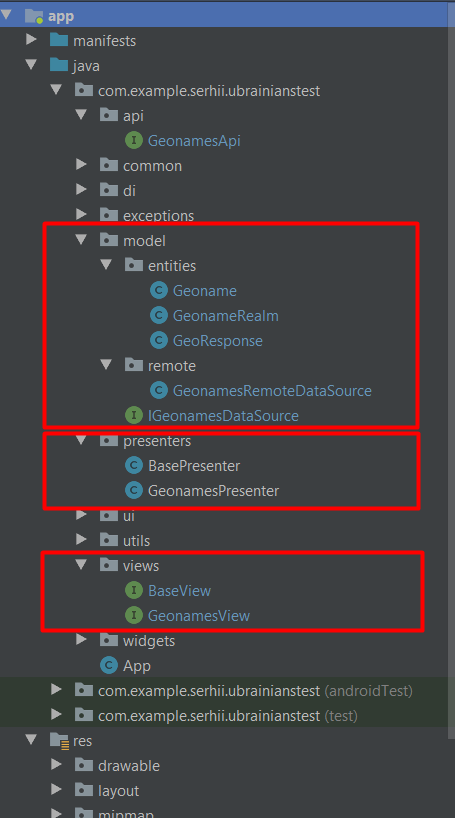


**Структура проекта:**

Во время написания проекта был использован рекомендуемый компанией-владельцем платформы Android подход – MVP. Данный подход позволяет создавать абстракцию представления. Для этого необходимо выделить интерфейс представления с определенным набором свойств и методов. Презентер, в свою очередь, получает ссылку на реализацию интерфейса, подписывается на события представления и по запросу изменяет модель. Если разбирать этот паттерн с помощью схемы взаимодействия компонентов, мы получим следующую схему:

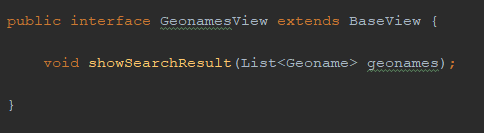


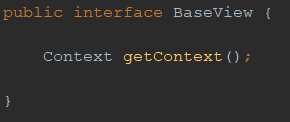
Рассмотрим использование структуры на конкретном проекте:



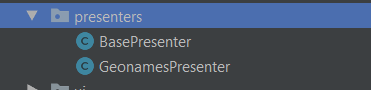
В проекте присутствуют пакеты для хранения каждого из компонентов паттерна MVP – model, view, presenters, каждый из которых содержит соответствующие классы:

1. Пакет model содержит классы, которые отвечают за хранение модели в проекте. Они инициализируются данными, полученными из интернета. Преобразование в нужный класс из объектов XML, описание и примеры кода из приложения присутствуют в соответствующей части этого документа.
2. Пакет view содержит интерфейсы, с помощью которых происходит связь презентера и графического интерфейса.





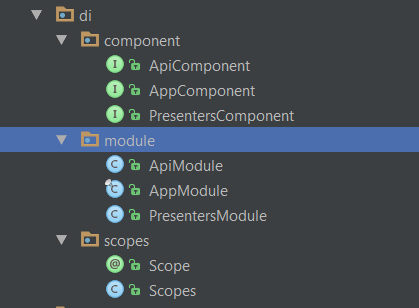
1. Пакет presenters содержит классы-презентеры для выполнения логики приложения. В нашем случае это загрузка данных из интернета.



Загрузка данных из интернета происходит с помощью библиотеки RXJava, описание с примерами кода присутствуют в соответствующем разделе этого документа.

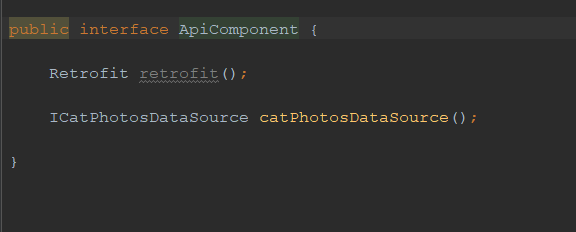
**Структура приложения**

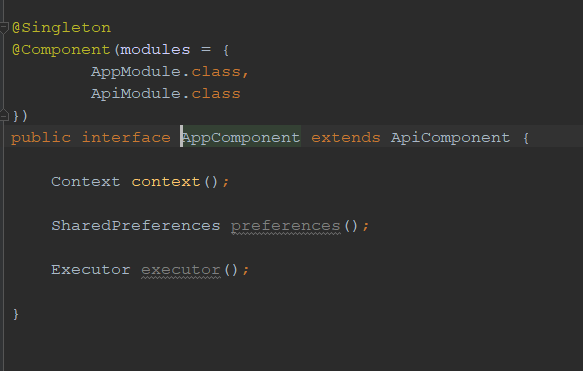
Структура приложения построена на предоставлении внешней зависимости программному компоненту используя внедрение зависимостей – dependency injection. В проекте это реализовано с помощью библиотеки Dagger 2.

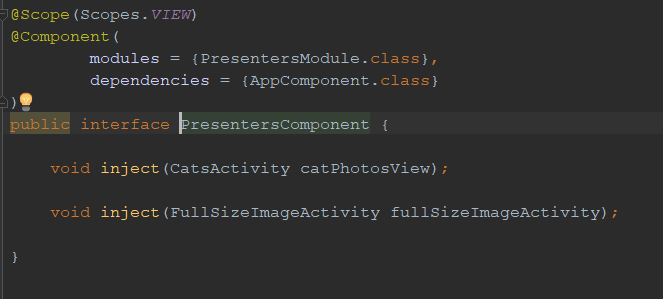


Каждый подпакет из пакета di содержит код для внедрения соответствующих компонентов приложения:

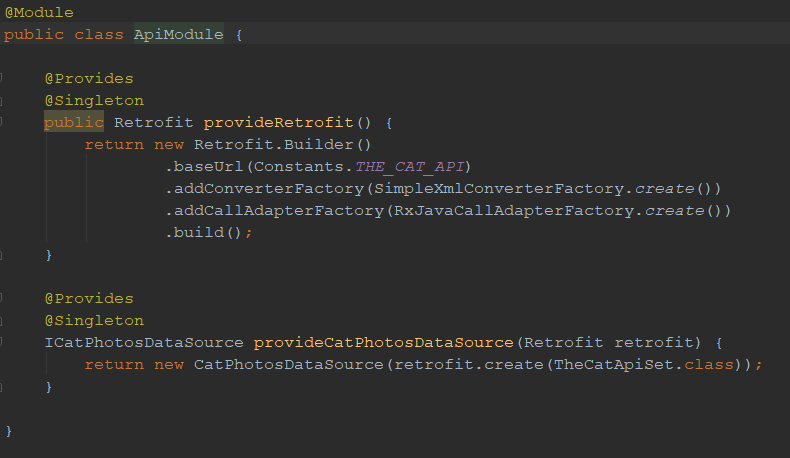
1. component – содержит код для последующего внедрения компонентов приложения.

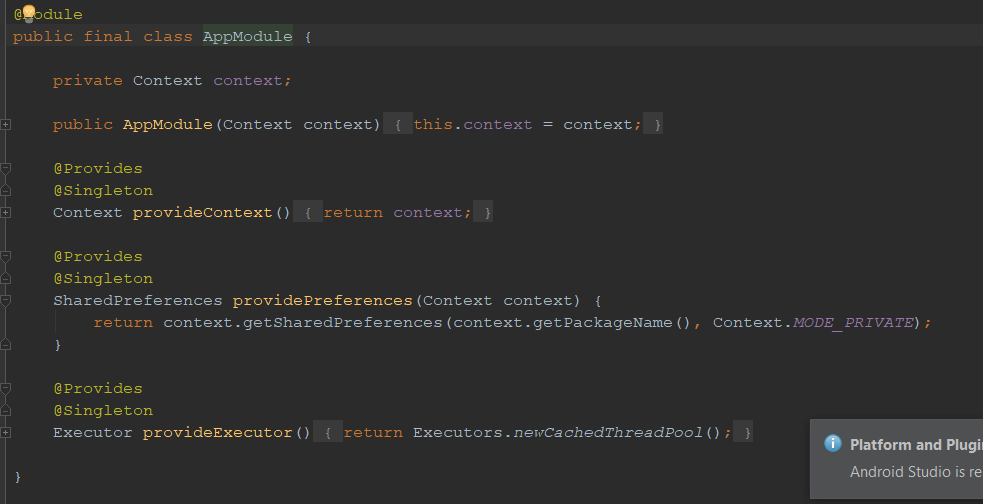


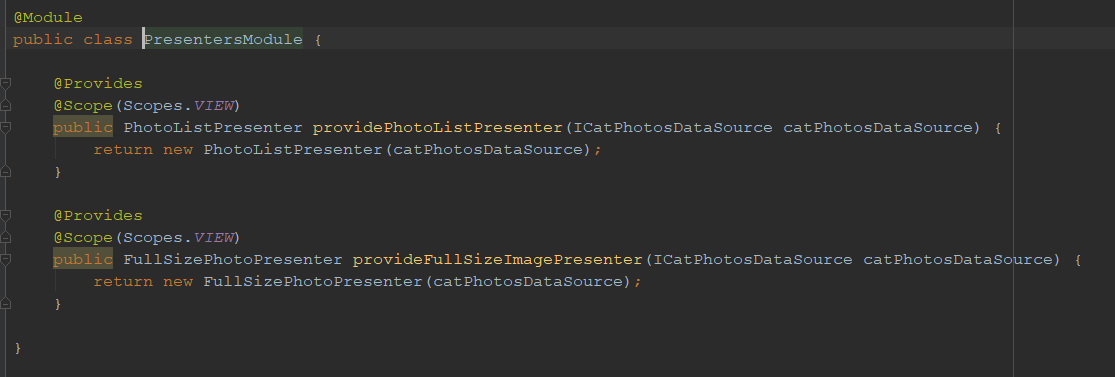




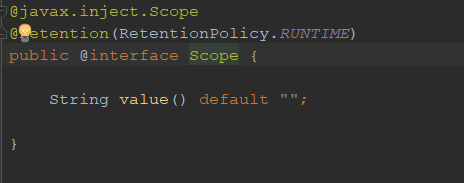
1. module – содержит код для последующего внедрения модулей приложения.

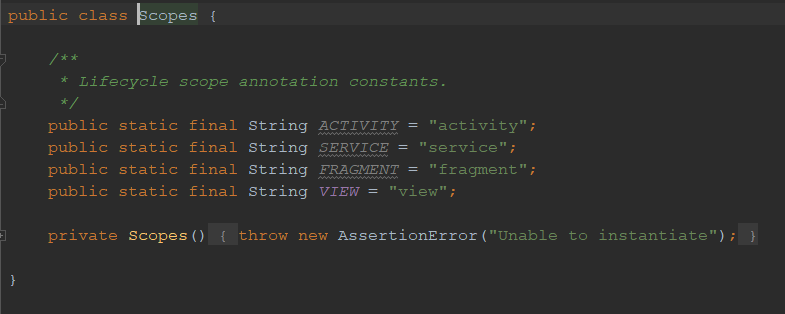






1. scopes – предоставляют возможность создания глобальных и “локальных синглтонов”





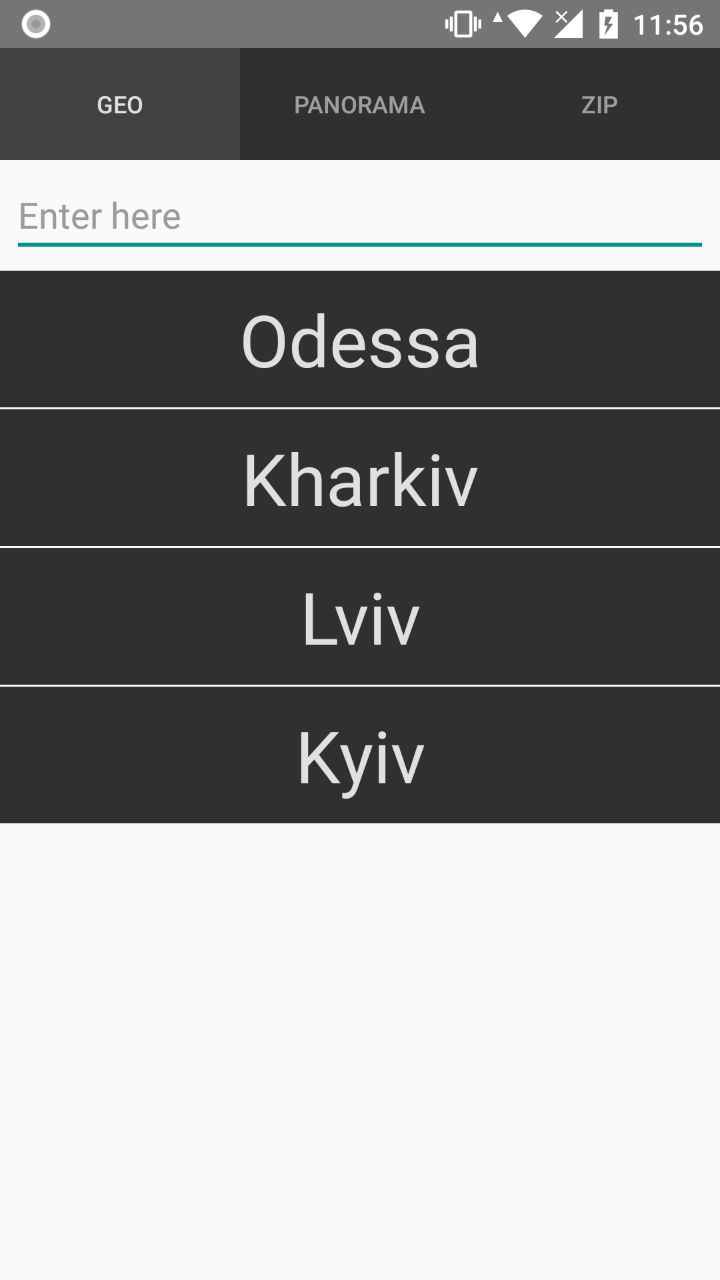
Приложение оперирует следующими классами для предоставления типов данных:

1. GeoResponse– общий класс для хранения всего дерева тела ответа.
2. Geoname – класс, который отвечает за представление данных о городе

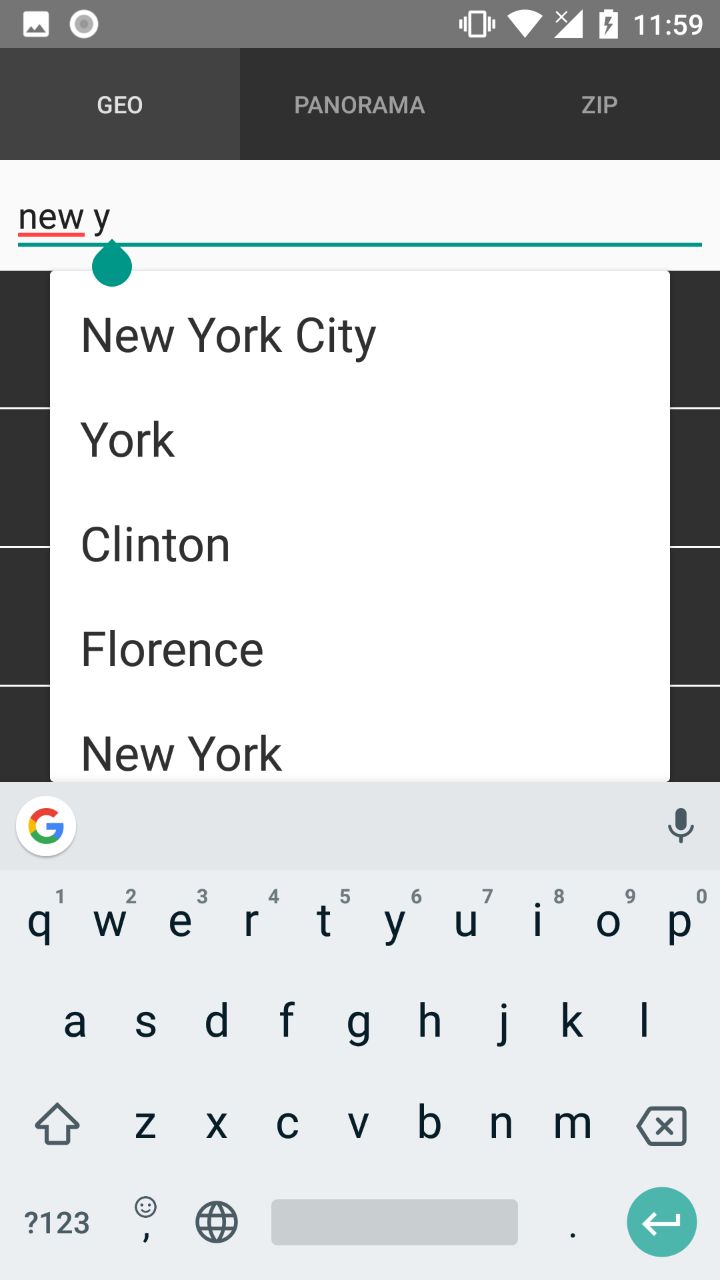
# Код програми

Исходный код програмы расположен в репозитории

# Приклади виконання програми



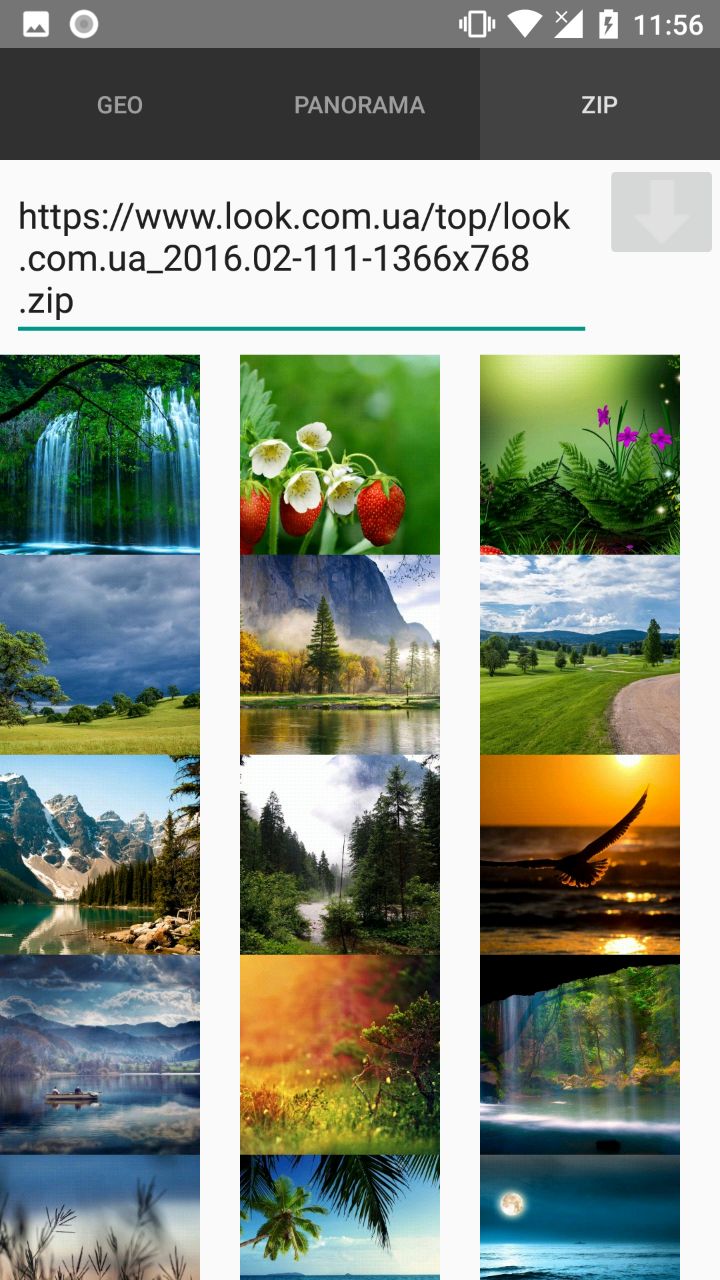
Пример окна со списком городов который хранится в базе данных



Пример экрана поиска городов



Пример отображения изображения с возможностью регулировать отображаемый участок с помощью акселерометра



Пример окна с отображением полной фотографии

# Висновок

В результате проекта было создано мобильное приложение, которое позволяет пользователям увидеть возможности реализации мобильного приложения с использованием базы данных, акселерометра и загрузка с отображением изображений. При написании проекта были использованы последние технологии в отрасли, которые были описаны в соответствующем разделе этого документа. Использована структура проекта, рекомендуемая Google.