Министерство образования Новосибирской области

ГБПОУ НСО «Новосибирский авиационный технический колледж имени Б.С.Галущака»

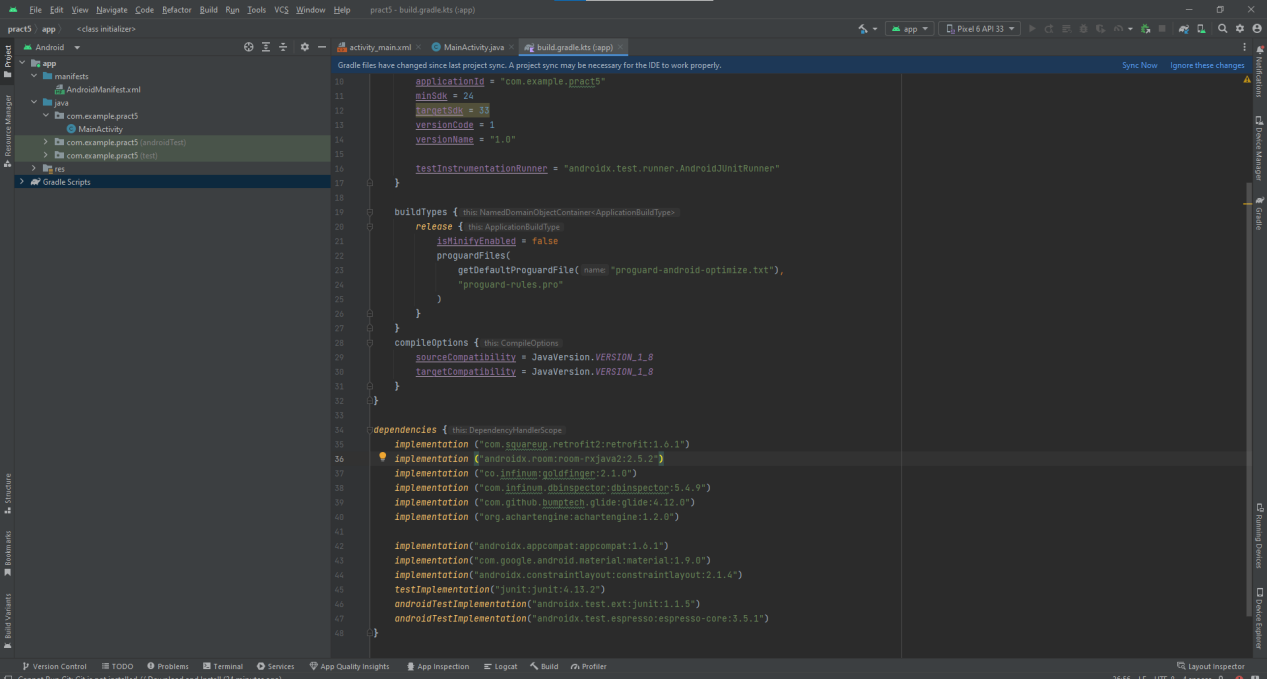
**Добавление библиотек к проекту**

Самостоятельная работа 5

Выполнил: Чайка Е.В.

Проверила: Климова И.С.

2023



* **Retrofit;**

Retrofit — это типобезопасный HTTP-клиент, позволяющий представить ваш REST API в виде интерфейса. Вы сможете управлять телами, заголовками, параметрами API-запросов и многим другим с помощью аннотаций, что сделает этот процесс простым и понятным. Retrofit также позволяет выполнять синхронные и асинхронные вызовы API.

interface ApiService {

@GET("movie/{id}")

fun getMovieDetails(@Path("id") id: String) : Call<MovieDetails>

}

Ко всему прочему, Retrofit предоставляет отдельный Rx-модуль. Если вы используете Rx, этот модуль вернёт Observable при обращении к API, что позволит связать его с остальной частью вашего приложения. И это лишь некоторые из множества причин, по которым мы по-прежнему любим и используем Retrofit.

* **AChartEngine;**

AChartEngine — мощная и гибкая библиотека для построения графиков и диаграмм в приложениях на языке программирования Java. Она предоставляет разработчикам широкий спектр возможностей для создания и настройки различных типов графиков, включая линейные, столбчатые, круговые, а также графики со сложными функциями и многое другое.

* **Glide;**

Glide — это библиотека загрузки изображений, которая предоставляет качественный API, позволяющий изменять изображение как вам угодно.

GlideApp.with(context)

.load(url)

.centerCrop()

.placeholder(R.drawable.ic\_placeholder)

.into(imageView)

Библиотека позволит вам легко загрузить удалённое изображение в ImageView, определить стабовые изображения, кэшировать и масштабировать изображения и т.д. Просто попробуйте проделать всё это без помощи Glide, и вы сразу поймёте, почему он находится в нашем списке ключевых библиотек. Glide даже поддерживает некоторые распространённые изменения по умолчанию, например, создание изображения круглой формы.

* **Room;**

Room — это официальный ORM для Android, и этот статус оправдывается рядом причин. Эта библиотека отличается удобным API, похожим на Retrofit. Room также в значительной степени опирается на аннотации и стандартный SQL-синтаксис.

@Dao

interface MovieDao {

@Query("SELECT details FROM movies WHERE id = :id")

fun getMovieDetails(id: String): MovieDetails

}

Кроме того, Room по умолчанию включает в себя поддержку Rx и LiveData, поэтому вы сможете использовать его, как вам захочется. Основное преимущество Room перед различными ORM — простота. Многие ORM имеют гораздо более сложный API, для использования которого вам потребуется тщательно изучить всю документацию. Благодаря стандартному SQL-синтаксису Room предельно доступен для понимания, что позволяет приступить к работе сразу, не тратя много времени на чтение документации.

* **Goldfinger;**

Goldfinger — упрощает реализацию аутентификации по отпечаткам пальцев;

DBInspector — просматривает и обеспечивает доступ к базам данных приложения;

* **DBInspector.**

DbInspector предоставляет простой способ просмотра содержимого встроенной базы данных в целях отладки

**Контрольные вопросы**

**1. Классификация библиотек по их назначению и возможности их подключения может быть следующей:**

- Библиотеки для работы с графическими элементами: такие библиотеки предоставляют возможности для создания и манипулирования графическими объектами, такими как фигуры, линии, цвета и т.д. Примером такой библиотеки может быть библиотека OpenGL.

- Библиотеки для работы с сетью: эти библиотеки предоставляют функциональность для работы с сетевыми протоколами, отправки и получения данных по сети и другие связанные задачи. Примером такой библиотеки может быть библиотека Retrofit.

- Библиотеки для работы с базами данных: эти библиотеки предоставляют удобный интерфейс для работы с базами данных, включая выполнение запросов, манипуляцию данными и другие операции. Примером такой библиотеки может быть библиотека Room.

- Библиотеки для работы с графиками: эти библиотеки предоставляют функциональность для создания и отображения графиков различных типов, таких как линейные, круговые, столбчатые и т.д. Примером такой библиотеки может быть AChartEngine.

**2**. **Безопасность использования библиотек** зависит от нескольких факторов, включая их качество, актуальность, авторизацию и проверку безопасности кода. Важно выбирать библиотеки, которые имеют активное сообщество разработчиков и регулярно обновляются для исправления уязвимостей и предоставления новых функций.

Также рекомендуется проверять исходный код библиотеки или доверять ее разработчикам, чтобы убедиться в отсутствии скрытых уязвимостей или злонамеренного кода.

**3. Для построения графиков с использованием библиотеки AChartEngine необходимо выполнить следующие шаги:**

- Подключить библиотеку AChartEngine к проекту. Это можно сделать путем добавления зависимости в файле конфигурации проекта (например, build.gradle для проектов на языке Java).

- Создать экземпляр класса ChartFactory, который предоставляет методы для создания различных типов графиков.

- Создать данные для графика, используя классы из пакета org.achartengine.model. Например, можно создать экземпляр класса XYSeries для построения линейного графика.

- Создать настройки графика, используя классы из пакета org.achartengine.renderer. Например, можно настроить цвета, шрифты и другие параметры отображения графика.

- Создать экземпляр класса GraphicalView, который представляет собой виджет для отображения графика.

- Добавить созданный GraphicalView на нужный макет или контейнер в пользовательском интерфейсе.

- Отобразить график, вызвав метод draw() для экземпляра GraphicalView.