Министерство образования Воронежской области

Государственное бюджетное профессиональное образовательное учреждение

Воронежской области «Борисоглебский техникум промышленных

и информационных технологий»

**ОТЧЕТ**

**по учебной практике**

**УП.03.01 Практика по сопровождению и продвижению ПО отраслевой направленности**

|  |  |
| --- | --- |
| Выполнил студент \_\_4\_\_ курса  спец. 09.02.05 «Прикладная информатика (по отраслям)»  Группа 4.2ПИ\_\_  Горст Яна Викторовна  (ФИО)  Дата сдачи \_\_13.04.2024 г.\_\_\_\_ | Проверил  преподаватель  Пеньков А.С.  (ФИО)  Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Борисоглебск

**2023 –2024 учебный год**

Оглавление

[Тема 1. Разрешение проблем совместимости программного обеспечения отраслевой направленности 3](#_Toc163921868)

[Тема 2.Осуществление продвижения и презентации программного обеспечения отраслевой направленности. 7](#_Toc163921869)

[Тема 3.Работа с системами управления взаимоотношениями с клиентами: создание базы данных клиентов. 12](#_Toc163921870)

[Тема 4.Работа с системами управления взаимоотношениями с клиентами: формирование статистических и аналитических отчетов по продажам. 13](#_Toc163921871)

[Тема 5.Проведение обслуживания, тестовых проверок, настройки программного обеспечения отраслевой направленности. 15](#_Toc163921872)

[Тема 6. Проектирование мобильного приложения 16](#_Toc163921873)

[Тема 7. Создание макета приложения 17](#_Toc163921874)

[Тема 8. Создание элементов пользовательского интерфейса 19](#_Toc163921875)

[Тема 9. Работа со списочными элементами интерфейса 23](#_Toc163921876)

[Тема 10. Использование View и ViewGroup 24](#_Toc163921877)

[Тема 11. Обработка событий в Android Тема 12. Архитектура: MVVM 25](#_Toc163921878)

[Тема 13. Отображение списков: RecyclerView Тема 14. CRUD: списки, добавление, удаление, изменение 26](#_Toc163921879)

[Тема 15. Material Design Тема 16. Intents Тема 17. Хранение данных 28](#_Toc163921880)

## **Тема 1. Разрешение проблем совместимости программного обеспечения отраслевой направленности**

Проект на GitHub — https://github.com/GorstYana/practice/tree/zadaniye1

**Задача**

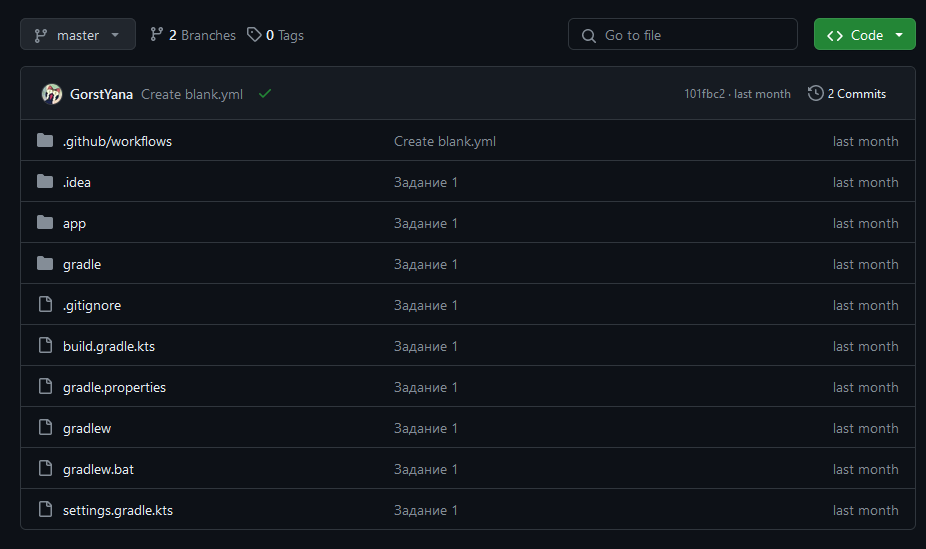
Создайте проект с текстовой надписью на экране Мы учимся в БТПИТ! вместо Hello, World



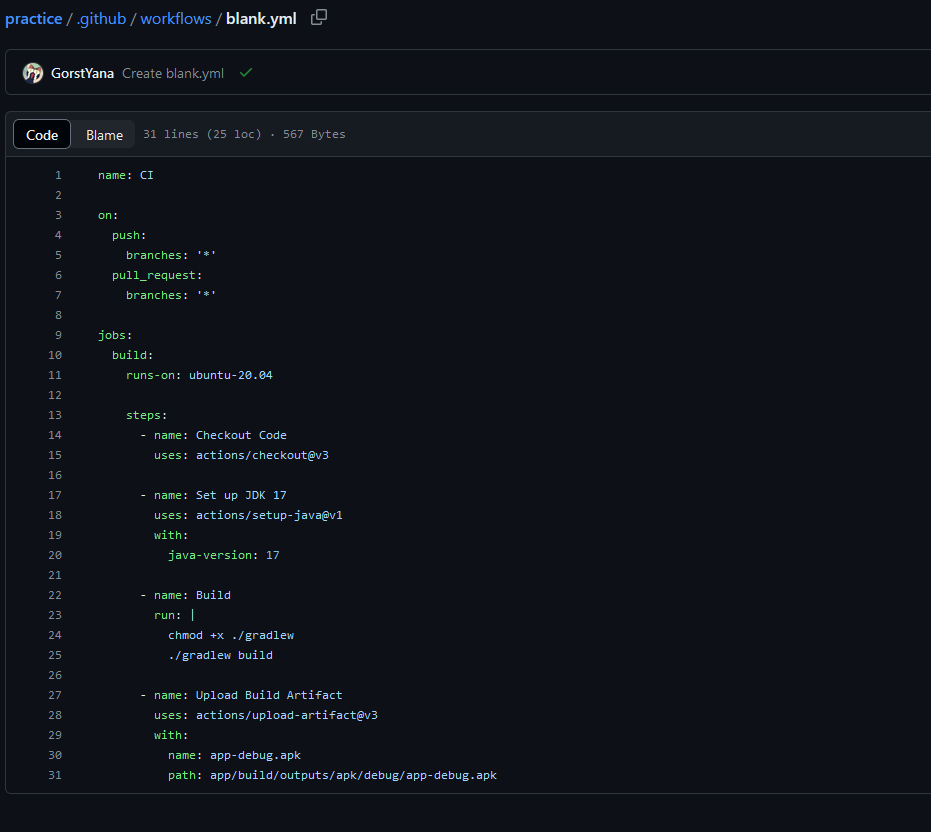
Настройте GitHub Actions и отправьте ссылку на репозиторий в личном кабинете студента.

Описание шагов выполнения

1. Публикуете свой проект на GitHub.



2. Переходите на вкладку Actions и выбираете любой: выбрала basic workflow



3. Заменяете содержимое на следующее (о предназначении читайте в разделе «Справка» выше):

name: CI

on:

push:

branches: '\*'

pull\_request:

branches: '\*'

jobs:

build:

runs-on: ubuntu-20.04

steps:

- name: Checkout Code

uses: actions/checkout@v3

- name: Set up JDK 17

uses: actions/setup-java@v1

with:

java-version: 17

- name: Build

run: |

chmod +x ./gradlew

./gradlew build

- name: Upload Build Artifact

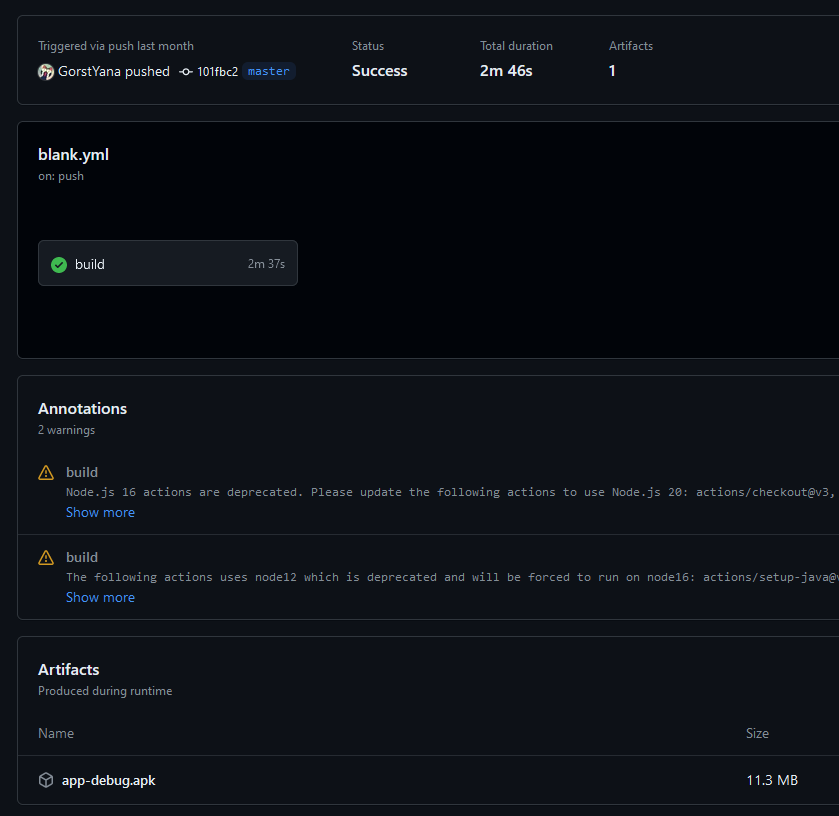
uses: actions/upload-artifact@v3

with:

name: app-debug.apk

path: app/build/outputs/apk/debug/app-debug.apk

4. Убедитесь, что сборка прошла успешно и в артефактах появился app-debug.apk:



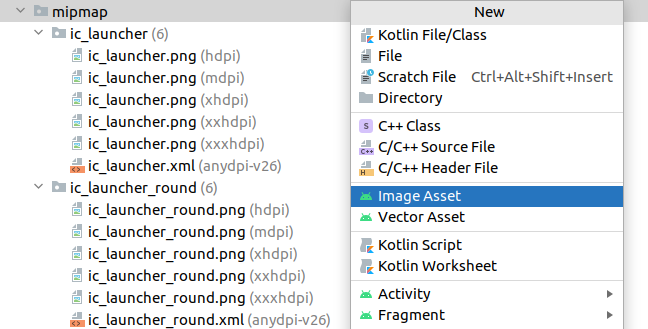
## **Тема 2.Осуществление продвижения и презентации программного обеспечения отраслевой направленности.**

Проект на GitHub — https://github.com/GorstYana/practice/tree/zadaniye2

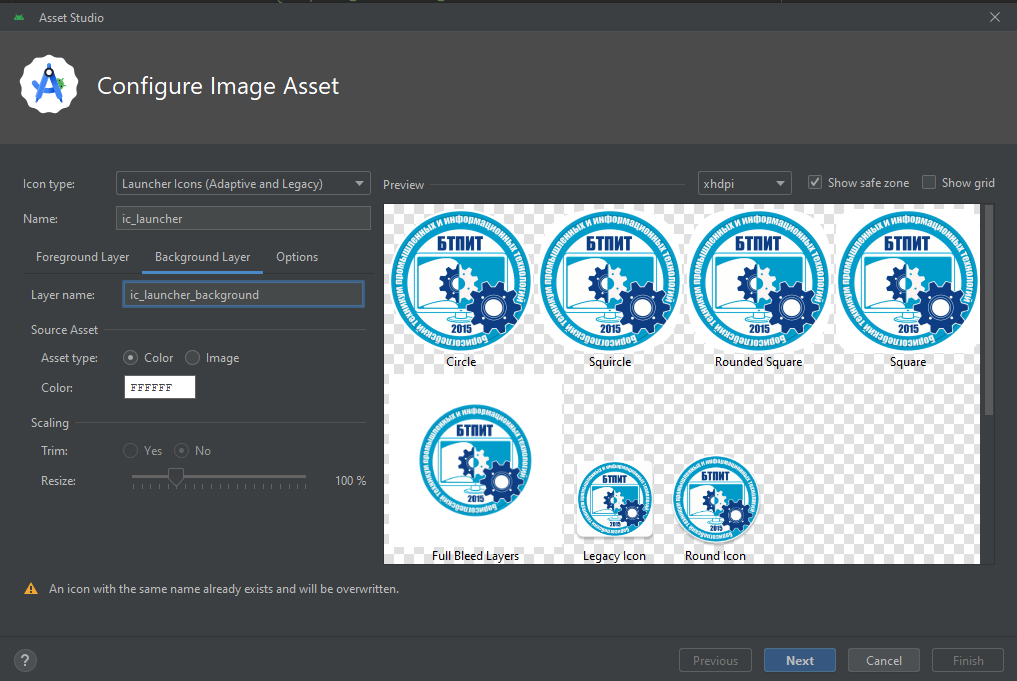
**Задача Launcher Icon**

Замените иконку вашего приложения из предыдущего ДЗ на [логотип БТПИТ](https://disk.yandex.ru/i/lQXoBPOFdtO5eQ).

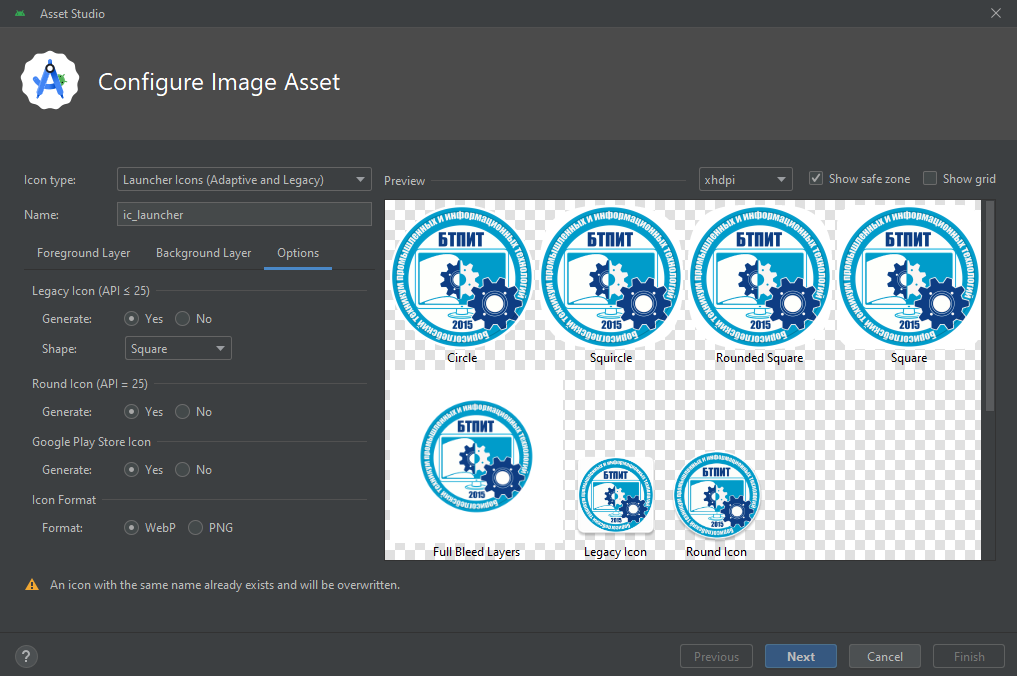
1. Кликните правой кнопкой мыши (или Alt + Insert) на каталоге mipmap и выберите пункт Image Asset:

[](https://github.com/netology-code/and2-homeworks/blob/master/02_resources/pic/asset01.png)

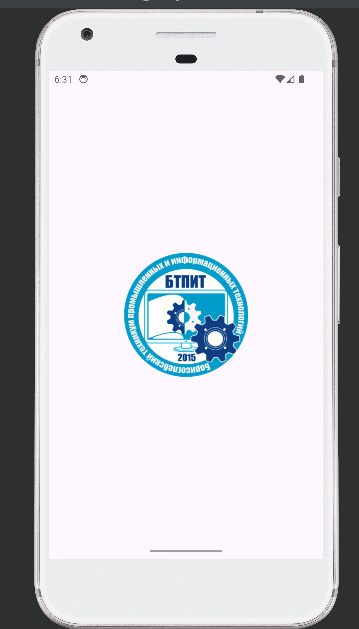
3. Перейдите на вкладку Background Layer (1), выберите Asset Type — Color (2) и поставьте цвет — FFFFFF (3):



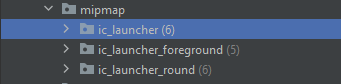
4. Перейдите на вкладку Options. По умолчанию настройки выставлены в генерацию иконок, т. е. будут сгенерированы изображения для тех версий Android, которые не поддерживают адаптивные иконки:



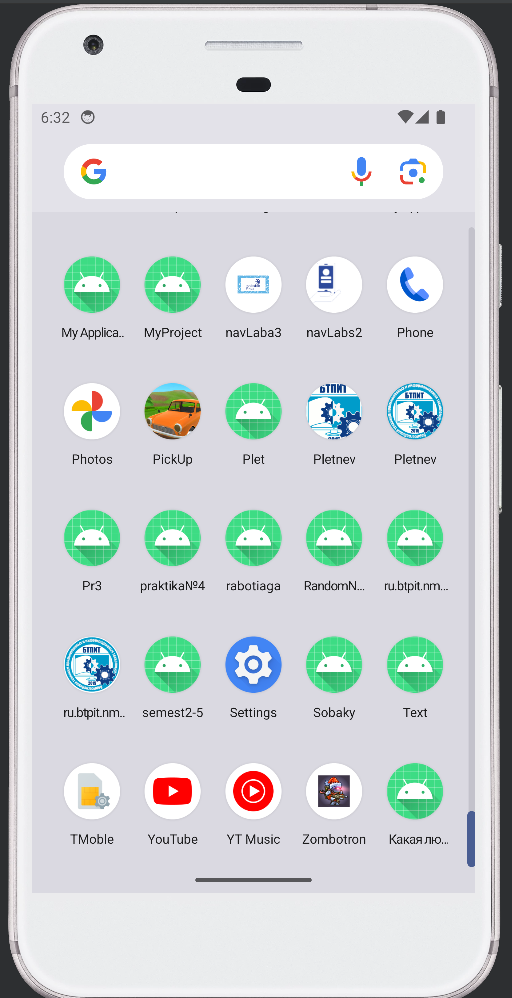
5. Подтвердите генерацию файлов, нажав Finish:



6. Удалите старые файлы (с иконкой Android):



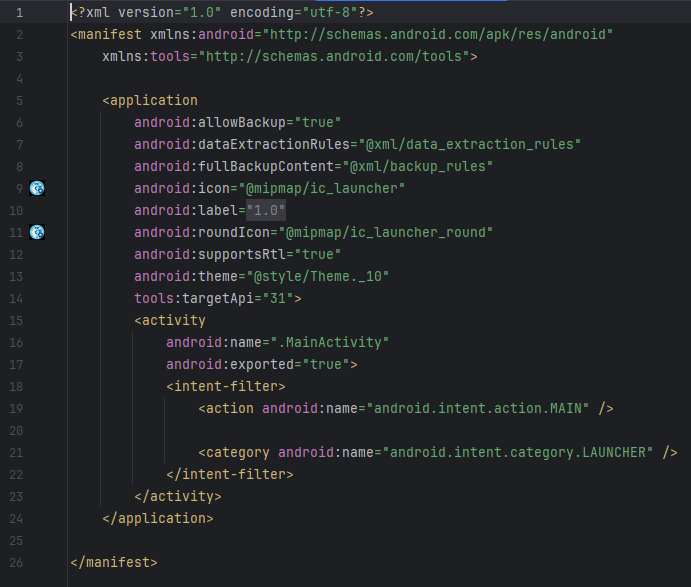
7. Запустите ваше приложение в эмуляторе и убедитесь, что иконка приложения изменилась.



Если при сборке возникают ошибки, нажмите два раза Ctrl, выполните gradlew clean и соберите заново

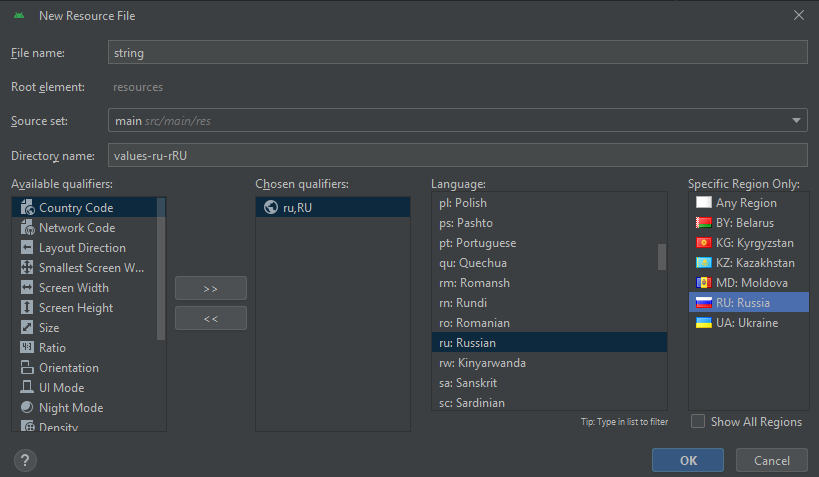
Подсказка:

Иконка указывается в манифесте (атрибуты android:icon и android:roundIcon)



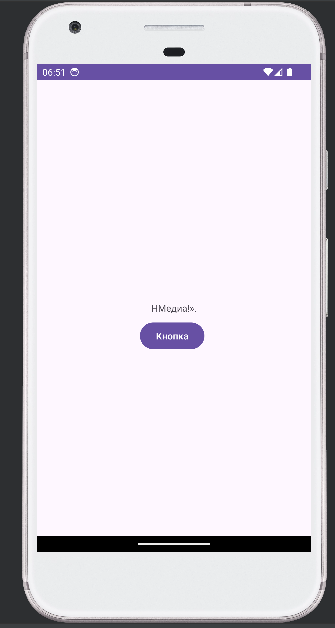
**Задача Translations**

1. Создайте файл переводов по аналогии с лекцией.

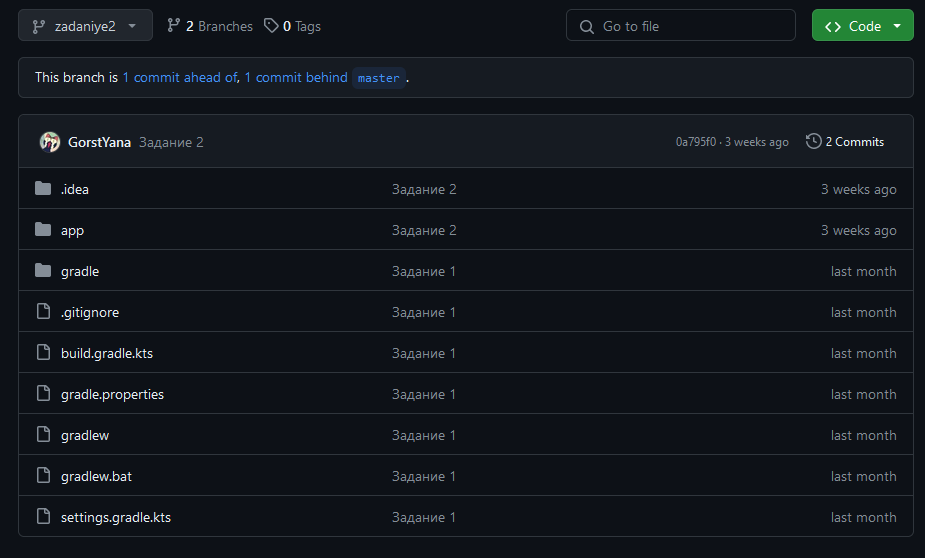




1. Убедитесь, что при изменении языка перевод отображается корректно.



Опубликуйте изменения в вашем проекте на GitHub. Убедитесь, что apk собирается с помощью GitHub Actions и при установке в эмуляторе приложение работает корректно.

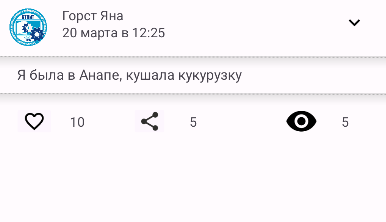


## **Тема 3.Работа с системами управления взаимоотношениями с клиентами: создание базы данных клиентов.**

**Задача Layout**

Проект на GitHub — https://github.com/GorstYana/practice/tree/zadaniye3

Сделайте разметку в соответствии с легендой. Иконки берите из стандартного набора.

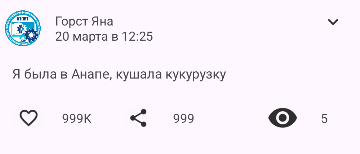


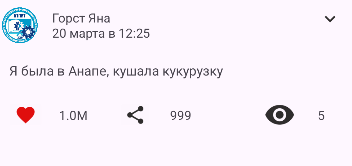
## **Тема 4.Работа с системами управления взаимоотношениями с клиентами: формирование статистических и аналитических отчетов по продажам.**

Проект на GitHub — https://github.com/GorstYana/practice/tree/zadaniye4

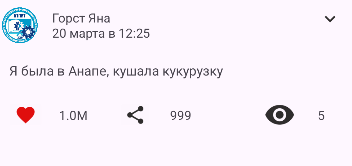
Задача Like, Share

1. При клике на like должна меняться не только картинка, но и число рядом с ней: лайкаете — увеличивается на 1, дизлайкаете — уменьшается на 1.





1. При клике на share должно увеличиваться число рядом: 10 раз нажали на share — +10.

 Изображение выглядит как текст, снимок экрана, Шрифт

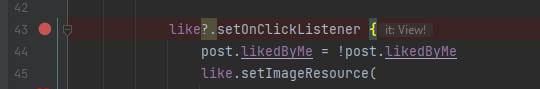
Автоматически созданное описание

1. Если количество лайков, share или просмотров перевалило за 999, должно отображаться 1K и т. д., а не 1 000. Предыдущие функции должны работать: если у поста было 999 лайков и вы нажали like, то должно стать 1К, если убрали лайк, то снова 999.



Задача Parent Child

1. Установите обработчики
2. Поставьте внутрь обработчиков точки останова.



1. Запустите приложение в режиме отладки.
2. Кликните на кнопку Like, на кнопку с тремя точками (на ней пока нет обработчика), на текст, на аватар.
3. Установите обработчик OnClickListener на аватар и кликните по нему снова.

В качестве результата напишите ответы на следующие вопросы:

1. Какой из обработчиков сработал при клике на кнопку Like?

like?.setOnClickListener

1. Сработал ли обработчик на binding.root при клике на кнопку с тремя точками?

нет

1. Сработал ли обработчик на binding.root при клике на текст?

нет

1. Сработал ли обработчик на binding.root при клике на аватар до установки на avatar собственного обработчика?

нет

1. Сработал ли обработчик на binding.root при клике на аватар после установки на avatar собственного обработчика?

нет

Попробуйте выявить закономерность: когда срабатывает обработчик на контейнере, а когда нет.

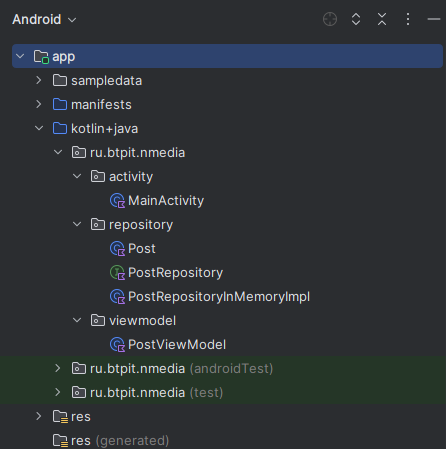
## **Тема 5.Проведение обслуживания, тестовых проверок, настройки программного обеспечения отраслевой направленности.**

Проект на GitHub — https://github.com/GorstYana/practice/tree/zadaniye5

**Задача MVVM**

Вам нужно переделать предыдущую реализацию на MVVM. Всё, что работало до этого: увеличение количества лайков, share и 1K — должно продолжать работать.

Опубликуйте изменения в вашем проекте на GitHub в ветке mvvm. Убедитесь, что apk собирается с помощью GitHub Actions и при установке в эмуляторе приложение работает корректно.





## **Тема 6. Проектирование мобильного приложения**

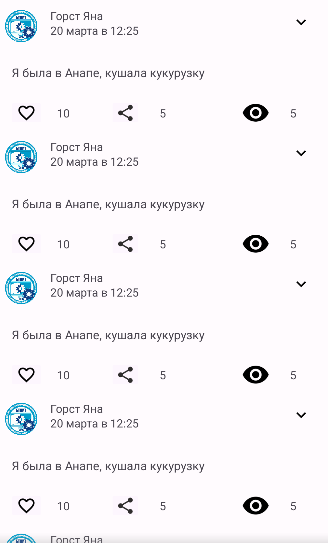
Проект на GitHub — https://github.com/GorstYana/practice/tree/zadaniye6

**Задача. RecyclerView**

Добавьте в свой проект реализацию отображения списков на базе RecyclerView и ListAdapter.

В примерах на лекции у нас был только OnLikeListener. Сделайте также OnShareListener.

Опубликуйте изменения в вашем проекте на GitHub. Убедитесь, что apk собирается с помощью GitHub Actions и при установке в эмуляторе приложение работает корректно.



## **Тема 7. Создание макета приложения**

Проект на Figma - [Мобильные приложение соц-сети(Практика) – Figma](https://www.figma.com/file/MEtDbFS1hkaA7kzQixvWws/Мобильные-приложение-соц-сети(Практика)?type=design&node-id=0-1&mode=design&t=eUGmHQ6vwwjJ5htF-0)

Запуск приложения: Авторизация:

Изображение выглядит как розовый, рисунок, Пурпурный цвет, Графика

Автоматически созданное описание Изображение выглядит как текст, розовый, снимок экрана, Пурпурный цвет

Автоматически созданное описание

Лента: ПоискИзображение выглядит как снимок экрана, текст, млекопитающее

Автоматически созданное описание Изображение выглядит как снимок экрана, текст, Веб-сайт, дизайн

Автоматически созданное описание

Сообщения Чат

Изображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описание Изображение выглядит как текст, домашнее животное, Порода собаки, снимок экрана

Автоматически созданное описание

## **Тема 8. Создание элементов пользовательского интерфейса**

Задача CRUD

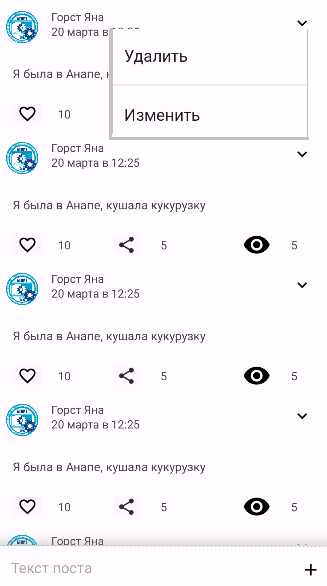
Добавьте реализацию CRUD в вашу версию приложения. Все функции, вроде лайков с отображением количества и share, должны по-прежнему работать.

Опубликуйте изменения в вашем проекте на GitHub. Убедитесь, что apk собирается с помощью GitHub Actions и при установке в эмуляторе приложение работает корректно.

В качестве результата пришлите ссылку на ваш GitHub-проект в личном кабинете студента на сайте [netology.ru](https://netology.ru/).

**Задача. Отмена редактирования**

Реализуйте отмену редактирования аналогично тому, как это сделано в Telegram:

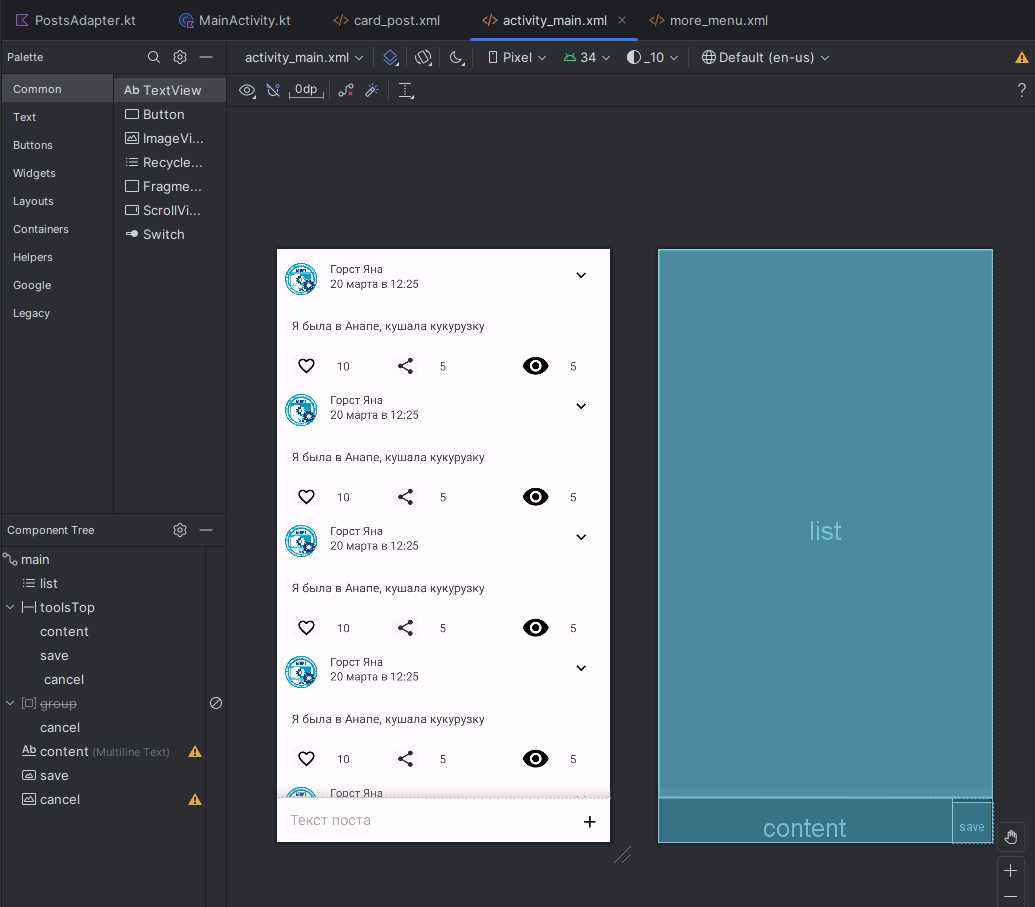
 Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

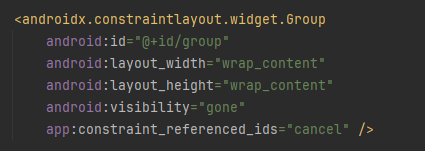
Достаточно с помощью ConstraintLayout сформировать соответствующую структуру над полем ввода поста.

Единовременно управлять отображением/скрытием сразу нескольких View можно с помощью объединения их [в виртуальную группу](https://developer.android.com/reference/androidx/constraintlayout/widget/Group).

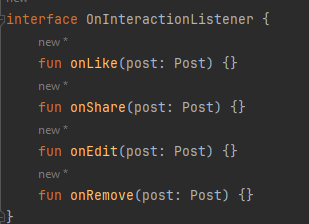
Скрыть можно с помощью:

group.visibility = View.GONE // перестаёт занимать место на экране

group.visibility = View.INVISIBLE // невидима, но занимает место на экране



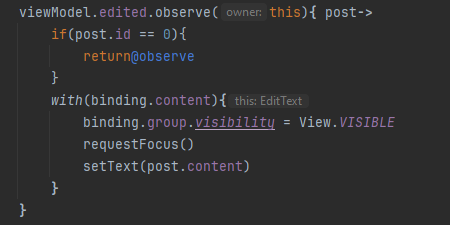
Заведем специальный интерфейс с базовыми функциями



Теперь нужно передать эти функции:



Обновление текста после вставки слов в edittext



Добавление



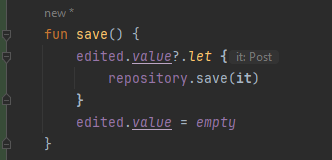
В Layout добавляем:



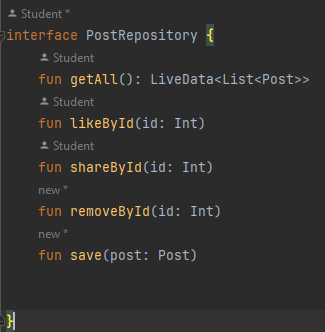
Нам нужен объект для заполнения в PostViewModel добавляем:



Функцию сохранения:



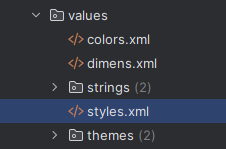
В repository

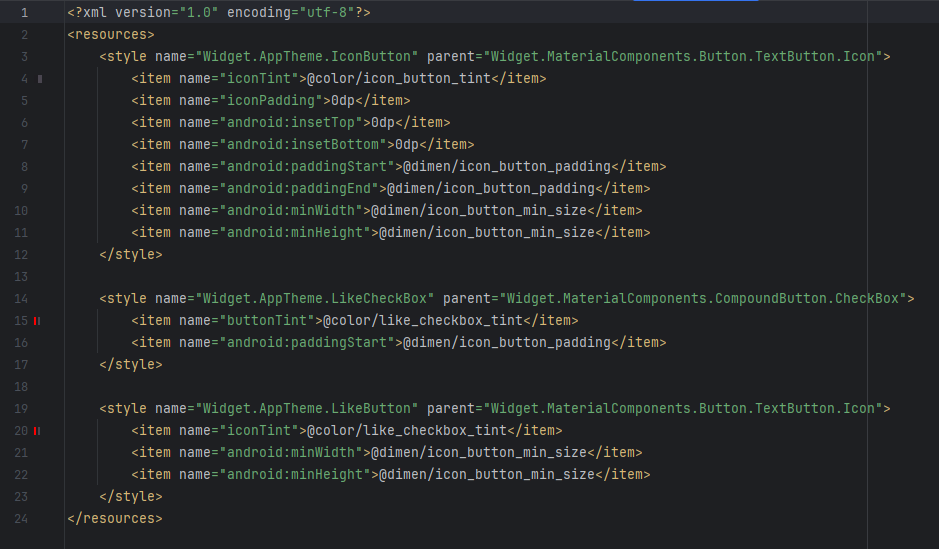


## **Тема 9. Работа со списочными элементами интерфейса**

Нужно стилизовать кнопки Share и Options — меню с тремя точками.

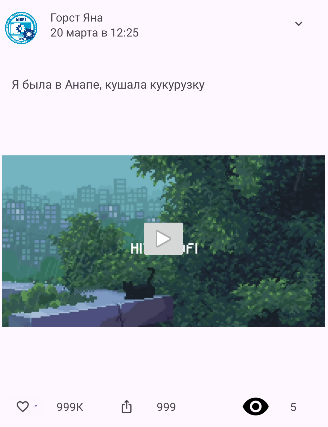
Текст можно задавать сразу через атрибуты кнопки (количество лайков, шеринга). Поэтому от TextView можно будет избавиться.



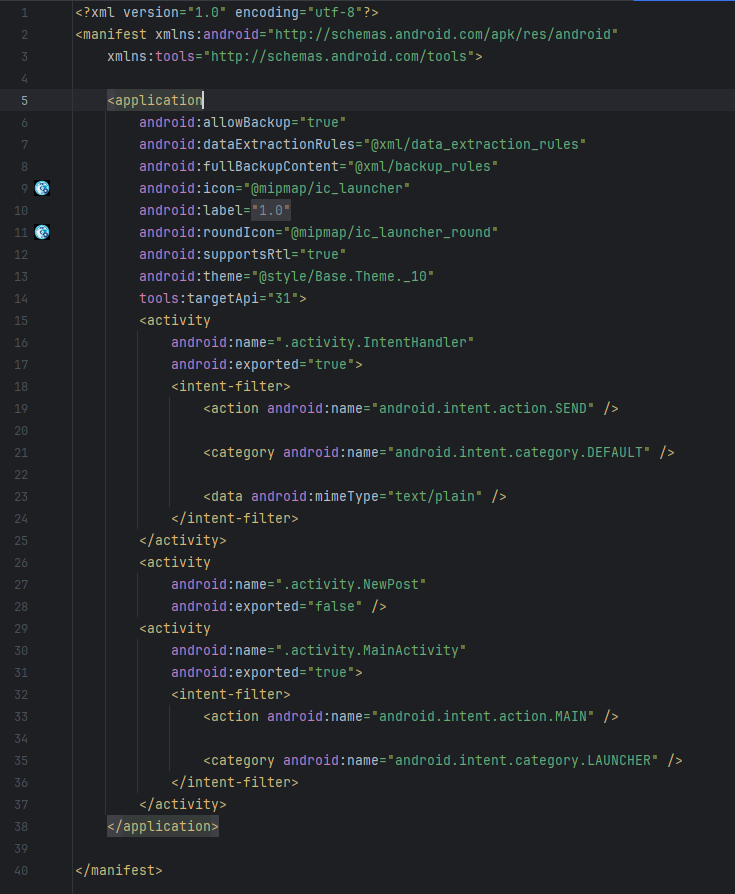


## **Тема 10. Использование View и ViewGroup**

Добавление видео в пост:



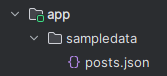
Так же нам нужно прописать Manifest:



## **Тема 11. Обработка событий в Android Тема 12. Архитектура: MVVM**

Задача

Сделайте альтернативную реализацию репозитория, которая работает с JSON-файлом в качестве постоянного хранилища вместо In Memory.

{

"data": [

{

"id": 1,

"author": "Горст Яна",

"authorAvatar": "@sample/posts\_avatars",

"published": "20 марта в 12:25",

"content": "Я была в Анапе, кушала кукурузку",

"likedByMe": true,

"likes": 100,

"sharedByMe": true,

"share": 100

}

]

}

В папке repository – добавим строчку - private val filename = "posts.json"

И создаем функцию sync с подвязкой к gson

private fun sync(){

with(prefs.edit()){

putString(key, gson.toJson(posts))

apply()

}

}

Так же нужно в import прописать строчку: import com.google.gson.Gson

## **Тема 13. Отображение списков: RecyclerView Тема 14. CRUD: списки, добавление, удаление, изменение**

Проект на GitHub — https://github.com/GorstYana/practice/tree/zadaniye11

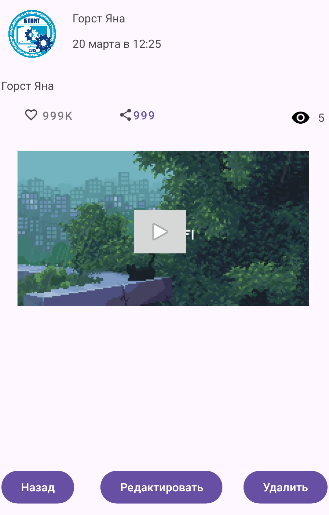
Задача

Сделайте так, чтобы при клике на элемент списка открывался фрагмент с конкретным постом. Если пользователь кликнул на меню, like, share или видео, перехода быть не должно.

Пока фрагмент с конкретным постом будет выглядеть так, как обычный пост из списка.

Операции для поста должны работать как из этого фрагмента, так и из фрагмента со списом постов.

С этого фрагмента можно попасть:



1. На фрагмент редактирования, если нажать на кнопку «Изменить».

2. На фрагмент со списком всех постов, если нажать на кнопку «Назад» (системную).

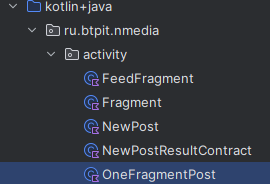
3. На фрагмент со списком всех постов, если нажать на кнопку «Удалить».

Для начала создадим xml файл-

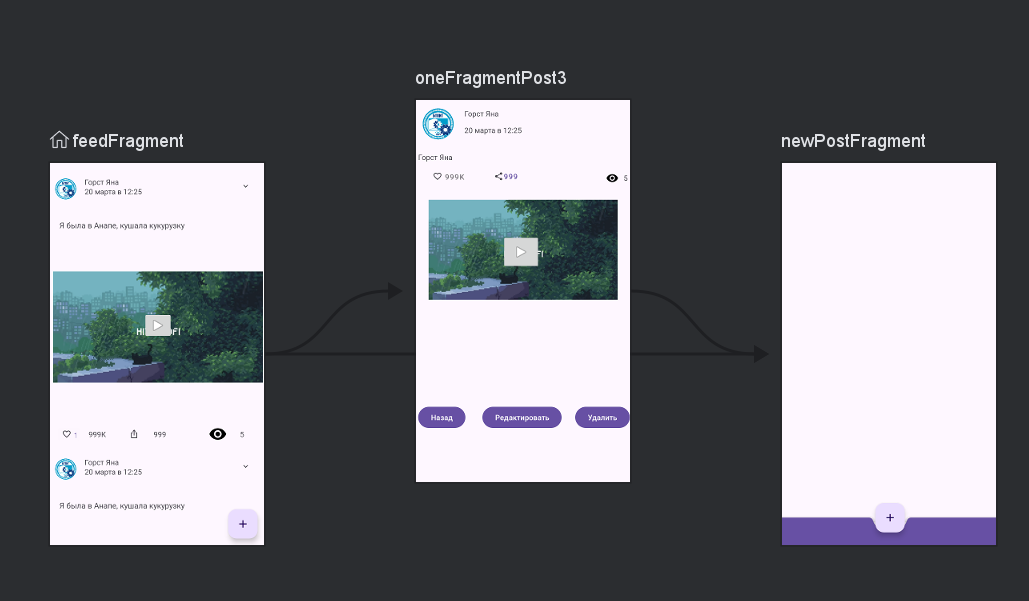


Слишком много строк кода





После чтобы работал переход нужно создать папку navigation и прописать три фрагмента:



## **Тема 15. Material Design Тема 16. Intents Тема 17. Хранение данных**

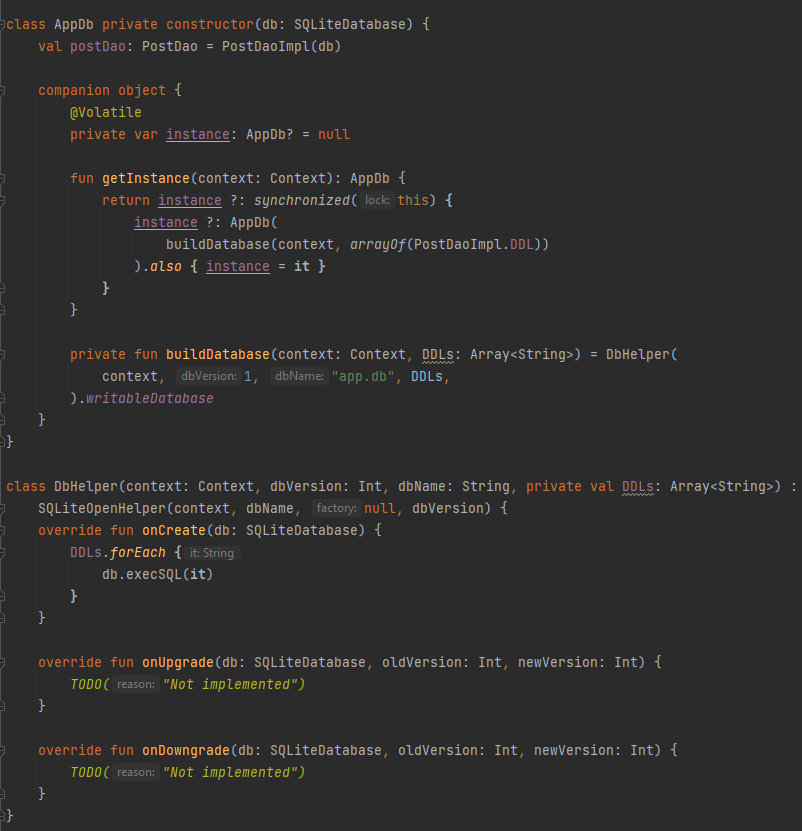
Проект на GitHub — https://github.com/GorstYana/practice/tree/zadaniye12

Задача

Первая задача достаточно простая: мигрируйте всё на SQLite, обязательно сохранив при этом работоспособность приложения.

Создадим папку DB

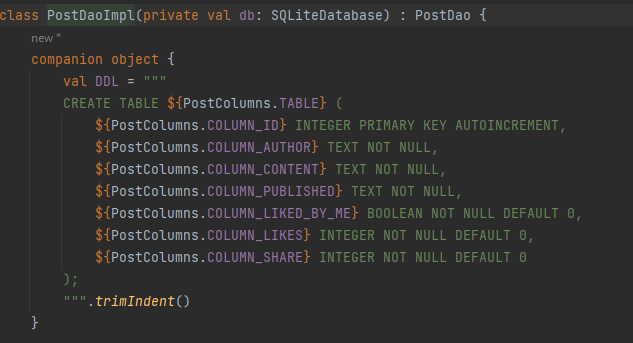


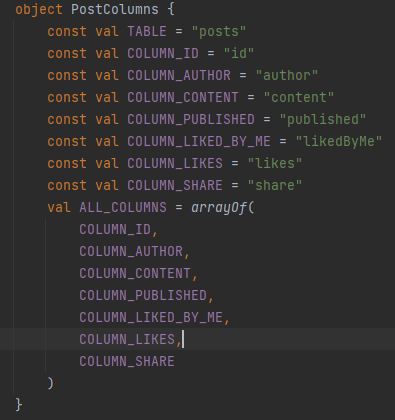


После создадим

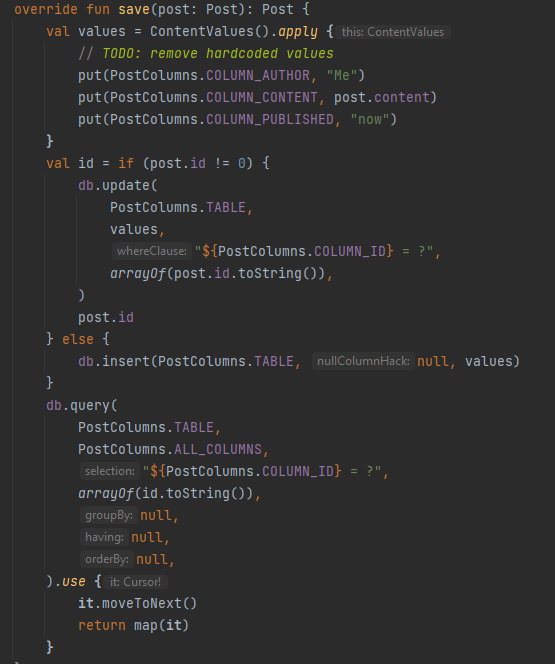


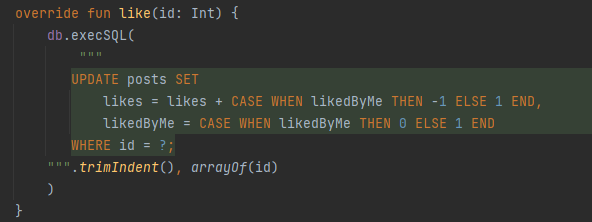
И пропишем код:

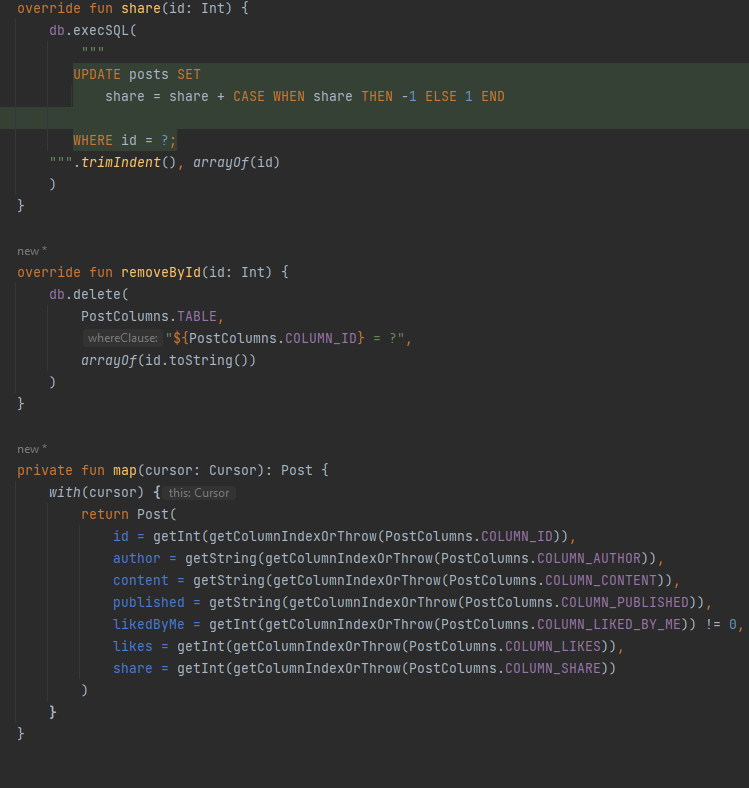




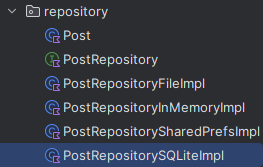


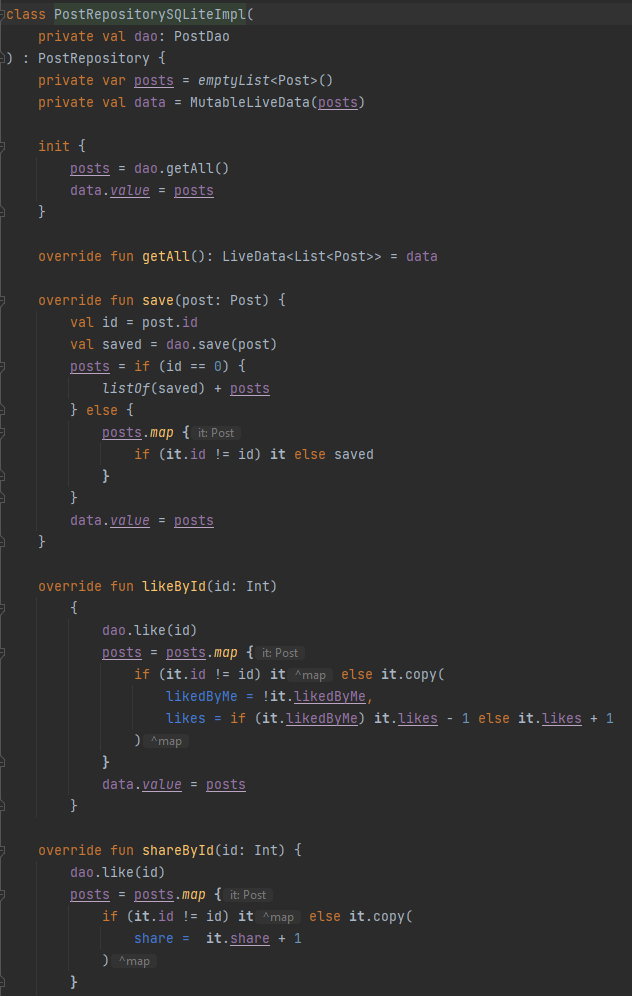


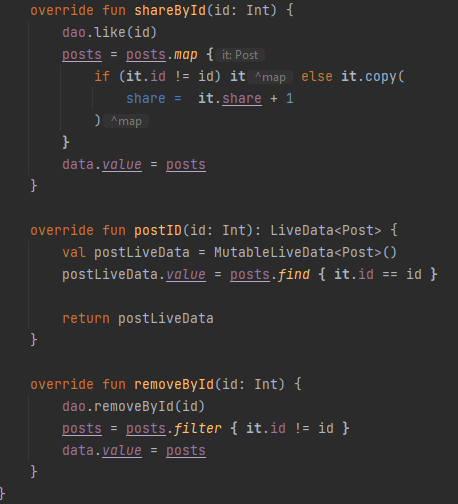




В папке repository создаем







И в postViewModel прописываем обязательную строчку

private val repository: PostRepository = PostRepositorySQLiteImpl(AppDb.getInstance(application).postDao)

После чего у нас создается БД в которой можно добавить пост и при выходе он сохранится.

До выхода После перезахода

