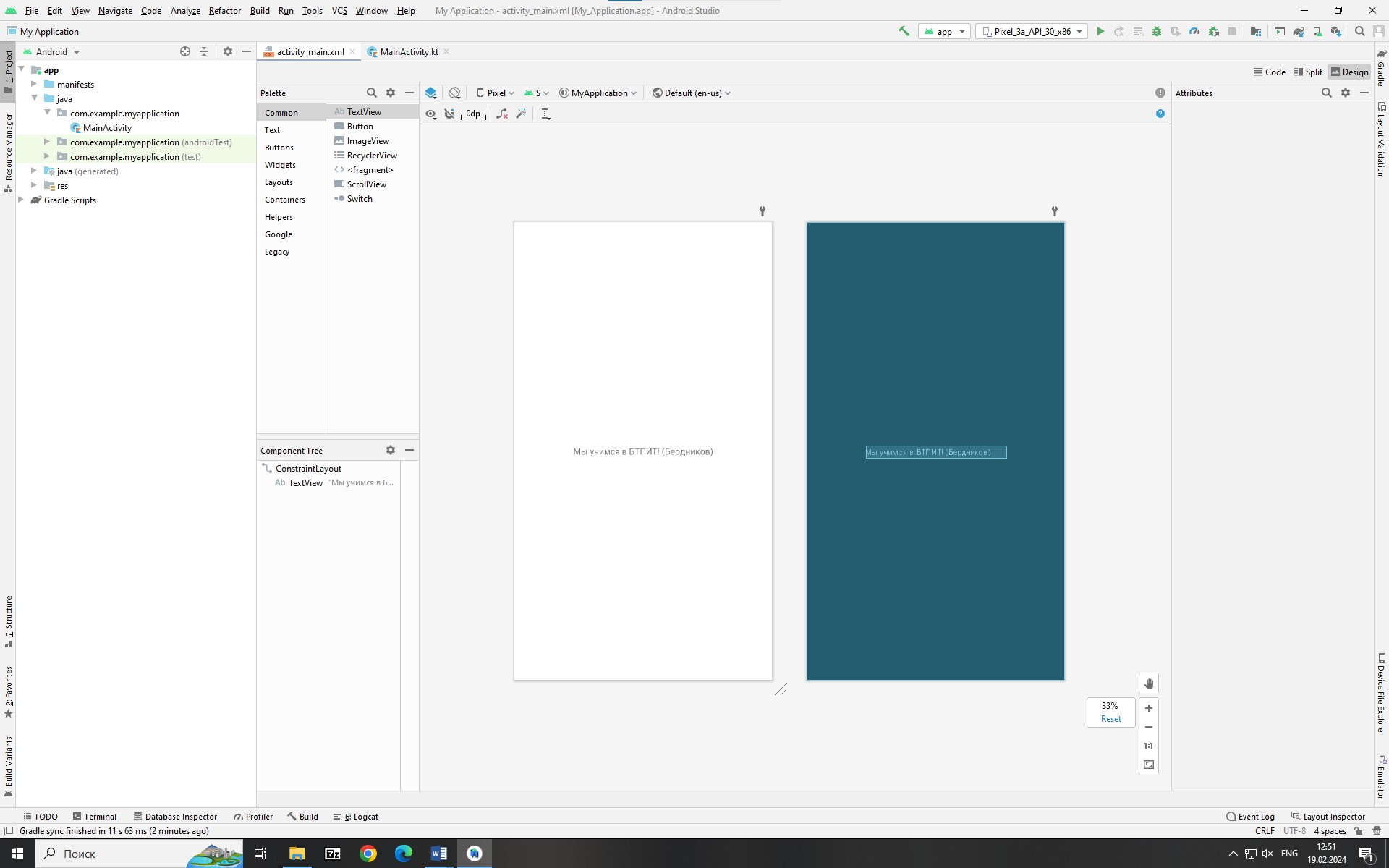
1 Занятие

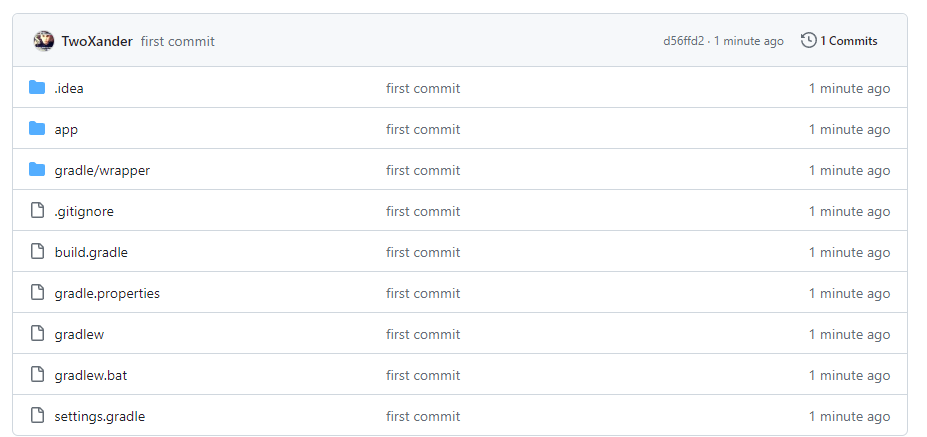
Ваша задача на первое занятие достаточно простая. По примеру из лекции создайте проект с текстовой надписью на экране Мы учимся в БТПИТ! вместо Hello, World



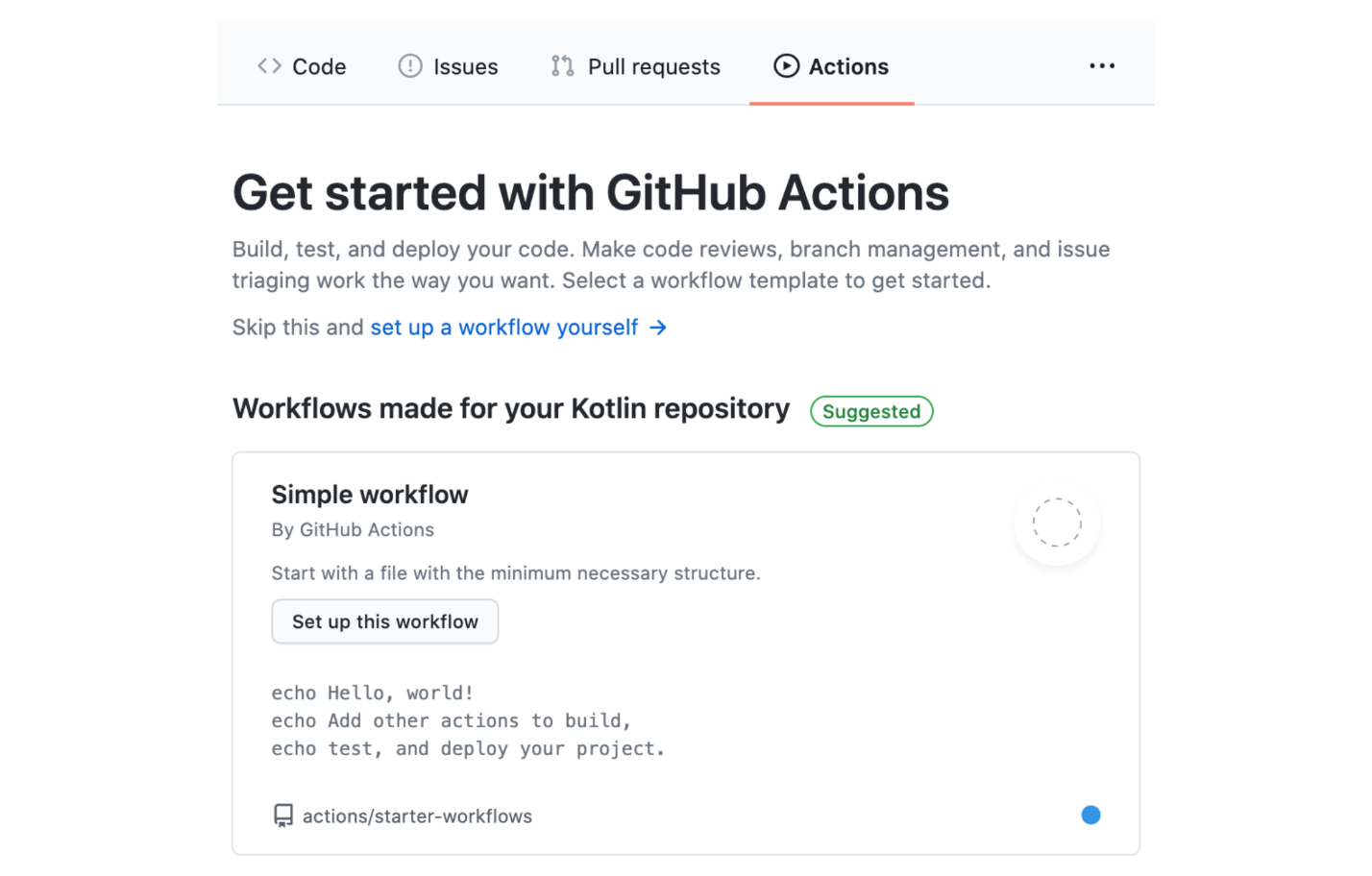
Настройте GitHub Actions и отправьте ссылку на репозиторий в личном кабинете студента.

Описание шагов выполнения

1. Публикуете свой проект на GitHub.



2. Переходите на вкладку Actions и выбираете любой:

[](https://github.com/netology-code/and2-homeworks/blob/master/01_firstapp/pic/actions.png)

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

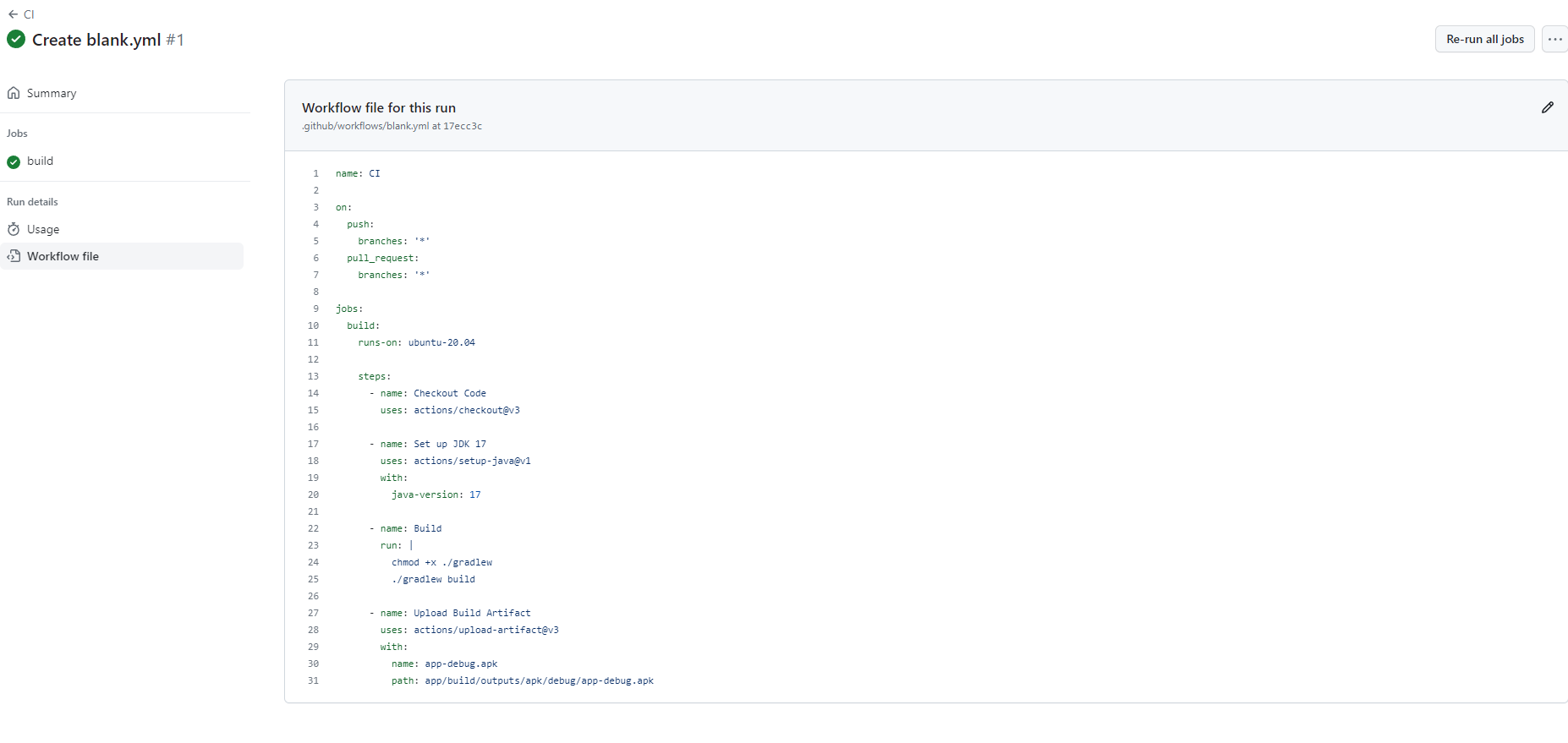
uses: actions/upload-artifact@v3

with:

name: app-debug.apk

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

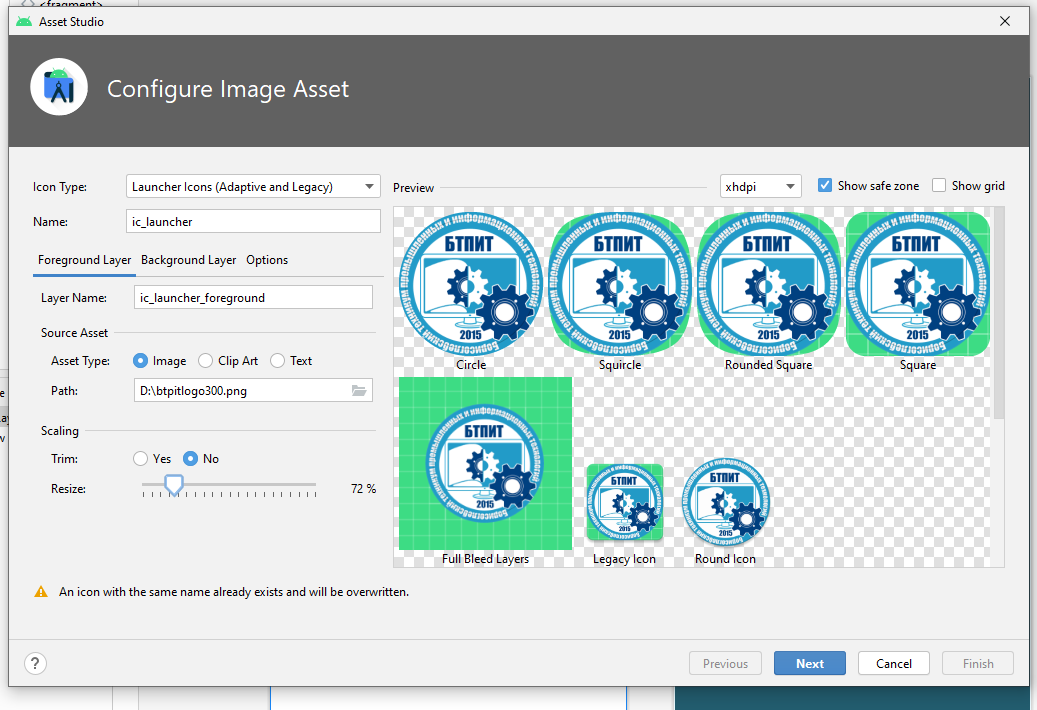
4. Убедитесь, что сборка прошла успешно и в артефактах появился app-debug.apk:



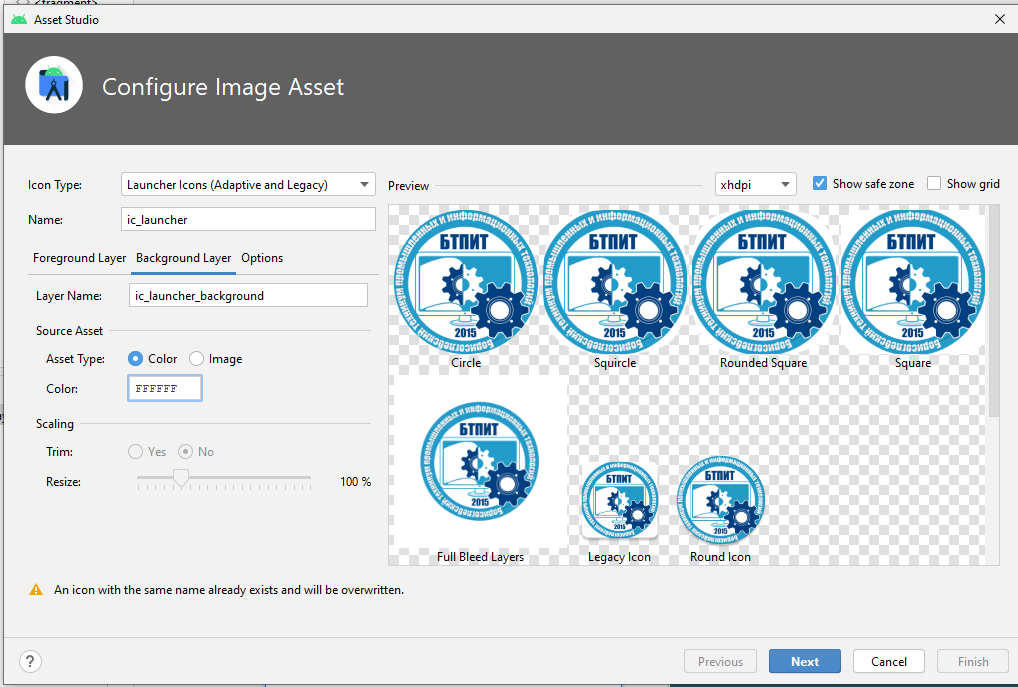
Замените иконку вашего приложения из предыдущего ДЗ на [логотип БТПИТ](https://disk.yandex.ru/i/lQXoBPOFdtO5eQ).

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

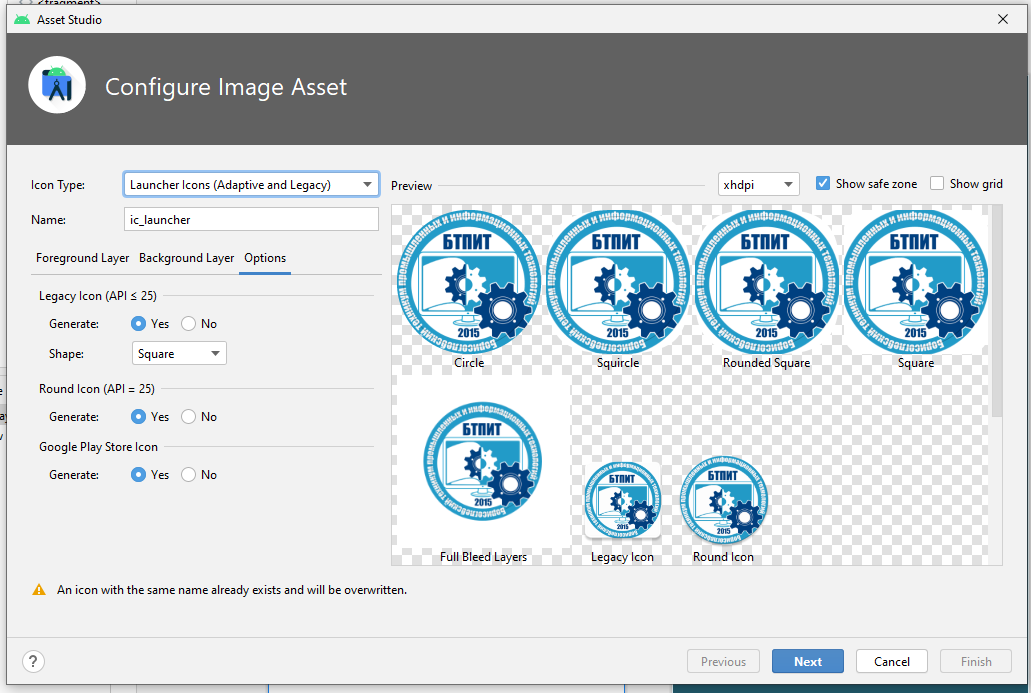
2. Выберите изображение (1) и переместите ползунок Resize (2), пока логотип не будет попадать в границы — они отмечены серым цветом:



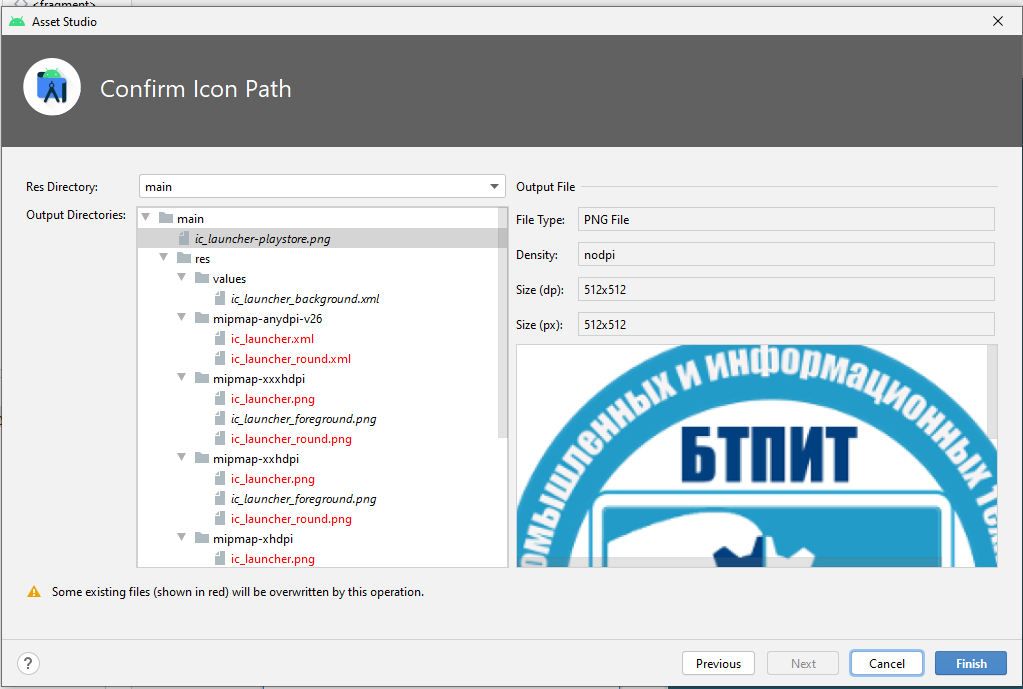
3. Перейдите на вкладку Background Layer (1), выберите Asset Type — Color (2) и поставьте цвет — FFFFFF (3):



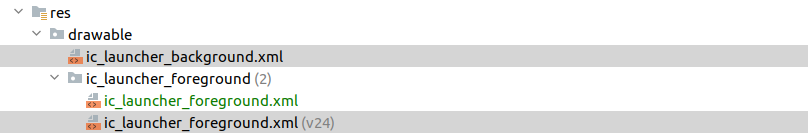
4. Перейдите на вкладку Options. По умолчанию настройки выставлены в генерацию иконок, т. е. будут сгенерированы изображения для тех версий Android, которые не поддерживают адаптивные иконки:



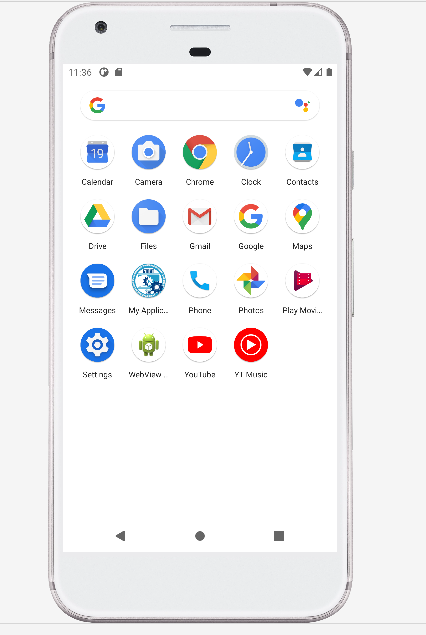
5. Подтвердите генерацию файлов, нажав Finish:



6. Удалите старые файлы (с иконкой Android):

[](https://github.com/netology-code/and2-homeworks/blob/master/02_resources/pic/asset06.png)

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)

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="ru.netology.nmedia">

<application

android:allowBackup="true"

android:icon="@mipmap/ic\_launcher"

android:label="@string/app\_name"

android:roundIcon="@mipmap/ic\_launcher\_round"

android:supportsRtl="true"

android:theme="@style/AppTheme">

<activity android:name=".MainActivity">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

</manifest>

Эти значения ведут на файлы mipmap/ic\_launcher и (mipmap/ic\_launcher\_round) соответственно. В зависимости от версии платформы это будут либо сгенерированные изображения в формате PNG или XML, в которых стоят ссылки на foreground и background ресурсы.

Опубликуйте изменения в вашем проекте на GitHub. Убедитесь, что apk собирается с помощью GitHub Actions и при установке в эмуляторе иконка приложения соответствует установленной вами.

**Задача Translations**

**Легенда**

Мы договорились, что всё делаем на двух языках: русском и английском.

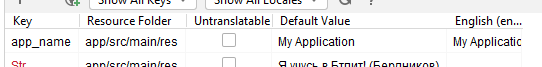
Добавьте файлы переводов в своё приложение.

Переводиться должны:

1. Название приложения: пусть на русском будет «НМедиа».
2. Текст: пусть на русском будет «НМедиа!».

**Задача**

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.

**Задача Layout**

**Легенда**

Доверстайте приложение, чтобы получился следующий вид:

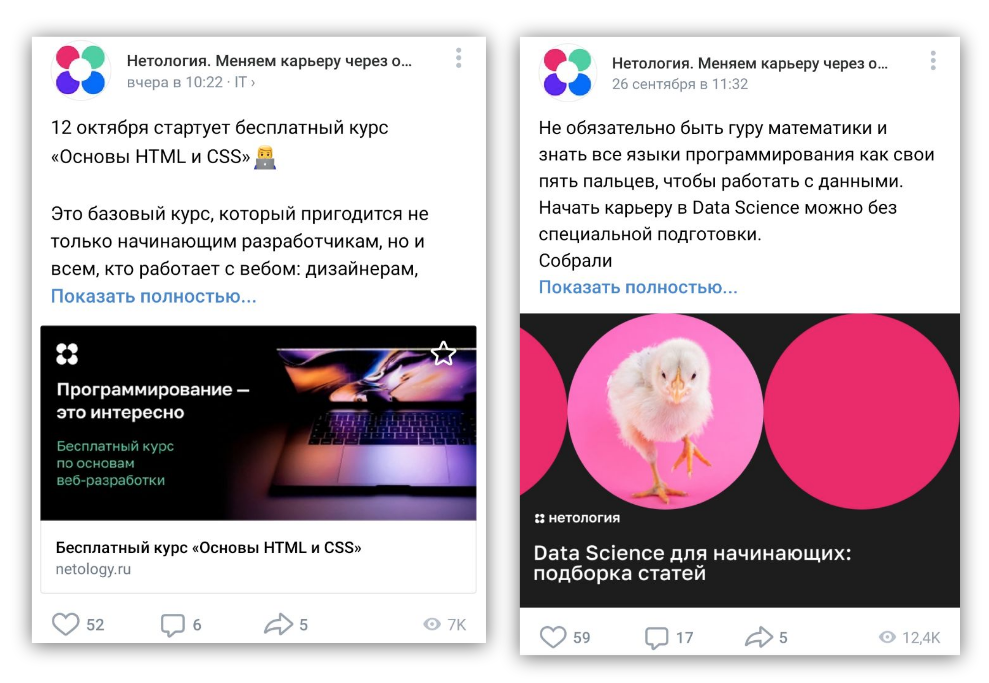
Обратите внимание на блок с количеством просмотров. С ним есть небольшая проблема, поскольку он расположен справа. Если количество просмотров вырастет, например, до 500, то есть два варианта:

1. Установить фиксированное расстояние от текста до границы родителя, а саму иконку приклеить к границе текста.
2. Оставить достаточное количество места, чтобы поместилось и 500, и 1К. Тогда на всех карточках положение этого блока будет одинаковым.

Проведите небольшое исследование и посмотрите, каким образом это реализовано во ВКонтакте.

Если нет доступа к Vk

Если у вас нет доступа к ВКонтакте, используйте наши скриншоты:

[](https://github.com/netology-code/and2-homeworks/blob/master/03_constraint_layout/pic/vk.png)

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

Опубликуйте изменения в вашем проекте на GitHub. Убедитесь, что apk собирается с помощью GitHub Actions.

**Задача**

1. При клике на like должна меняться не только картинка, но и число рядом с ней: лайкаете — увеличивается на 1, дизлайкаете — уменьшается на 1.
2. При клике на share должно увеличиваться число рядом: 10 раз нажали на share — +10.
3. 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.

Обратите внимание:

1. 1.1К отображается по достижении 1 100.
2. После 10К сотни перестают отображаться.
3. После 1M сотни тысяч отображаются в формате 1.3M.
4. Подумайте, можно ли это вынести в какую-то функцию вместо того, чтобы хранить эту логику в Activity.

**Задача**

1. Установите обработчики OnClickListener на binding.root и binding.like.
2. Поставьте внутрь обработчиков точки останова.
3. Запустите приложение в режиме отладки.
4. Кликните на кнопку Like, на кнопку с тремя точками (на ней пока нет обработчика), на текст, на аватар.
5. Установите обработчик OnClickListener на аватар и кликните по нему снова.

В качестве результата напишите ответы на следующие вопросы:

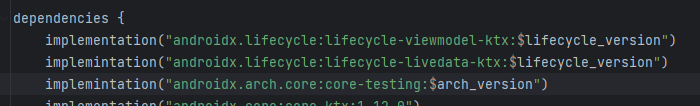
1. Какой из обработчиков сработал при клике на кнопку Like?
2. Сработал ли обработчик на binding.root при клике на кнопку с тремя точками?
3. Сработал ли обработчик на binding.root при клике на текст?
4. Сработал ли обработчик на binding.root при клике на аватар до установки на avatar собственного обработчика?
5. Сработал ли обработчик на binding.root при клике на аватар после установки на avatar собственного обработчика?

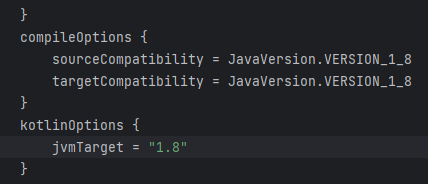
Опубликуйте изменения в вашем проекте на GitHub. Убедитесь, что apk собирается с помощью GitHub Actions и при установке в эмуляторе приложение работает корректно.

**Задача**

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.

****

****

**Repository**

**ViewModel**