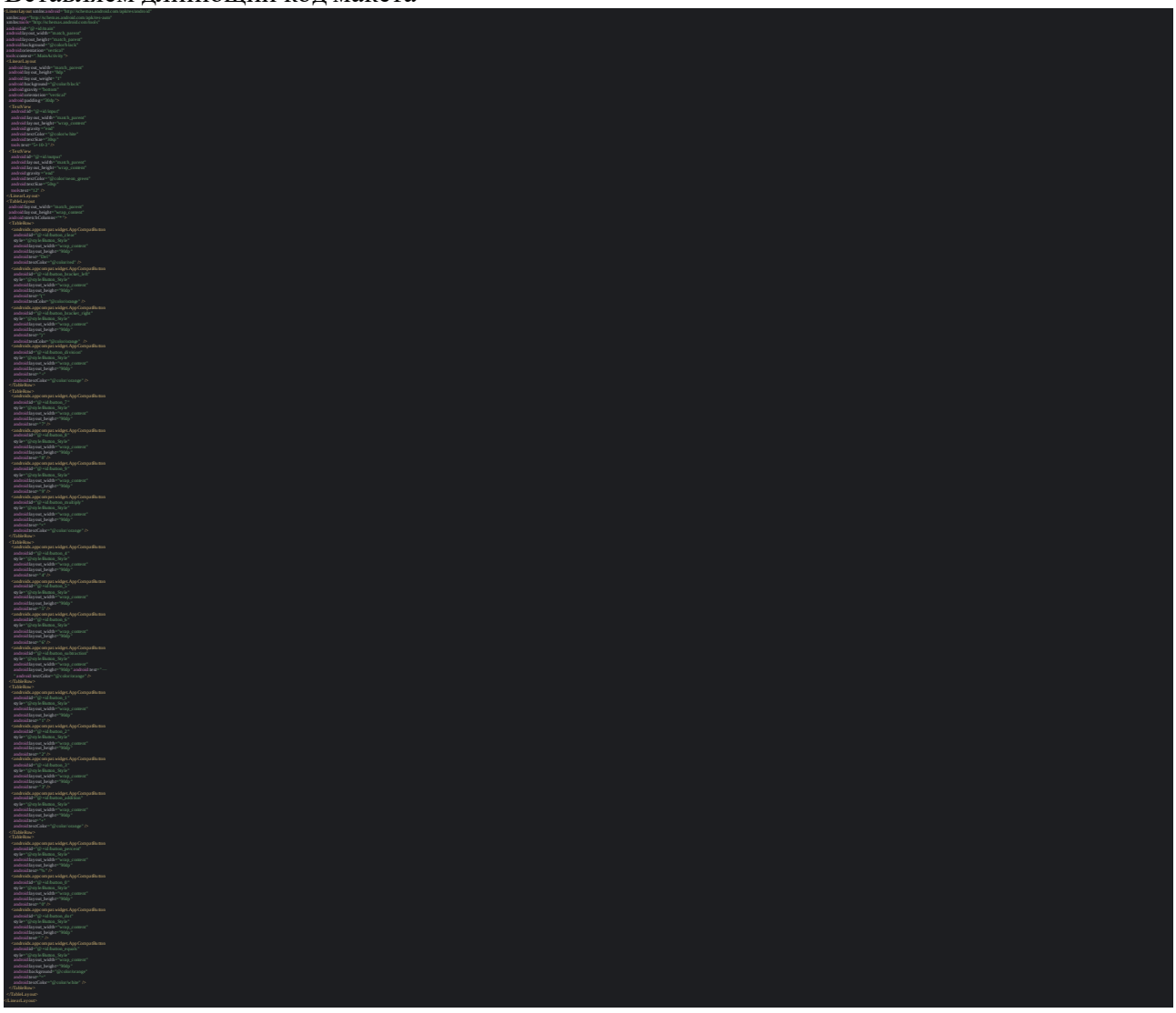
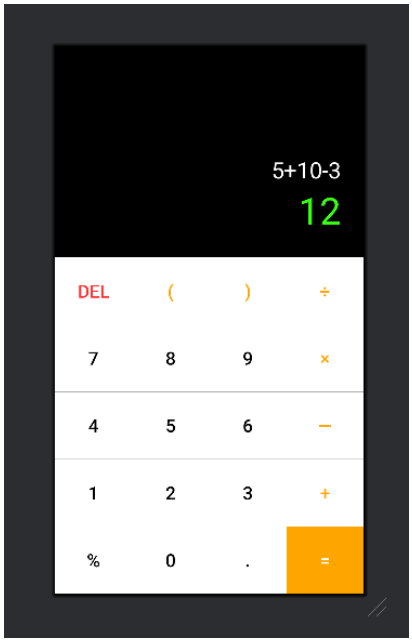
Практика 9: Разработка калькулятора

Согласно инструкции, добавим цвета и стиль для кнопки

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <color name="purple_200">#FFBB86FC</color>
  <color name="purple_500">#FF6200EE</color>
  <color name="purple_700">#FF3700B3</color>
  <color name="teal_200">#FF03DAC5</color>
  <color name="teal_700">#FF018786</color>
  <color name="black">#FF000000</color>
  <color name="white">#FFFFFFFF</color>
  <color name="red">#FF3131</color>
  <color name="pink">#FFC0CB</color>
  <color name="orange">#FFA500</color>
  <color name="neon_green">#39FF14</color>
  <style name="Button_Style" parent="Widget.AppCompat.Button.Colored">
    <item name="android:background">@color/white</item>
    <item name="android:textSize">24sp</item>
    <item name="android:textColor">@color/black</item>
    <item name="android:gravity">center</item>
  </style>
</resources>
```

Вставляем длинющий код макета





Для дальнейшей работы подключаем ViewBinding

```
buildFeatures{
  viewBinding = true
}
```

Привязываем действие каждой кнопки

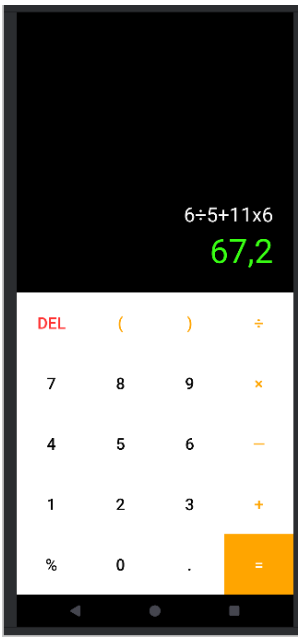
```
binding.buttonClear.setOnClickListener{
  binding.input.text = " "
  binding.output.text = " "
}
binding.buttonBracketLeft.setOnClickListener {
  addToInputText("(")
}
binding.buttonBracketLeft.setOnClickListener {
  addToInputText(")")
}
implementation ("net.objecthunter:exp4j:0.4.8")
```

Добавим функции для ввода символов и получения всего выражения

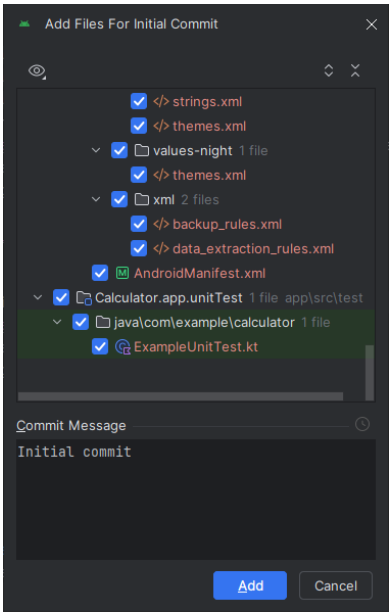
```
// Функция для добавления текста к полю ввода
private fun addToInputText(value: String) {
  binding.input.append(value)// Добавляем переданное значение в конец поля ввода
}
// Функция для получения строки ввода
private fun getInputExpression(): String {
  return binding.input.text.toString()// Возвращаем текст из поля ввода
}
private fun showResult() {
  try {
    val expression = getInputExpression()
      .replace("%","/100") // Замена %
      .replace("x", "*") // Замена x
      .replace("÷", "/") // Замена ÷
    val result = ExpressionBuilder(expression).build().evaluate()
    binding.output.text = DecimalFormat("0.######").format(result).toString()
    binding.output.setTextColor(
      ContextCompat.getColor(
        this,
        R.color.neon_green
```

```
    )
  )
} catch (e: Exception) {
  binding.output.text = "Ошибка"
  binding.output.setTextColor(
  ContextCompat.getColor(
      this,
      R.color.red
    )
  )
 }
}
```

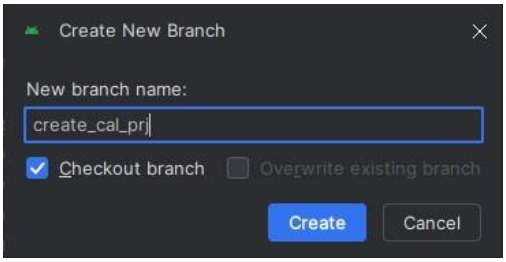
Тестируем



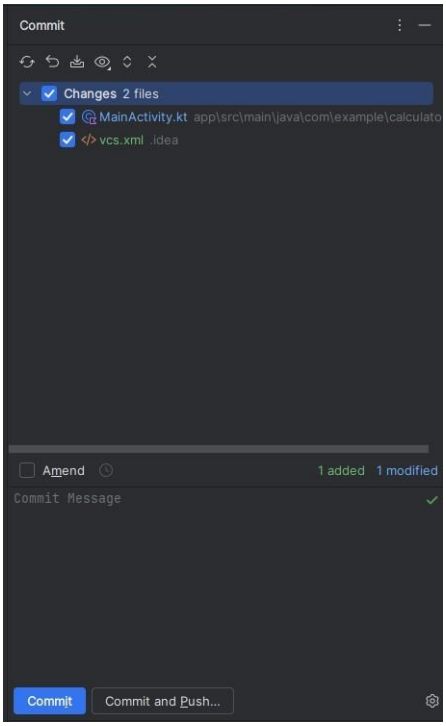
Выкладываем проект на GitHub



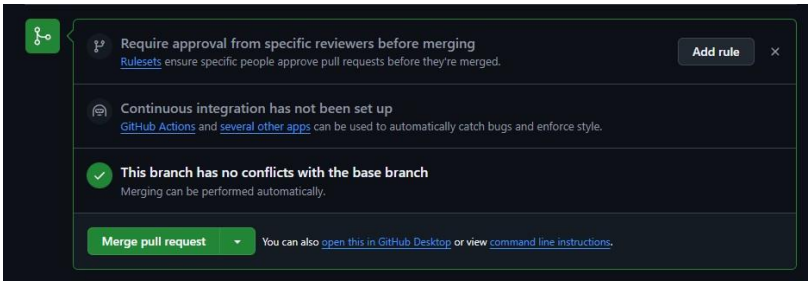
Создаем новую ветку



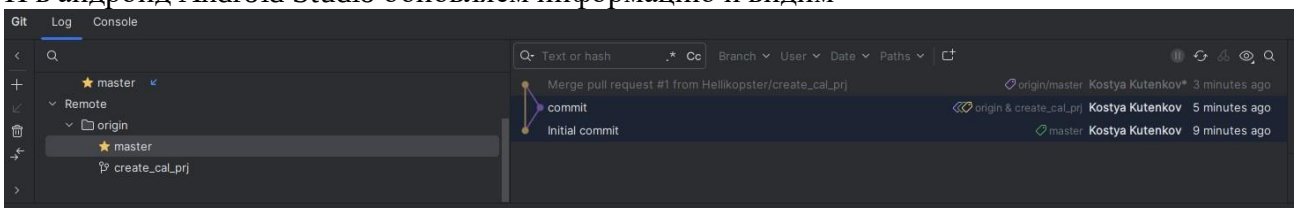
Коммитим изменения и пушим



Сливаем две ветки вместе



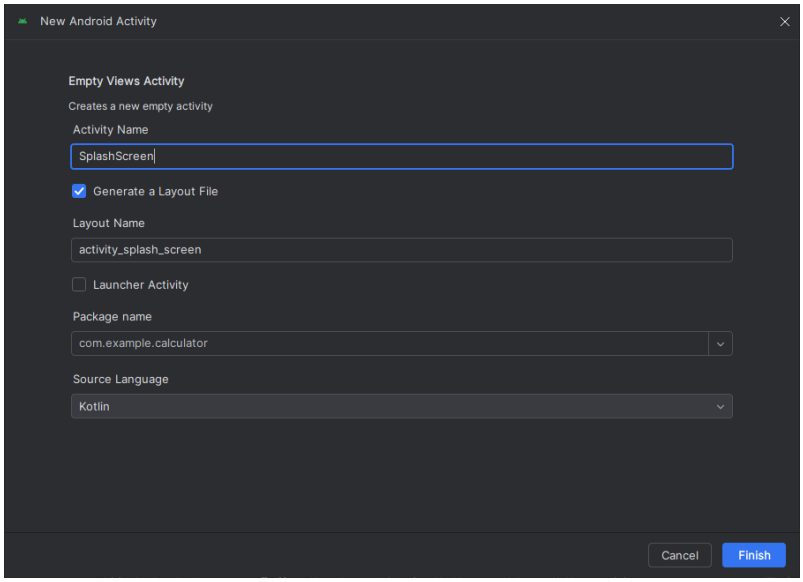
И в андроид Android Studio обновляем информацию и видим



Ссылка на GitHub

Заставка приложения

Добавим новую активность



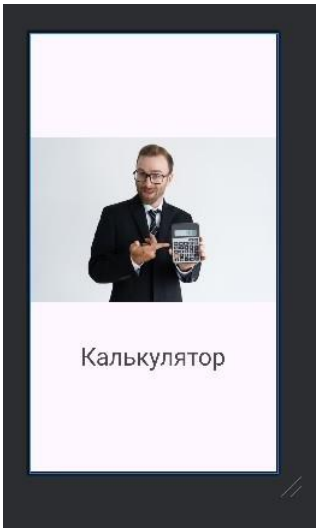
Оформляем

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 android:id="@+id/main"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 tools:context=".SplashScreen">

 <ImageView
  android:id="@+id/imageView"
  android:layout_width="match_parent"
  android:layout_height="400dp"
  android:layout_marginTop="112dp"
  android:src="@drawable/splash_screen"
  app:layout_constraintEnd_toEndOf="parent"
  app:layout_constraintHorizontal_bias="0.495"
  app:layout_constraintStart_toStartOf="parent"
  app:layout_constraintTop_toTopOf="parent" />

 <TextView
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:gravity="center"
  android:text="@string/title"
  android:textSize="40sp"
  app:layout_constraintTop_toBottomOf="@+id/imageView"
  tools:layout_editor_absoluteX="0dp" />

</androidx.constraintlayout.widget.ConstraintLayout>
```



В функции onCreate класса SplashScreen следующий код

```
override fun onCreate(savedInstanceState: Bundle?) {
  super.onCreate(savedInstanceState)
  setContentView(R.layout.activity_splash_screen)
  android.os.Handler(Looper.getMainLooper()).postDelayed({
  val intent = Intent(this, MainActivity::class.java)
  startActivity(intent)
  }, 2000)
}
```

Немного подправляем manifest

```
<activity
  android:name=".MainActivity"
  android:exported="false" />
<activity
  android:name=".SplashScreen"
  android:exported="true">
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />

    <category   android:name="android.intent.category.LAUNCHER"   />
  </intent-filter>
</activity>
```

И теперь каждый появляется новая заставка