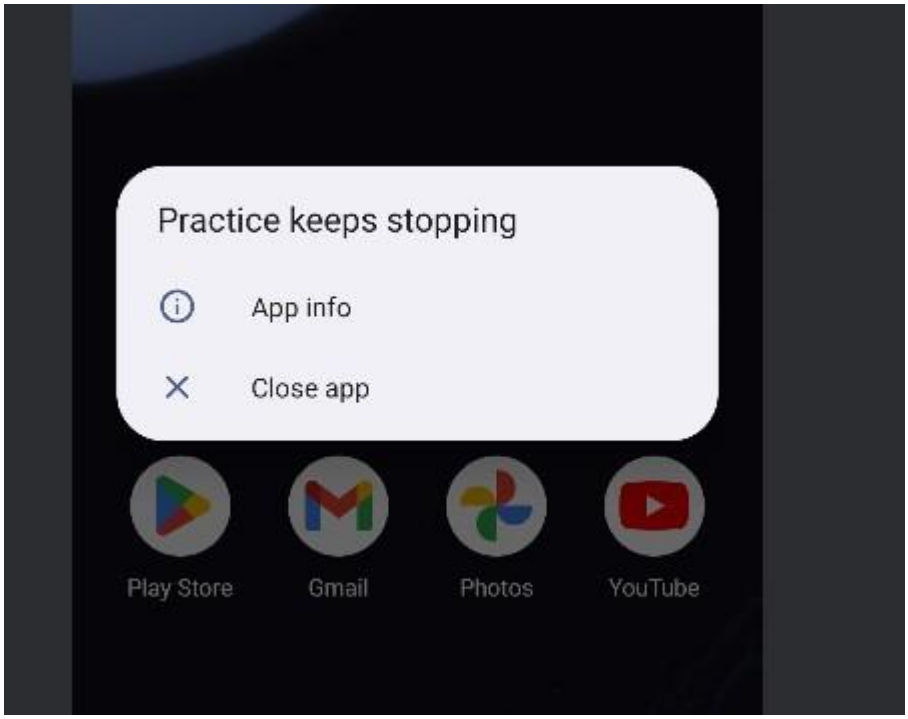
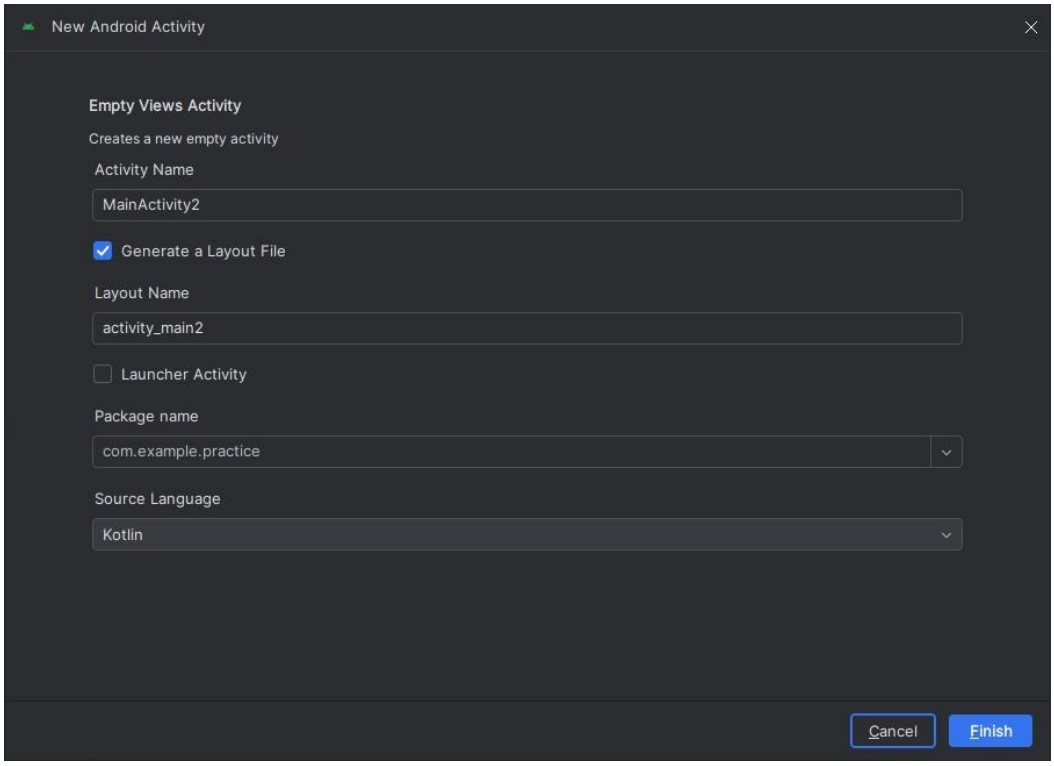
Практика 4

Работа с Intent. Переход между активностями.

Добавляем класс Activity… Снова? Хотя в прошлой практике мы и так его создали



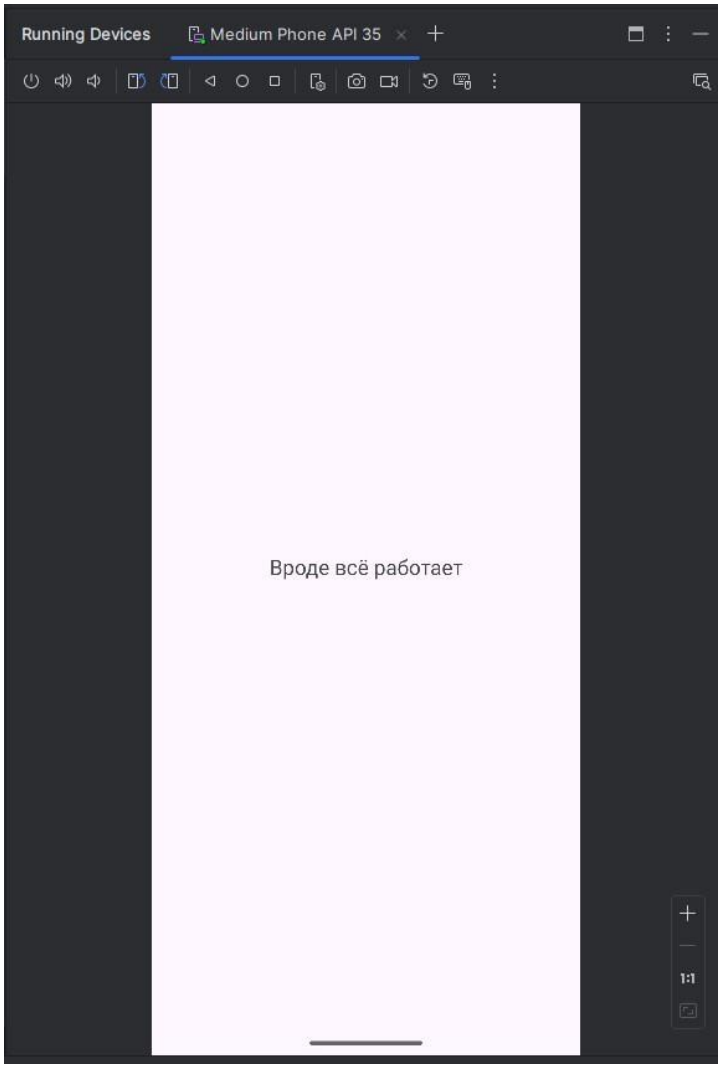
А, ну если так, тогда окей. Добавляем новую пустую активность (теперь уже правильно)



И поменяем название активности в коде кнопки (с Activivty2 на MainActivity2)

```
val btn : Button = findViewById(R.id.btn_1)
btn.setOnClickListener{
  val intent = Intent( packageContext: this, MainActivity2::class.java)
  startActivity(intent)
}
```

Тестируем



Вроде всё работает

Передача значений к другой активности

Для передачи значений используется объект класса Intent. Попробуем передать через него текст из редактируемого поля.

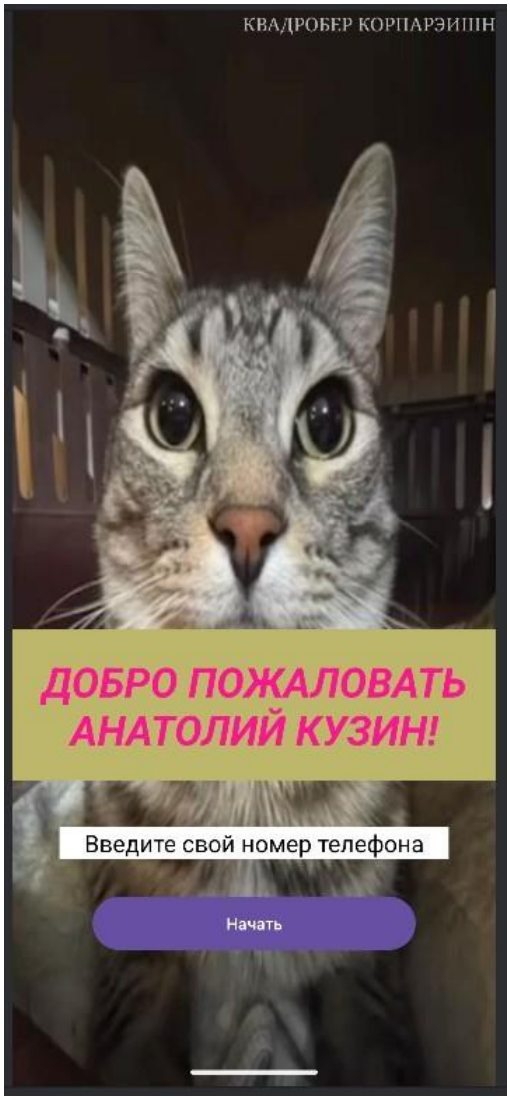
Отправка данных

```
val btn : Button = findViewById(R.id.btn_1)
btn.setOnClickListener{
  val intent = Intent( packageContext: this, MainActivity2::class.java)
  intent.putExtra( name: "phone_number", findViewById<EditText>(R.id.editText).text.toString())
  startActivity(intent)
}
```

Получение данных

```
val text : TextView = findViewById(R.id.title)
text.text = intent.getStringExtra( name: "phone_number")
```

Протестируем



Задание 1

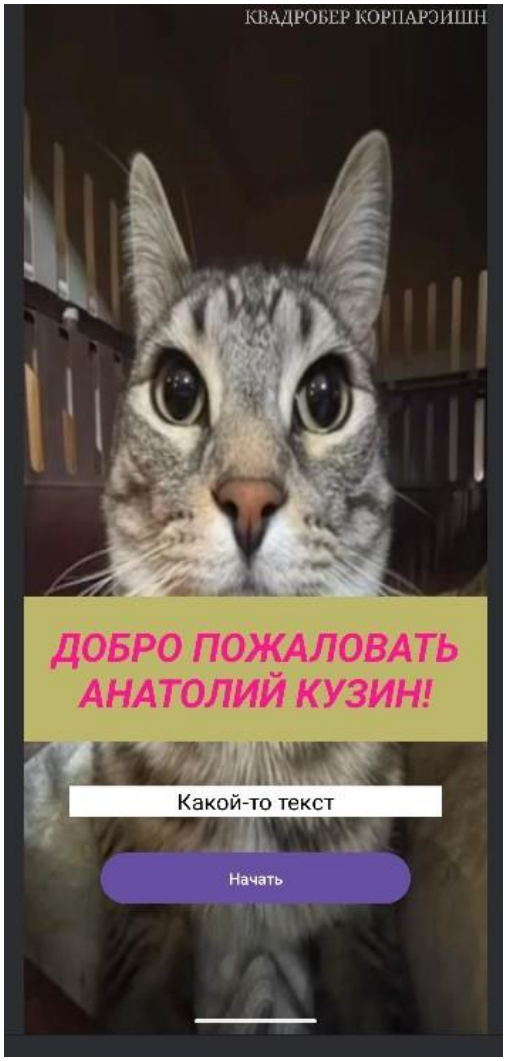
Для передачи числа используется точно такая же функция, как и для строки. Передадим число 123

```
val btn : Button = findViewById(R.id.btn_1)
btn.setOnClickListener{
  val intent = Intent( packageContext: this, MainActivity2::class.java)
  intent.putExtra( name: "phone_number", findViewById<EditText>(R.id.editText).text.toString())
  intent.putExtra( name: "num",  value: 123)
  startActivity(intent)
}
```

А для получения Int используется функция getIntExtra, которая работает точно так же

```
val text: TextView = findViewById(R.id.title)
var res = "Получены данные:\n"
res += intent.getStringExtra( name: "phone_number") + "\n"
res += intent.getIntExtra( name: "num",  defaultValue: 0).toString()
text.text = res
```

Результат

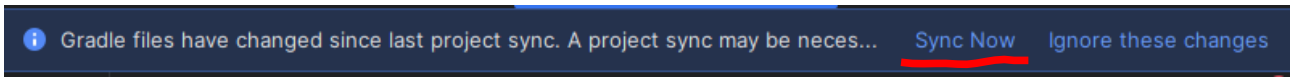


Задание 2

ViewBinding используется для обращения к элементам интерфейса напрямую, а не через поиск по id как это делалось в функции findViewById. Сам процесс подключения немного требует выполнения некоторых шагов, но в дальнейшем обращаться к элементам становится куда проще.

Для его подключения нужно добавить в файл build.grade.kts viewBinding = true и синхронизировать изменения:

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



После, внутри основного класса добавить переменную типа ActivityMainBinding, которая автоматически формируется при создании активности. К примеру для MainActivity2 был создан класс ActivityMain2Binding

MainActivity

```
class MainActivity : AppCompatActivity() {
  lateinit var binding: ActivityMainBinding

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    binding = ActivityMainBinding.inflate(layoutInflater)
    setContentView(binding.root)
```

MainActivity2

```
class MainActivity2 : AppCompatActivity() {
    lateinit var binding: ActivityMain2Binding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMain2Binding.inflate(layoutInflater)
        setContentView(binding.root)
```

Функция inflate возвращает макет активности который передается в переменную binding, и уже через неё можно обратиться к любому элементу

MainActivity с findViewById

```
val btn : Button = findViewById(R.id.btn_1)
btn.setOnClickListener{
  val intent = Intent( packageContext: this, MainActivity2::class.java)
  intent.putExtra( name: "phone_number", findViewById<EditText>(R.id.editText).text.toString())
  intent.putExtra( name: "num",  value: 123)
  startActivity(intent)
}
```

MainActivity с BindingView

```
val btn : Button = binding.btn1
btn.setOnClickListener{
  val intent = Intent( packageContext: this, MainActivity2::class.java)
  intent.putExtra( name: "phone_number", binding.editText.text.toString())
  intent.putExtra( name: "num",  value: 123)
  startActivity(intent)
}
```

MainActivity2 с findViewById

```
val text: TextView = findViewById(R.id.title)
var res = "Полученые данные:\n"
res += intent.getStringExtra( name: "phone_number") + "\n"
res += intent.getIntExtra( name: "num",  defaultValue: 0).toString()
text.text = res
```

MainActivity2 с BindingView

```
val text: TextView = binding.title
var res = "Полученые данные:\n"
res += intent.getStringExtra( name: "phone_number") + "\n"
res += intent.getIntExtra( name: "num",  defaultValue: 0).toString()
text.text = res
```

Результат на лицо. Более того, BindingView хранит все компоненты одной активности отдельно и нельзя по ошибке обратиться к компоненту из другой активности. Например, надпись, приветствующая в начале, остается доступной во второй активности.

И вот эта безобидная строка может вызвать ошибку:

```
class MainActivity2 : AppCompatActivity() {
    lateinit var binding: ActivityMain2Binding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMain2Binding.inflate(layoutInflater)
        setContentView(binding.root)

        findViewById<TextView>(R.id.welcome).text = "Привет!"
        val text: TextView = binding.title
        var res = "Полученые данные:\n"
        res += intent.getStringExtra( name: "phone_number") + "\n"
        res += intent.getIntExtra( name: "num",  defaultValue: 0).toString()
        text.text = res
    }
}
```

Вывод: BindingView гораздо безопаснее, удобней и быстрее, чем findViewById