Практическая работа 5
Настройка проверки полей

Откройте прототип приложения по ссылке: https://www.figma.com/proto/AP3K5lRj7pOlMhKrI8u5eD/%D0%9F%D0%A03?node-id=44%3A138&scaling=min-zoom&page-id=1%3A2&starting-point-node-id=44%3A126

Нажмите R для запуска интерактивного прототипа с первого экрана. Все элементы прототипа кликабельны. Если щелкнуть левой кнопкой мыши по черному полю вокруг экрана - будут выделены элементы доступные для нажатия. Проанализируйте, как реализованы переходы и взаимодействия между элементами (в данной работе вам не нужно копировать интерактивный прототип в свои черновики, вам нужен только режим проигрывания прототипа)

Откройте проект, выполненный **в четвертой практической работе**, вам предстоит настроить проверку текстовых полей на заполнение..

На экране входа есть два текстовых поля, переход на следующий экран возможен только если и почта и пароль будут указаны. В противном случае будет появляться диалоговое окно с сообщением об ошибке.

Объявите два объекта класса EditText и свяжите с текстовыми полями (взаимодействие с компонентами экрана возможно только через объекты классов, соответствующих элементам интерфейса, которые будут с ними связаны по id). Пропишите проверку на пустоту полей: в случае, если все поля заполнены, будет переход на экран-заглушку, если поля пусты - появится диалоговое окно с сообщением об ошибке. Пропишите в файле логики SigninActivity (имя активити, соответствующей вашему экрану входа) следующие строки кода:

```
3  import android.content.Intent
4  import androidx.appcompat.app.AppCompatActivity
5  import android.os.Bundle
6  import android.view.View
7  import android.widget.EditText
8  import androidx.appcompat.app.AlertDialog
9
10 class SigninActivity : AppCompatActivity() {
11     lateinit var mail:EditText
12     lateinit var pass:EditText
13     override fun onCreate(savedInstanceState: Bundle?) {
14         super.onCreate(savedInstanceState)
15         setContentView(R.layout.activity_signin2)
16         mail=findViewById(R.id.mailText)
17         pass=findViewById(R.id.passText)
18     }
19
20     fun next(view: View) {
21         if(mail.text.toString().isNotEmpty()&&pass.text.toString().isNotEmpty())
22         {
23             val intent = Intent( packageContext: this@SigninActivity, PatchActivity::class.java)
24             startActivity(intent)
25             finish()
26         }
27         else
28         {
29             val alert=AlertDialog.Builder( context: this)
30                 .setTitle("Заполните тестовые поля")
31                 .setPositiveButton( text: "OK",  listener: null)
32                 .create()
33                 .show()
34         }
35     }
36 }
```

11 и 12 строки - объявление объектов класса EditText с отложенной инициализацией

16 и 17 строки - связывание объектов класса и элементов разметки по идентификатору

21 строка - условие проверяет указанные поля на пустоту (isNotEmpty - не пустые, имеют какие-то значения, т.е. в текстовые поля был введен любой текст)

.text - проверяет значение параметра text для текстового поля, то, что вводится в текстовые поля автоматически попадает в параметр text

.toString() - конвертировать значение в формат String - преобразовать введенные символы в строку

Если необходимо обратиться или получить значение, введенное в текстовое поле, к нему следует обратиться используя данную языковую конструкцию: имя объекта класса EditText.text.toString()

28 - 34 строки - появление диалогового окна

Запустите приложение, проверьте как работает проверка на пустоту.

Еще раз просмотрите работу прототипа, самостоятельно реализуйте проверку заполнения полей для экрана регистрация. Ваше приложение должно работать как показано в интерактивном прототипе.

Доработайте код приложения: при нажатии на кнопку “Зарегистрироваться” экрана Регистрация, если все поля заполнены, при переходе на экран-заглушку должно появляться всплывающее уведомление с текстом “Регистрация прошла успешно”.

Пример реализации всплывающего уведомления представлен ниже, измените его в соответствии с поставленным заданием и допишите данную строку кода в условие перед строчками кода, отвечающими за переход на другой экран:

```
Toast.makeText( context: this,  text: "поля пусты", Toast.LENGTH_LONG).show()
```

Сейчас переход на следующий экран происходит после прохождения проверки на пустоту, усложните процедуру, добавив проверку на корректность вводимого адреса электронной почты. Требование к вводимому email: имя может состоять только из строчных букв латинского алфавита, количество не более ста+ @ + доменное имя второго уровня может содержать строчные буквы латинского алфавита не более шести, домен верхнего уровня может состоять только из строчных букв, допускается не более пяти знаков после точки

Дополните код проверки на пустоту полей следующими строками (код может содержать ошибки по ранее пройденному материалу):

```
    lateinit var pass:EditText

    val pattern=("[a-z]{1,100}" +"@"+"[a-z]{1,6}"+"\\."+"[a-z]{1,5}")

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        mail = findViewById(R.id.editTextTextPersonName)
        pass = findViewById(R.id.editTextTextPersonName2)
    }
    fun emailValid(text:String):Boolean
    {
        return Pattern.compile(pattern).matcher(text).matches()}

    fun but(view: View) {
      if (mail.text.toString().isNotEmpty()&&pass.text.toString().isNotEmpty())
        {
            if (emailValid(mail.text.toString()))
            {
           val intent=Intent( packageContext: this@MainActivity, PatchActivity::class.java)
            startActivity(intent)
            finish()
        }
            else {
                Toast.makeText( context: this, text: "ошибка при заполнении поля email",Toast.LENGTH_SHORT).show()
            }

        }
else
        {
            val alert=AlertDialog.Builder( context: this)
```

17 строка - настройка паттерна для проверки допустимых символов в текстовом поле ([допустимые символы для ввода]{количество символов})

25 строка - функция проверки символов в текстовом поле на наличие недопустимых (true - символы попадают в указанный паттерн, false - присутствуют недопустимые символы)

32 строка - если значение функции истина - происходит переход на следующий экран, иначе появляется уведомление об ошибке.

Запустите приложение, проверьте как работает проверка. Настройте проверку правильности заполнения почтового поля для экрана Регистрация.

Самостоятельное задание

Откройте проект, выполненный во второй контрольной работе. Все текстовые поля должны проверяться на пустоту, если хотя бы одно поле пустое – должно появляться диалоговое окно с сообщением об ошибке. Почтовые поля должны проверяться на валидацию паттерна: имя может состоять из сочетаний прописные или строчных букв и цифр от 0 до 6. Количество символов не больше 30, домен второго уровня должен состоять только из строчных букв, не более восьми символов. Домен верхнего уровня только из строчных букв длиной не более пяти символов. Если почтовое поле содержит недопустимые символы, должно появляться всплывающее сообщение об ошибке. На экране создания аккаунта пароль и повтор пароля должны сравниваться, если пароли не равны, должно появляться всплывающее сообщение.