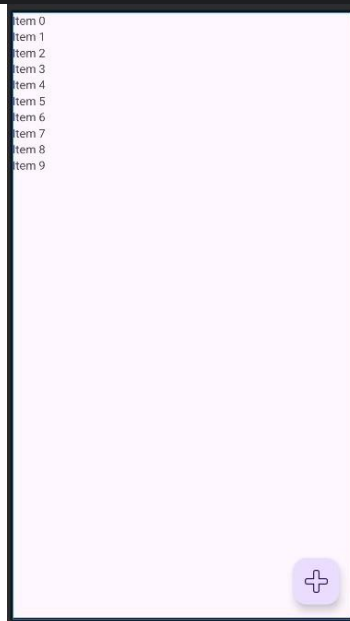


Практика 14: Добавление элементов в список

Задаем макет с RecyclerView и с FloatingActionButton

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/main">
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_margin="16dp"
        app:maxImageSize="30dp"
        android:src="@drawable/add" />
</RelativeLayout>
```



Определяем в отдельном файле класс данных одного элемента Item.kt

```
data class Item(val text: String)
```

Затем создаем list_rec.xml для определения макета одного элемента

```
<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:layout_width="match_parent"
    android:layout_height="wrap_content">
    <TextView
        android:id="@+id/txt1"
        android:layout_width="0dp"
        android:layout_height="64dp"
        android:layout_marginTop="16dp"
        tools:text="test"
        android:textSize="36sp"
```

```

app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>

```



Создаем класс адаптера ItemAdapter.kt

```

class ItemAdapter(private val item: List<Item>) :
    RecyclerView.Adapter<ItemAdapter.ItemViewHolder>() {
    class ItemViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        val textView : TextView = view.findViewById(R.id.txt1)
    }
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
        ItemViewHolder {
        val view =
            LayoutInflater.from(parent.context).inflate(R.layout.item_rec, parent, false)
        return ItemViewHolder(view)
    }
    override fun onBindViewHolder(holder: ItemViewHolder, position: Int) {
        holder.textView.text = item[position].text
    }
    override fun getItemCount(): Int {
        return item.size
    }
}

```

И в классе MainActivity добавим логику

```

class MainActivity : AppCompatActivity() {
    // Входные компоненты для RecyclerView и FloatingActionButton
    private lateinit var recyclerView: RecyclerView
    private lateinit var fab: FloatingActionButton
    // Список элементов, которые будут отображаться в RecyclerView
    private val items: MutableList<Item> = mutableListOf()
    // Адаптер для управления отображением элементов в RecyclerView
    private lateinit var adapter: ItemAdapter
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) {
            v, insets ->
            val systemBars =
                insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right,
                systemBars.bottom)
            insets
        }
        recyclerView = findViewById(R.id.recyclerView)
        fab = findViewById(R.id.fab)
    }
}

```

```
// Создаем адаптер, передавая ему список элементов
adapter = ItemAdapter(items)
recyclerView.layoutManager = LinearLayoutManager(this) // Устанавливаем LinearLayoutManager для
RecyclerView
recyclerView.adapter = adapter // Устанавливаем адаптер для RecyclerView
// Устанавливаем обработчик нажатия на FloatingActionButton
fab.setOnClickListener {
    showInputDialog() // Вызываем метод для отображения диалогового окна
}
}
// Метод для отображения диалогового окна для ввода текста
private fun showInputDialog() {
    val builder = AlertDialog.Builder(this) // Создаем экземпляр AlertDialog.Builder
    builder.setTitle("Добавить элемент") // Устанавливаем заголовок диалогового окна
    val input = EditText(this) // Создаем EditText для ввода текста
    builder.setView(input) // Устанавливаем созданный EditText в диалог
    // Устанавливаем кнопку "Добавить" в диалоговом окне
    builder.setPositiveButton("Добавить") { dialog, _ ->
        val text = input.text.toString() // Получаем текст из EditText
        if (text.isNotEmpty()) { // Проверяем, не пустой ли он
            items.add(Item(text)) // Добавляем новый элемент в список
        }
    }
}
```

```

    adapter.notifyItemInserted(items.size - 1) // Уведомляем адаптер о добавлении нового элемента
  }
  dialog.dismiss() // Закрываем диалог
}
// Устанавливаем кнопку "Отмена" в диалоговом окне
builder.setNegativeButton("Отмена") { dialog, _ -> dialog.cancel() }
builder.show() // Показываем диалоговое окно
}
}

```

Результат

