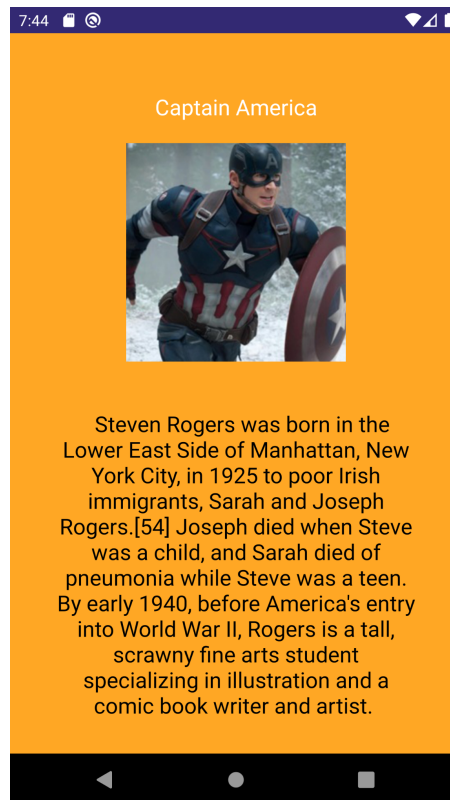


## Практическая работа 13

### “GET запросы”



**<https://www.simplifiedcoding.net/demos> - начальный URL к API**

Продолжите работу с приложением Cinema.

Вам необходимо оформить экран, который будет содержать информацию о четырех фильмах: название, постер, описание. Для получения данных используйте GET запрос. Недостающие строки кода пропишите самостоятельно.

Создайте дата класс, с помощью которого будут получены данные с сервера. Обратите внимание, в каком формате будет приходить ответ на запрос <https://www.simplifiedcoding.net/demos/marvel/> пропишите соответствующие поля дата класса (данные приходят в виде массива, но названия у него нет поэтому просто перечислите поля класса, которые будут получены):

```

1 package com.example.testproject.model
2
3 data class movie(
4     val name:String,
5     val realname:String,
6     val taem:String,
7     val firstappearance: String,
8     val createby: String,
9     val publisher:String,
10    val imageUrl:String,
11    val bio:String
12 )

```

Создайте класс-интерфейс для GET запроса:.

```

1 package com.example.testproject.`interface`
2
3 import com.example.testproject.model.movie
4 import retrofit2.Call
5 import retrofit2.http.GET
6
7 interface movie_interface {
8     @GET("marvel")
9     fun getposter():Call<MutableList<movie>>
10 }

```

Обратите внимание, в Call должен вернуться список значений. Для этого объявим контейнер **MutableList** в угловых скобках пропишите название дата класса.

*Класс **List** входит в состав стандартной библиотеки и не требует явного импорта. Списки являются частью коллекций и служат контейнером для объектов. List представляет последовательный список элементов. При этом List представляет неизменяемую (immutable) коллекцию, которая в основном только обеспечивает получение элементов по позиции.*

*Изменяемые списки представлены интерфейсом **MutableList**. Он расширяет интерфейс **List** и позволяют добавлять и удалять элементы. Данный*

интерфейс реализуется классом *ArrayList*.

<https://metanit.com/kotlin/tutorial/7.2.php>

*List* (список) - упорядоченная коллекция, в которой к элементам можно обращаться по индексам — целым числам, отражающим положение элементов в коллекции.

<https://kotlinlang.ru/docs/collections-overview.html> - коллекции, определение, типы, использование

В кавычках укажите название ветки, где хранится информация о фильмах - *marvel*.




Самостоятельно создайте класс для ретрофита, который в дальнейшем будет использовать в активностях для обработки результатов.

Настройте ресайклер для отображения изображения и двух текстовых блоков. Размер текста 18 sp. Настройте адаптер для заполнения списка данными, полученными с сервера:

```
1 package com.example.testproject
2
3 import android.content.Context
4 import android.view.LayoutInflater
5 import android.view.View
6 import android.view.ViewGroup
7 import android.widget.ImageView
8 import android.widget.TextView
9 import androidx.recyclerview.widget.RecyclerView
10 import com.bumptech.glide.Glide
11 import com.example.testproject.model.movie
12
13 class adapterh(val con:Context, val Listt:MutableList<movie>):RecyclerView.Adapter<adapterh.Link1>() {
14     ...
15
16     override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): Link1 {
17         ...
18     }
19
20     override fun onBindViewHolder(holder: Link1, position: Int) {
21         Glide.with(con).load(Listt[position].imageurl).into(holder.im_di)
22         holder.ti_di.setText(Listt[position].name)
23         ...
24     }
25
26     override fun getItemCount(): Int {
27         return 3
28     }
29 }
30
31
32
33
34 }
```

Самостоятельно пропишите на месте многоточий недостающие строки

кода. Ниже представлен список после заполнения содержимым, у вас должна быть показана информация о четырех фильмах:

<p>Captain America</p>  <p>Steven Rogers was born in the Lower East Side of Manhattan, New York City, in 1925 to poor Irish immigrants, Sarah and Joseph Rogers.[54] Joseph died when Steve was a child, and Sarah died of pneumonia while Steve was a teen. By early 1940, before America's entry into World War II, Rogers is a tall, scrawny fine arts student specializing in illustration and a comic book writer and artist.</p>	<p>Anthony Edward Stark, the son of wealthy industrialist and head of Stark Industries, Howard Stark, and Maria Stark. A boy genius, he enters MIT at the age of 15 to study electrical engineering and later receives master's degrees in electrical engineering and physics. After his parents are killed in a car accident, he inherits his father's company.</p> <p>Wolverine</p>  <p>Wolverine was born James Howlett in northern Alberta, Canada, during the late 1880s, purportedly to rich farm owners John and Elizabeth Howlett,[27] though he is actually the illegitimate son of the Howletts' groundskeeper, Thomas Logan.[28] After Thomas is thrown off the Howletts' property for an attempted rape perpetrated by his other son, named simply Dog, he returns to the Howlett manor and kills John Howlett. In retaliation, young James kills Thomas with bone claws that emerge from the back of his hands, as his mutation manifests.[29] He flees with his childhood companion, Rose, and grows into manhood on a mining colony in the Yukon, adopting the name "Logan".[30] When Logan accidentally kills Rose with his claws, he flees the colony and lives in the wilderness among wolves,[31] until he is captured and placed in a circus. [32] Saul Creed, brother of Victor Creed, frees Logan, but after he betrays Logan and Clara Creed to Nathaniel Essex, Logan drowns Creed in Essex's potion.[33] Logan returns to civilization, residing with the Blackfoot people. Following the death of his Blackfoot lover, Silver Fox, at the hands of Victor Creed, now known as Sabretooth,[34] he is ushered into the Canadian military during World War I. Logan spends time in Madripoor before settling in Japan, where he marries Itsu and has a son, Daken. Logan is unaware of his son for many years.</p>
<p>Iron Man</p> 	

Подключите возможность обращения к ретрофиту в файле логики активити и подкорректируйте подключение адаптера (в коде нет пропущенных строк):

```

1 package com.example.testproject
2 import android.content.SharedPreferences
3 import androidx.appcompat.app.AppCompatActivity
4 import android.os.Bundle
5 import android.view.View
6 import android.widget.TextView
7 import android.widget.Toast
8 import androidx.core.content.ContentProviderCompat.requireContext
9 import androidx.recyclerview.widget.RecyclerView
10 import com.example.testproject.`interface`.RetrofitServices
11 import com.example.testproject.`interface`.movie_interface
12 import com.example.testproject.model.data_base
13 import com.example.testproject.model.movie
14 import retrofit2.Call
15 import retrofit2.Response
16 import retrofit2.create
17
18 class MainActivity : AppCompatActivity() {
19     lateinit var rv33:RecyclerView
20     lateinit var rv22:RecyclerView
21     lateinit var res:TextView
22     lateinit var adapter:adapterh
23     override fun onCreate(savedInstanceState: Bundle?) {
24         super.onCreate(savedInstanceState)
25         setContentView(R.layout.activity_main)
26
27         rv33=findViewById(R.id.rv)
28         val retro=movieretrofit().getRetrofit()
29         val urlretr=retro.create(movie_interface::class.java)
30         val retro_call:Call<MutableList<movie>> = urlretr.getposter() //пробелы не забыть
31         retro_call.enqueue(object:retrofit2.Callback<MutableList<movie>>
32         {
33             override fun onResponse(call: Call<MutableList<movie>>, response: Response<MutableList<movie>>)
34             {
35                 adapter=adapterh(baseContext,response.body() as MutableList<movie>)
36                 rv33.adapter=adapter
37             }
38             override fun onFailure(call: Call<MutableList<movie>>, t: Throwable) {
39                 Toast.makeText(applicationContext, t.localizedMessage, Toast.LENGTH_SHORT).show()
40             } })}

```

Обратите внимание на 30 строку. Абстрактный класс Call должен вернуть коллекцию значений, поэтому прописываем не просто название дата класса, который содержит описание всех полей, но и указываем, что они должны быть получены списком <MutableList<movie>> (см.9 строку класса-интерфейса)

`enqueue()` асинхронно отправляет запрос и уведомляет ваше приложение обратным вызовом, когда возвращается ответ. Поскольку этот запрос асинхронный, *Retrofit* обрабатывает выполнение в фоновом потоке, чтобы основной поток пользовательского интерфейса не блокировался и не мешал.

Чтобы использовать метод `enqueue()`, вы должны реализовать два метода обратного вызова: `onResponse()` и `onFailure()`. Только один из этих методов будет вызван в ответ на данный запрос. `onResponse()`: вызывается для полученного ответа HTTP. Этот метод вызывается для ответа, который можно правильно обработать, даже если сервер возвращает сообщение об ошибке. Поэтому, если вы получите код состояния 404 или 500, этот метод все равно будет вызываться. Чтобы получить код состояния для обработки ситуаций на их основе, вы можете использовать метод `response.code()`. Вы также можете использовать метод `isSuccessful()` чтобы выяснить, находится ли код состояния в диапазоне 200-300, что указывает на успех.

`onFailure()`: вызывается, когда при подключении к серверу возникло сетевое исключение или произошло непредвиденное исключение при обработке запроса или обработке ответа.

*Context* – это объект, который предоставляет доступ к базовым функциям приложения: доступ к ресурсам, к файловой системе, вызов активности и т.д. *Activity* является подклассом *Context*, поэтому в коде мы можем использовать её как *ИмяАктивности.this* (напр. *MainActivity.this*), или укороченную запись *this*.

Классы *Service*, *Application* и др. также работают с контекстом. Доступ к контексту можно получить разными способами. Существуют такие методы как `getApplicationContext()`, `getContext()`, `getBaseContext()` или *this*, который упоминался выше, если используется в активности.

На первых порах не обязательно понимать, зачем он нужен. Достаточно

*помнить о методах, которые позволяют получить контекст и использовать их в случае необходимости, когда какой-нибудь метод или конструктор будет требовать объект Context в своих параметрах.*

<https://developer.alexanderklimov.ru/android/theory/context.php>

**ResponseBody** дает фреймворку понять, что объект, который вы вернули из метода надо конвертировать, чтобы получить готовое к отправке на клиент представление.

Запустите приложение, если вы все сделали правильно ваш экран будет выглядеть так:

## Iron Man



Anthony Edward Stark, the son of wealthy industrialist and head of Stark Industries, Howard Stark, and Maria Stark. A boy genius, he enters MIT at the age of 15 to study electrical engineering and later receives master's degrees in electrical engineering and physics. After his parents are killed in a car accident, he inherits his father's company.

## Wolverine





```

18 class MainActivity : AppCompatActivity() {
19     lateinit var rv33: RecyclerView
20     ...
21     lateinit var res: TextView
22     lateinit var adapter: adapterh
23     override fun onCreate(savedInstanceState: Bundle?) {
24         super.onCreate(savedInstanceState)
25         setContentView(R.layout.activity_main)
26         ...
27         rv33=findViewById(R.id.rv)
28         val retro=movieetrofit().getRetrofit()
29         val urlretr=retro.create(movie_interface::class.java)
30         val retro_call: Call<MutableList<movie>> = urlretr.getposter() //пробелы не забыть
31         retro_call.enqueue(object: retrofit2.Callback<MutableList<movie>> {
32             override fun onResponse(call: Call<MutableList<movie>>, response: Response<MutableList<movie>>)
33             {
34                 ...
35                 adapter=adapterh(baseContext,response.body() as MutableList<movie>)
36                 rv33.adapter=adapter
37                 Toast.makeText(applicationContext, response.code().toString(), Toast.LENGTH_SHORT).show()
38             }
39             override fun onFailure(call: Call<MutableList<movie>>, t: Throwable) {
40                 Toast.makeText(applicationContext, t.localizedMessage, Toast.LENGTH_SHORT).show()
41             }
42         })

```

Что изменилось в коде? Какое действие было добавлено? Что означает “новая” отображаемая информация?

Контрольное задание. Самостоятельно добавьте экран, на котором будет отображаться следующая информация о фильме (о всех фильмах, что представлены на сервере): название, как зовут главного героя, в каком комиксе он появился, когда была первая публикация комикса и кто его автор. Внешний вид экрана оформите самостоятельно, информация должна отображаться читаемо и понятно.