Retrieving data from the web

Applikationsudvikling

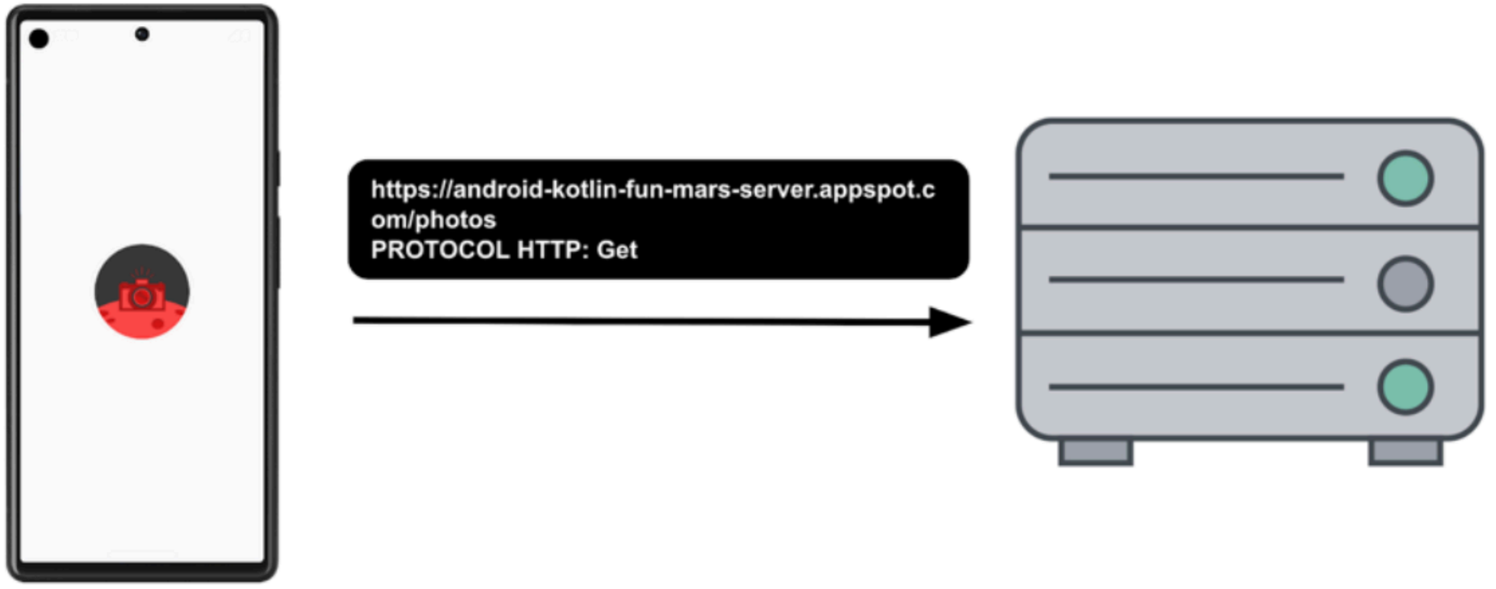
NIFR 2024

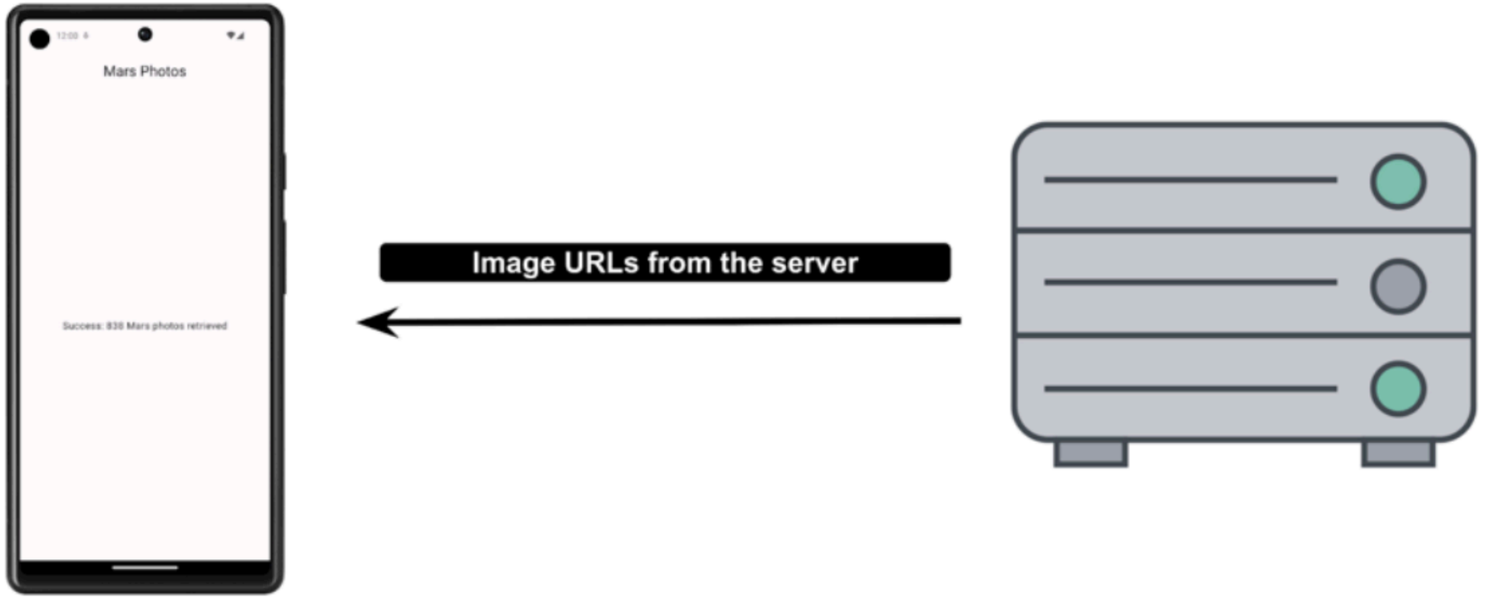
Interface

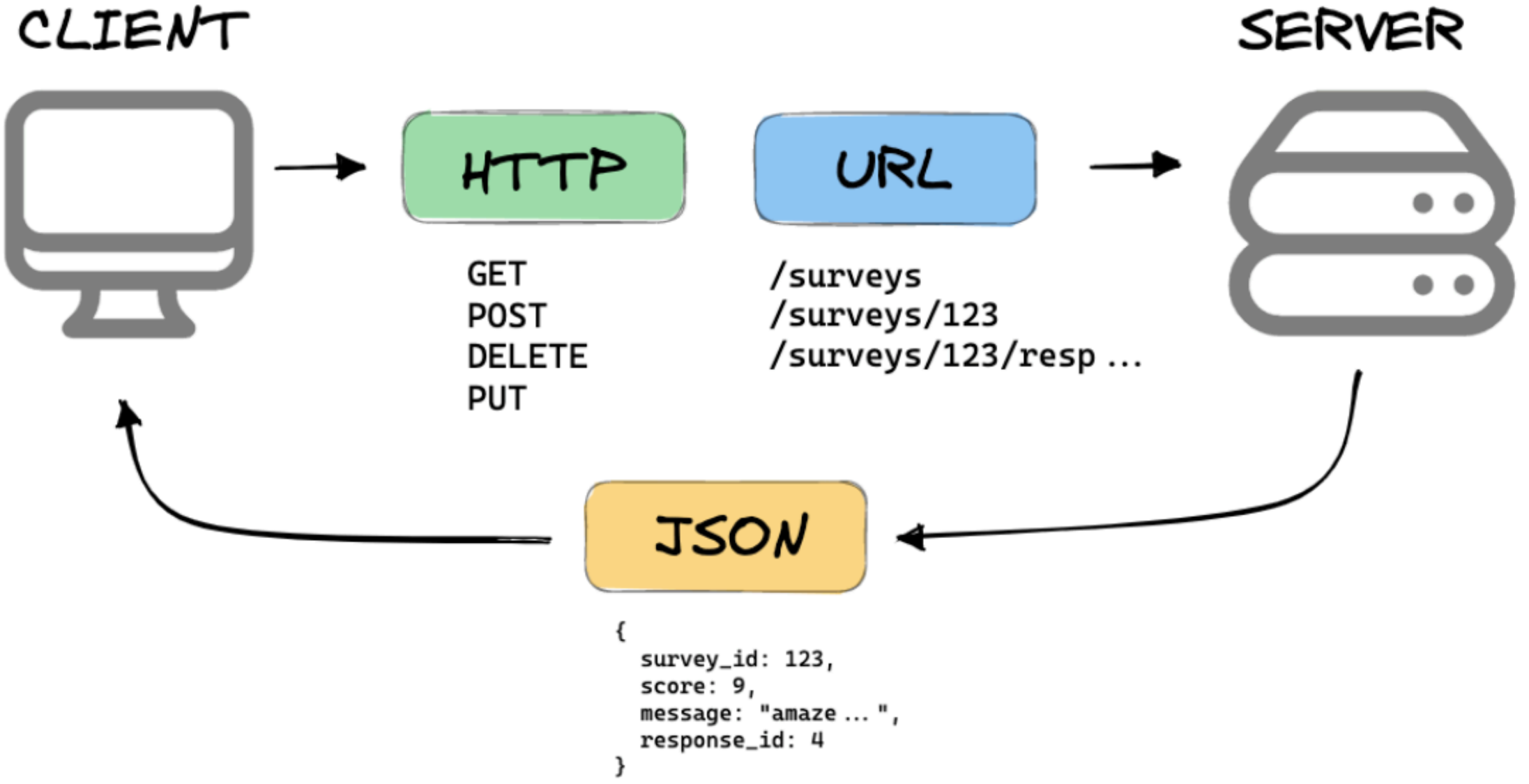
Retrofit HTTP Client

- In Kotlin, an interface is a collection of abstract methods and properties that define a common contract for classes that implement the interface
- An interface is basically some rules a class has to abide by.

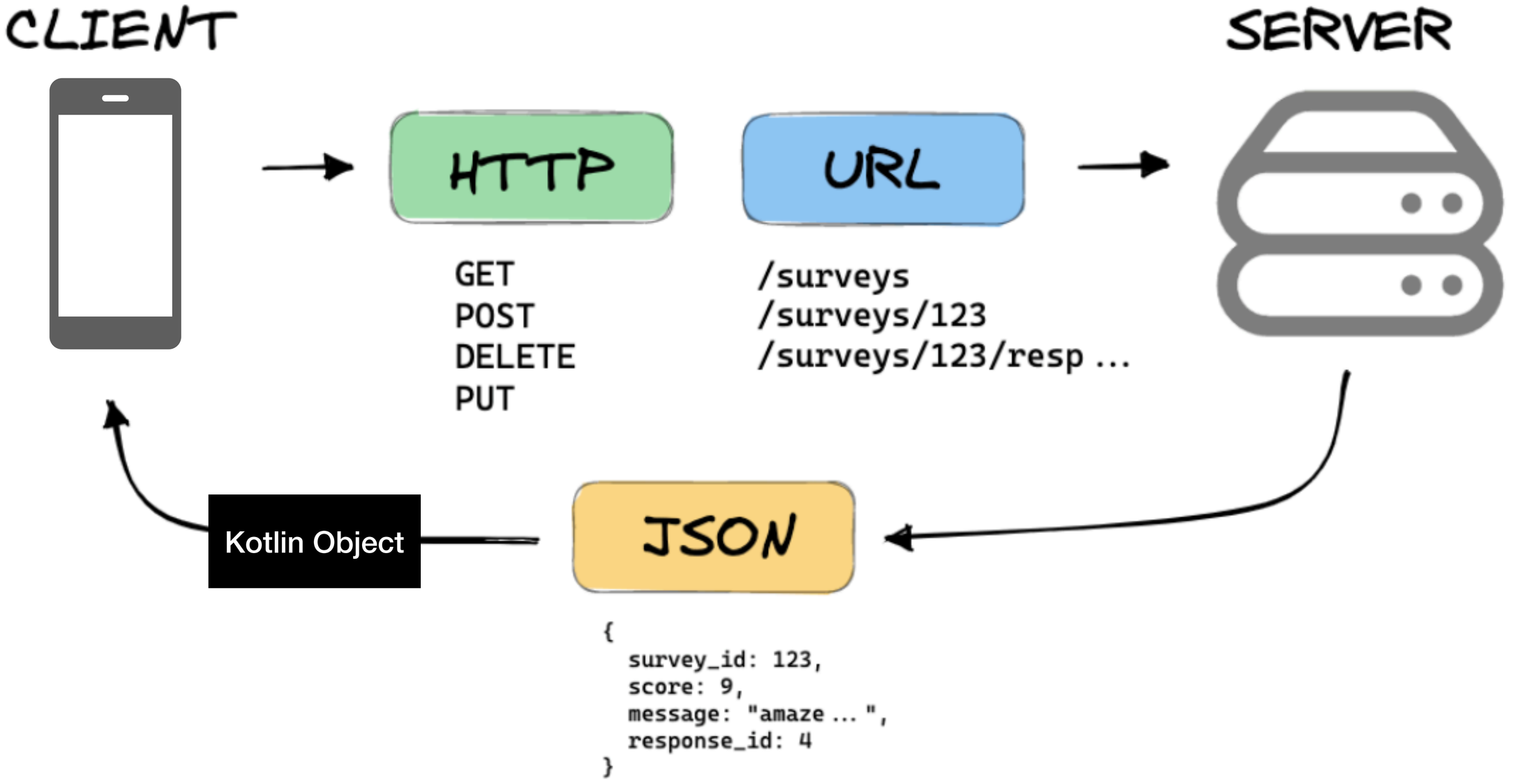
```kotlin
interface MyInterface {
    val test: Int

    fun foo() : String

    fun hello() : Unit
}

class InterfaceImp : MyInterface {
    override val test: Int = 0;

    override fun foo(): String {
        return "asd"
    }

    override fun hello() {}
}

fun main(args: Array<String>) {
    val obj = InterfaceImp()

    println("test = ${obj.test}")
    print("Calling hello(): ")

    obj.hello()

    print("Calling and printing foo(): ")
    println(obj.foo())
}
```


https://android-kotlin-fun-mars-server.appspot.com/photos
PROTOCOL HTTP: Get


Mars Photos
Success: 838 Mars photos retrieved
Image URLs from the server



https://mannhowie.com/rest-api



https://mannhowie.com/rest-api

If you application needs a backend

You can build the backend

Main components

Retrofit HTTP Client

API interface

Model /
Data Transfer Object

Retrofit Instance

Main components

Retrofit HTTP Client

API interface

```
interface CatFactsApi {
    @GET("/fact")
    suspend fun getFact(
    ): CatFact
}
```

Model /
Data Transfer Object

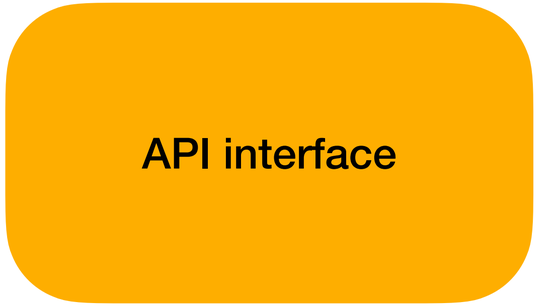
```
data class CatFact(
    @SerializedName("fact")
    val fact: String,
    @SerializedName("length")
    val length: Int
)
```

Retrofit Instance

```
class RetrofitInstance {
    private val baseURL = "https://catfact.ninja/";

    private val retrofitClient = Retrofit.Builder()
        .baseUrl(baseURL)
        .addConverterFactory(GsonConverterFactory.create())
        .build()

    val apiService = retrofitClient.create(CatFactsApi::class.java)
}
```

API interface

Retrofit framework



- The API interface represents all of the endpoints that your application will interact with
- The functions are suspended as they are “awaiting” answers
- If more endpoints should be called new functions should be provided

```
interface CatFactsApi {
    @GET("/fact")
    suspend fun getFact(
    ): CatFact
}
```

API interface

Retrofit framework

**Model /
Data Transfer Object**

- The model or data transfer object (DTO) represents the data that is provided by the API or to the API
- The @SerializedName provides the name of the JSON object that should be deserialized to the object property
  - Serialization refers to the proces of converting objects to strings de-serialization is opposite
- Not all data from the JSON response needs to be converted necessarily

```
data class CatFact(
    @SerializedName("fact")
    val fact: String,
    @SerializedName("length")
    val length: Int
)
```

```
{
  "fact": "British cat
owners spend roughly 550
million pounds yearly on
cat food.",
  "length": 71
}
```

API interface

Retrofit framework

- The RetrofitInstance represents the core client of the application.
- The RetrofitInstance builds the client with a
  - URL (baseURL)
  - ConverterFactory (Serialization & de-serialization)
  - The client creates objects from the JSON response

```
class RetrofitInstance {
    private val baseURL = "https://catfact.ninja/";

    private val retrofitClient = Retrofit.Builder()
        .baseUrl(baseURL)
        .addConverterFactory(GsonConverterFactory.create())
        .build()

    val apiService = retrofitClient.create(CatFactsApi::class.java)
}
```

Main components

Retrofit HTTP Client

```
val baseUrl = "https://gist.githubusercontent.com/"
val retrofitClient = Retrofit.Builder()
    .baseUrl(baseUrl)
    .addConverterFactory(GsonConverterFactory.create())
    .build()

val apiService = retrofitClient.create(MoviesAPI::class.java)

lifecycleScope.launch {
    try {
        val moviesFromApi = apiService.getMovies()
        movies = moviesFromApi;
    } catch (e: Exception) {
        println("Error fetching movies: ${e.message}")
    }
}
```

Example Movies GET