Retrieving data from the web

Applikationsudvikling

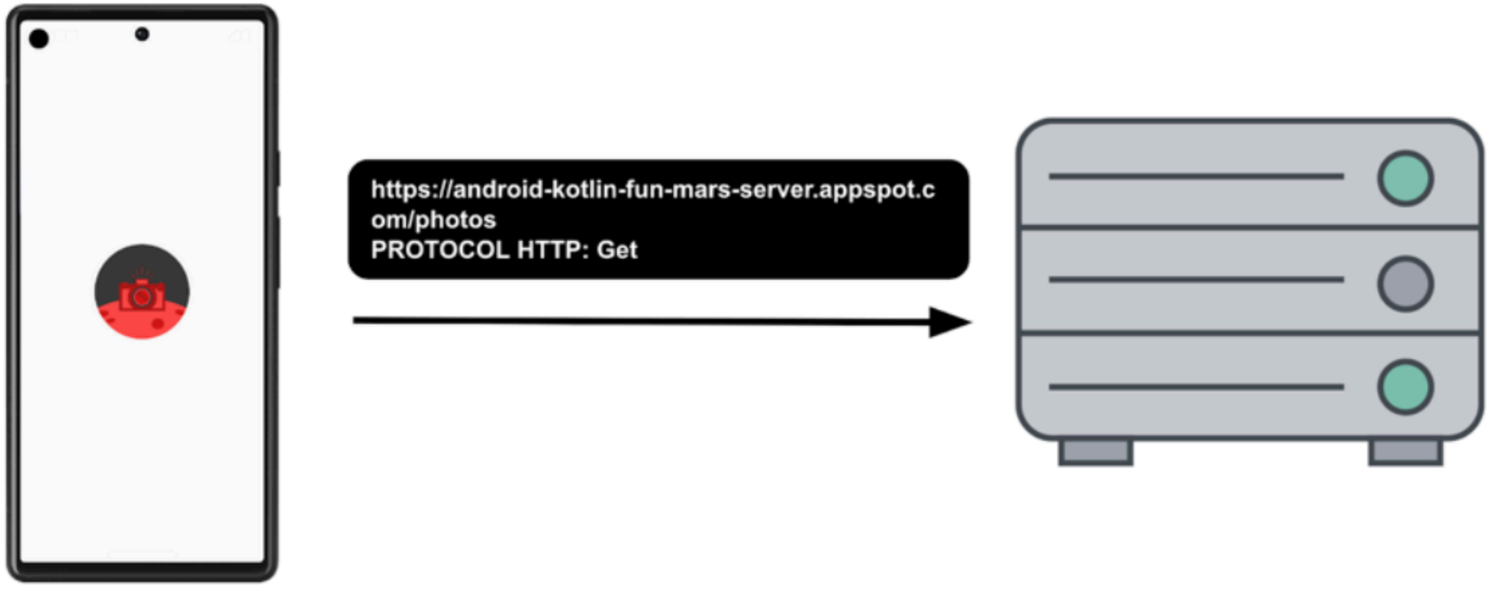
NIFR 2024

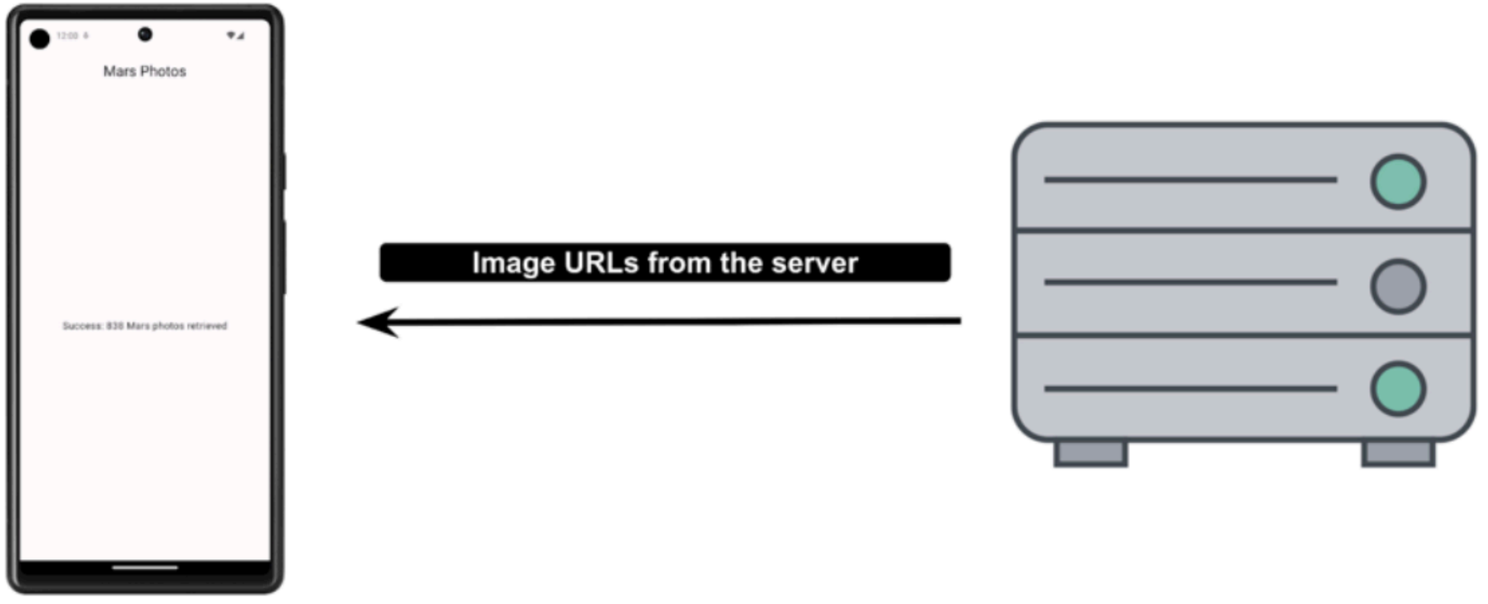
Agenda

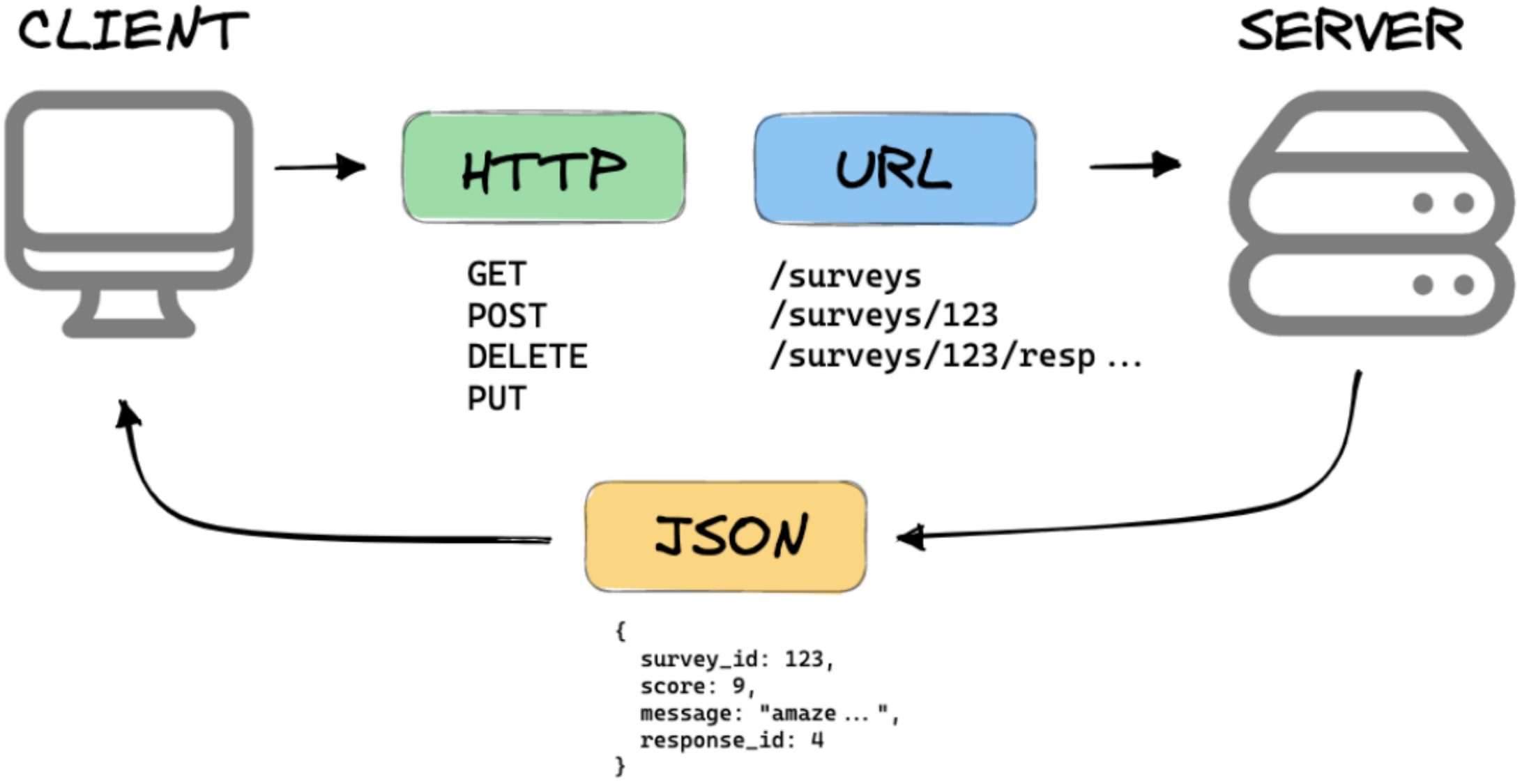
Retrieving data from the web

- Retrieving data from the web using retrofit
  - Retrofit outside of the Android framework
    - GET / POST
    - Serialization & deserialization of objects
  - Repository pattern
  - Error handling

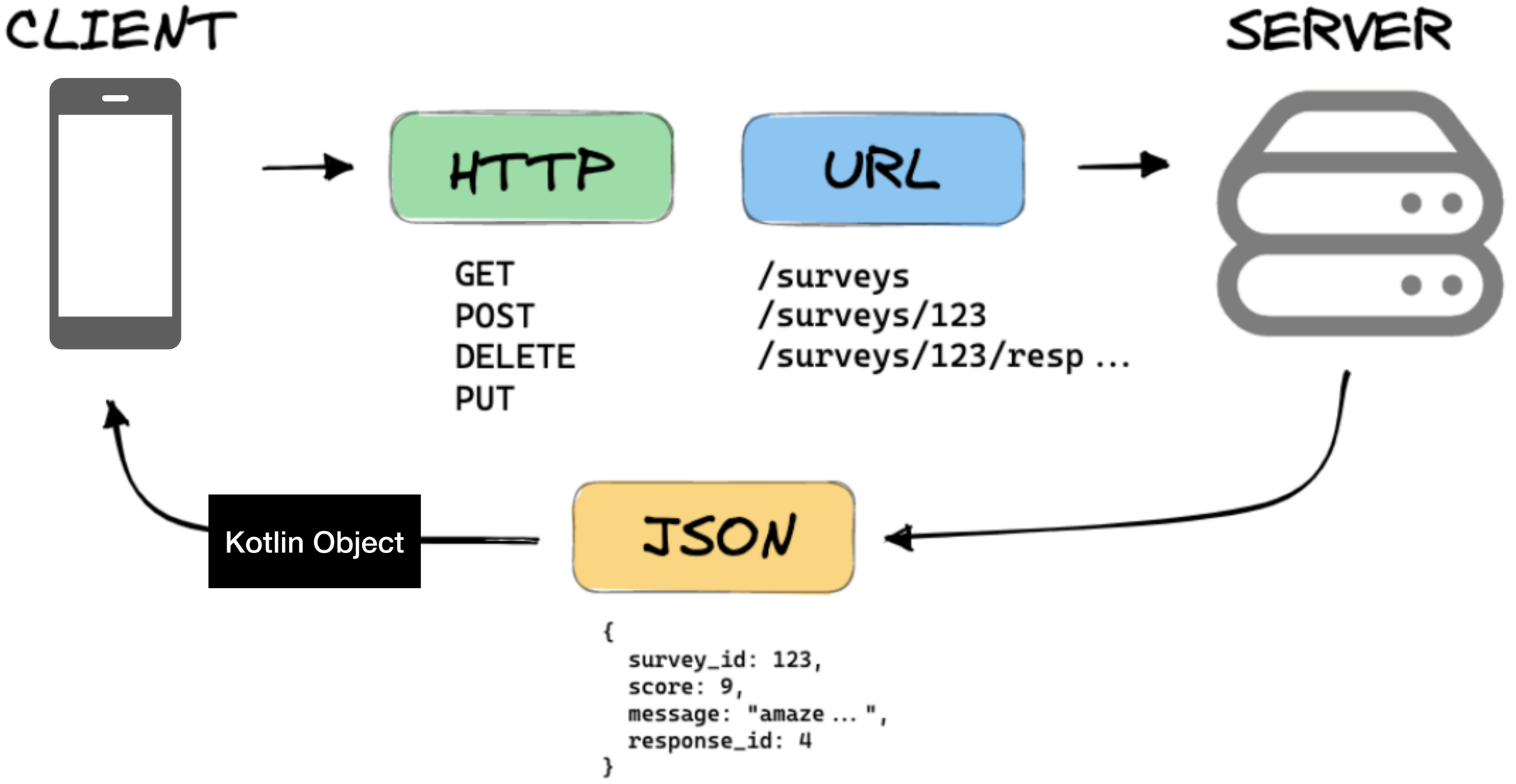
- Tomorrow Retrofit within the Android Framework
  - Co-routines & concurrency
  - Refactoring from example project to application


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

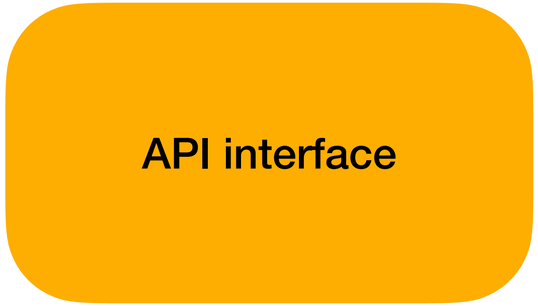
```
data class CatFact(
    @SerializedName("fact")
    val fact: String,
    @SerializedName("length")
    val length: Int
)
```

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 is populated with the required code by the Retrofit framework (Such as the fetch call code itself etc.)
- The API interface represents all of the end points that your application will interact with
- The functions are suspended as they are “awaiting” answers
- If more end points 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

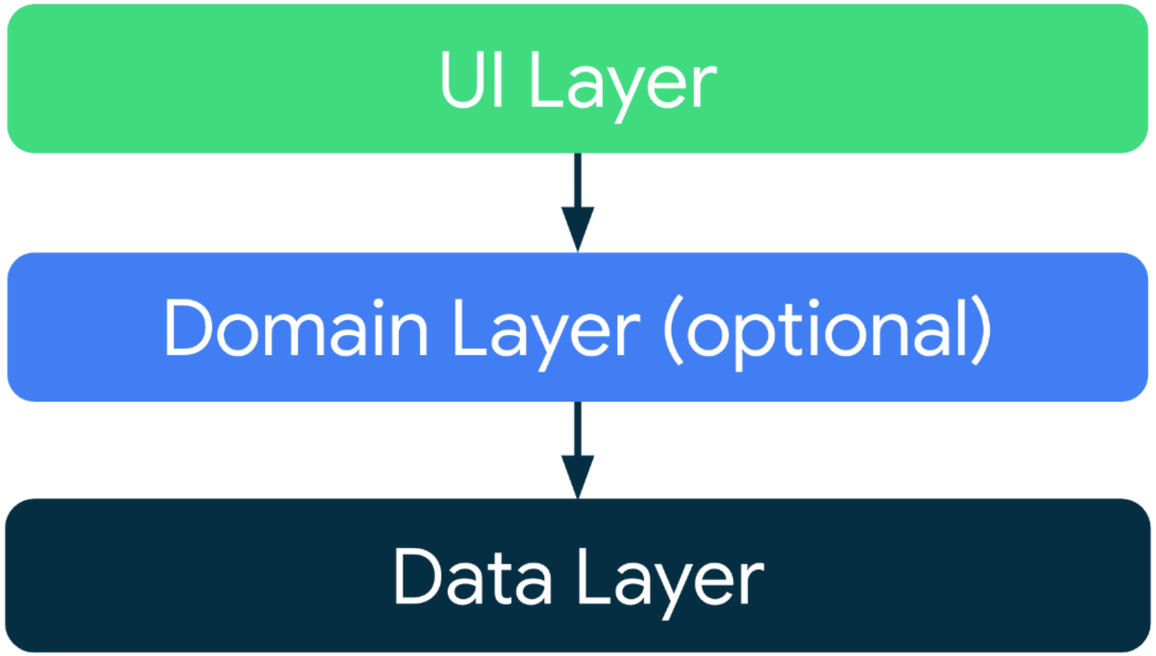
- The RetrofitInstance represents the core client of the application.
- The RetrofitInstance builds the client with a
  - URL (baseURL)
  - ConverterFactory (Serialization & de-serialization)
  - The client creates objects from the JSON response

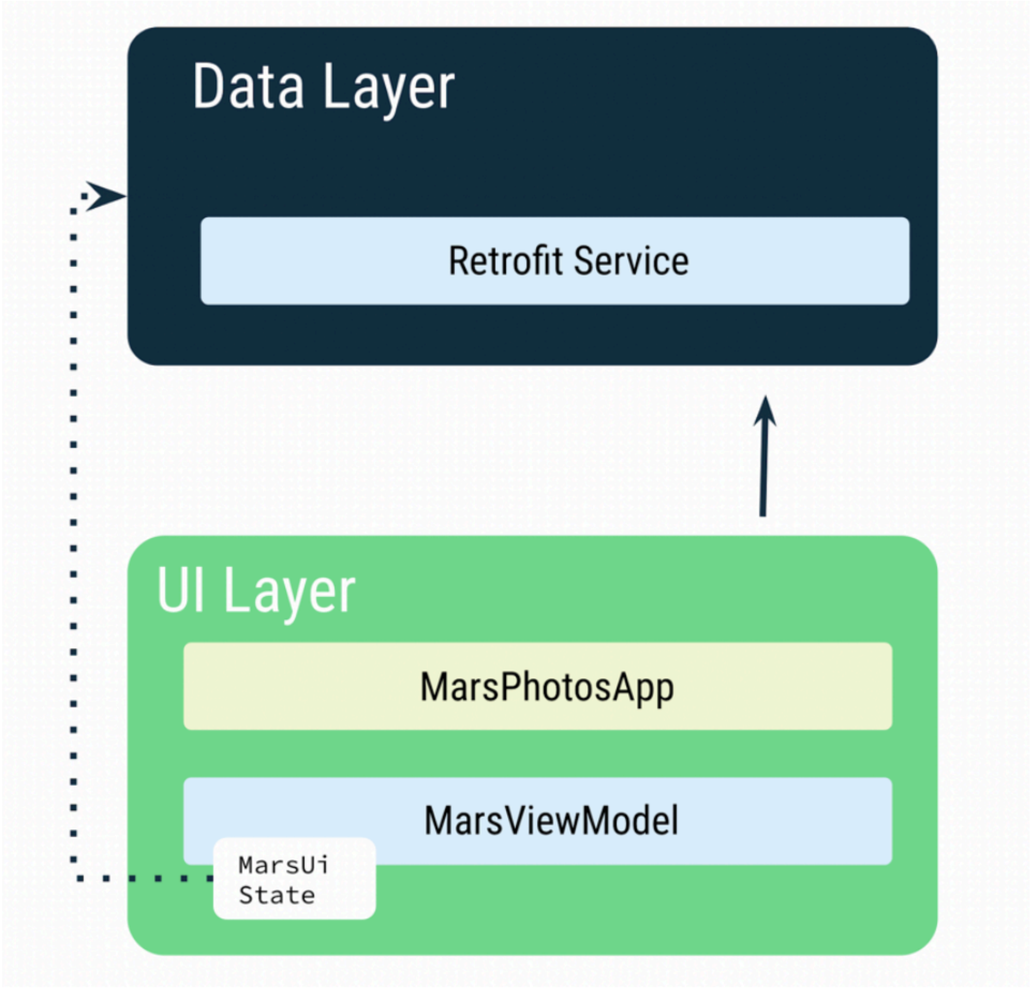
```
class RetrofitInstance {
    private val baseURL = "https://catfact.ninja/";

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

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

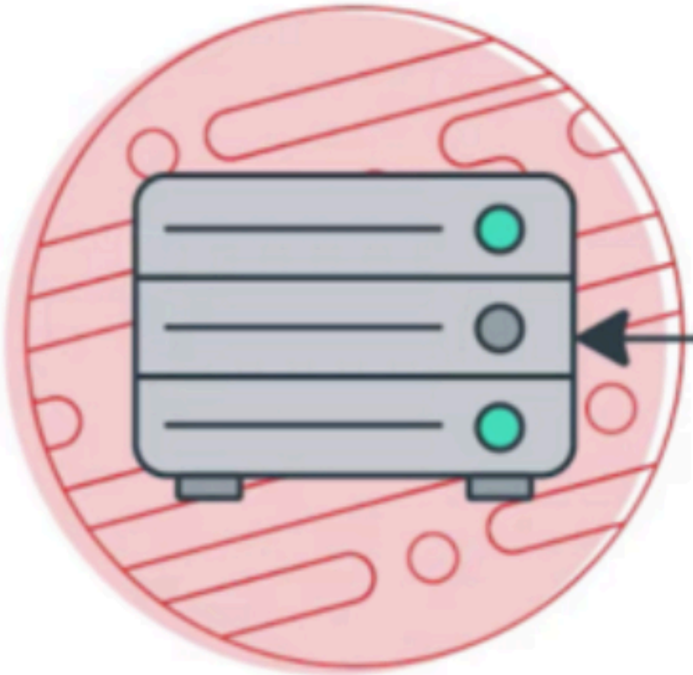
Example CatFacts GET

Example ReqRes POST

Returning a call


UI Layer
Domain Layer (optional)
Data Layer


Data Layer
Retrofit Service
UI Layer
MarsPhotosApp
MarsViewModel
MarsUi State



Mars Web Service

Retrofit

Converter Library

MutableState

ViewModel

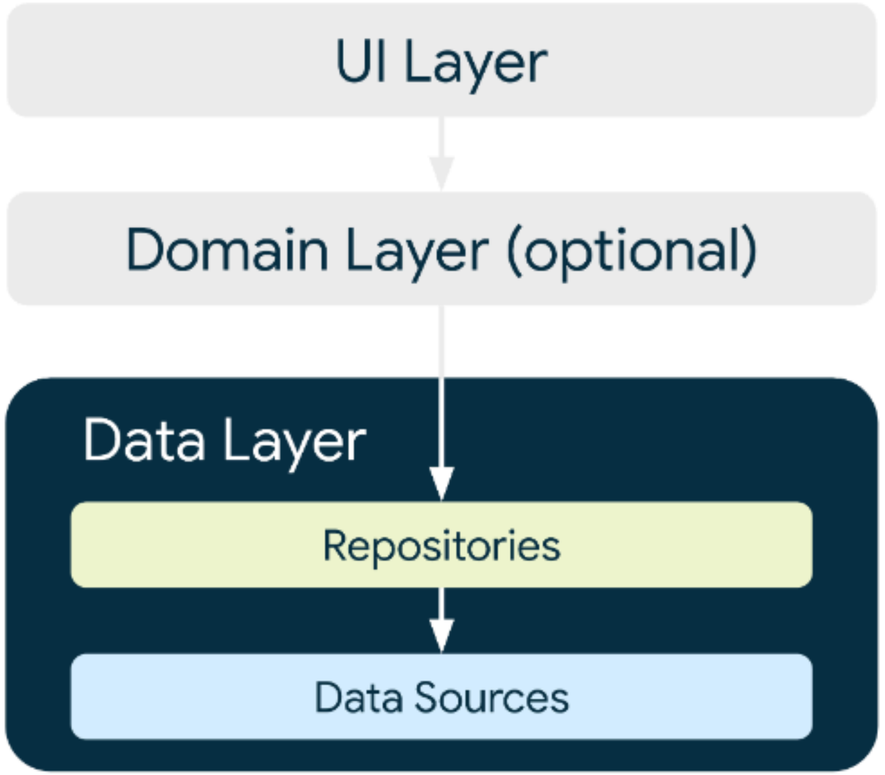

Make it work, make it right, make it fast.
Kent Beck
quotefancy

The Repository pattern

Retrofit

Within the Android Framework

- The data layer is often represented as a pattern called the ‘repository’ pattern
- The repository pattern is a type of a ‘facade’ pattern
- We are not implementing the full pattern as this requires:
  - Dependency Injection
  - More interface implementation
- But the core idea is represented



https://developer.android.com/topic/architecture

Retrofit

Within the Android Framework