

Retrieving data from the web

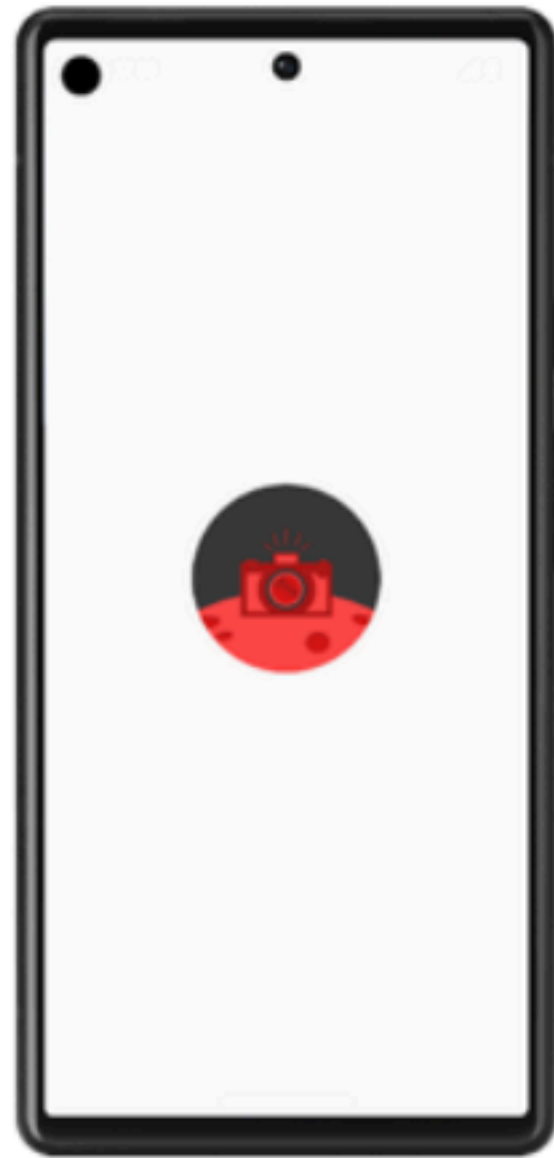
Applikationsudvikling

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.

```
3  I↓ interface MyInterface {
4  I↓     val test: Int
5
6  I↓     fun foo() : String
7
8  I↓     fun hello() : Unit
9     }
10
11     class InterfaceImp : MyInterface {
12  I↑         override val test: Int = 0;
13
14  I↑         override fun foo(): String {
15             return "asd"
16         }
17
18  I↑         override fun hello() {}
19     }
20
21  ▶ fun main(args: Array<String>) {
22         val obj = InterfaceImp()
23
24         println("test = ${obj.test}")
25         print("Calling hello(): ")
26
27         obj.hello()
28
29         print("Calling and printing foo(): ")
30         println(obj.foo())
31     }
```



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

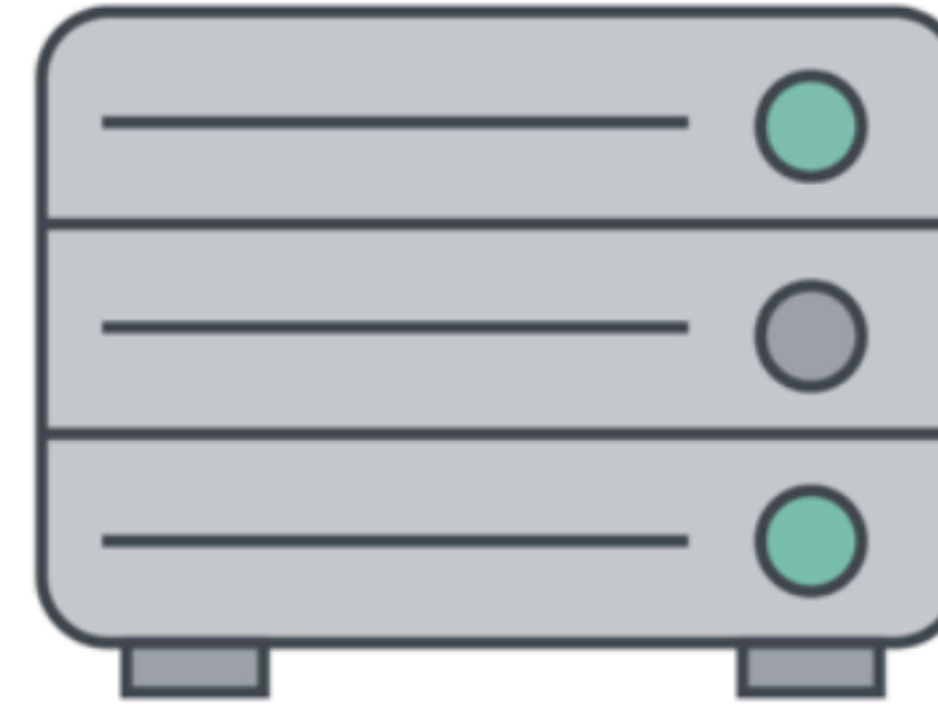
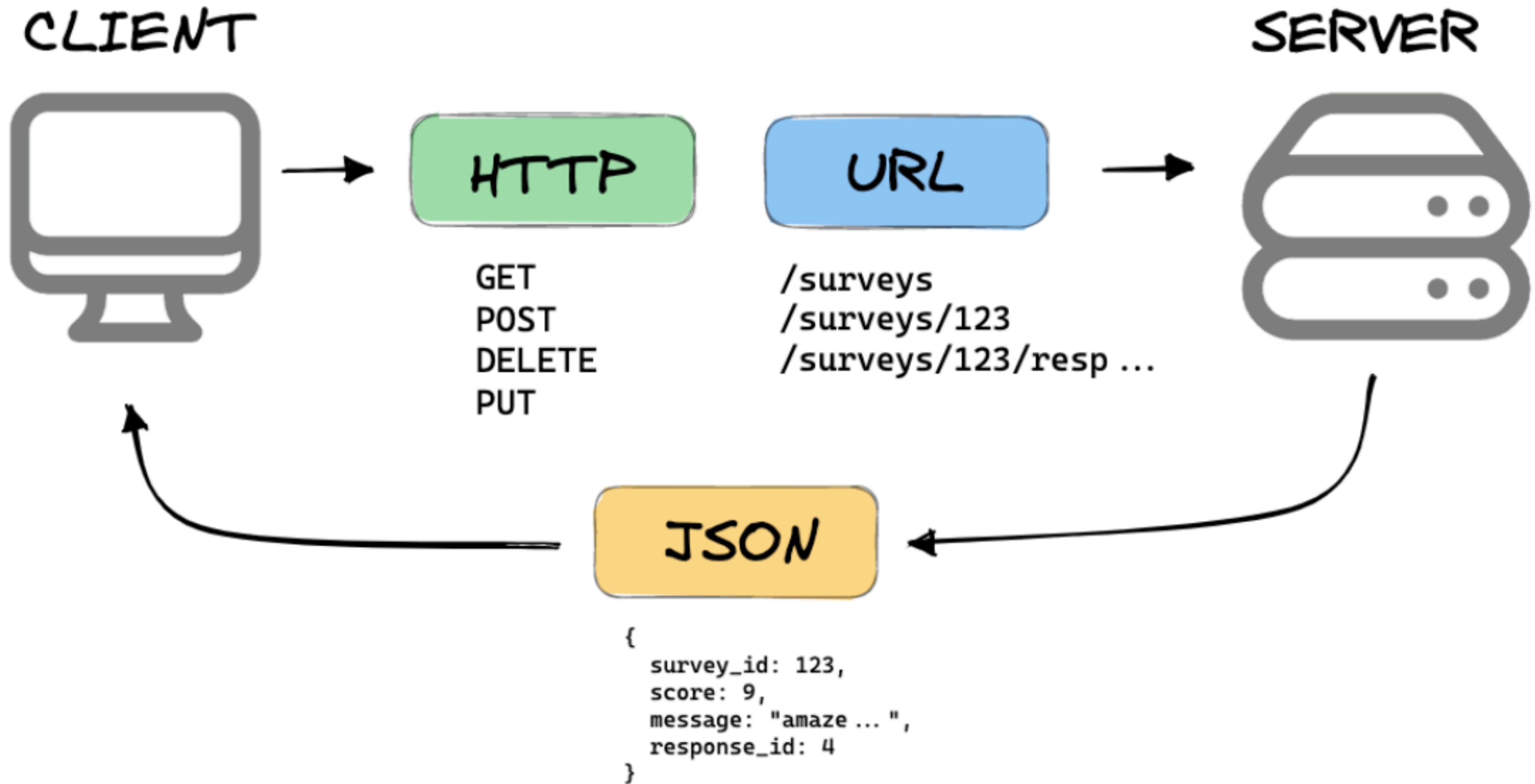
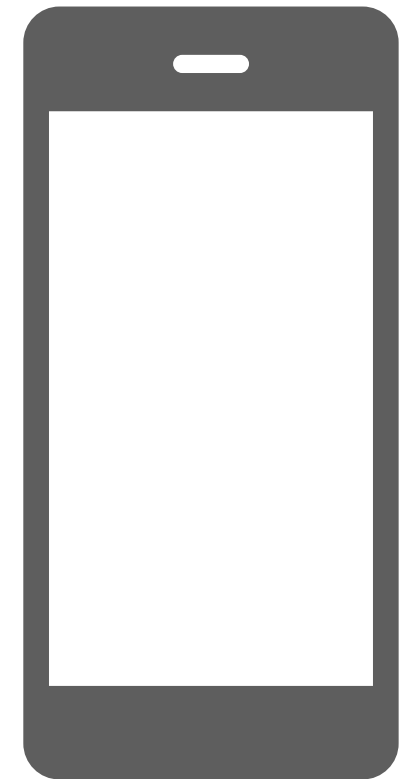


Image URLs from the server





CLIENT



HTTP

GET
POST
DELETE
PUT

URL

/surveys
/surveys/123
/surveys/123/resp ...



SERVER



Kotlin Object



JSON

```
{  
  survey_id: 123,  
  score: 9,  
  message: "amaze ... ",  
  response_id: 4  
}
```



If your 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

API interface

```
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

Retrofit Instance

- 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