Android Dev Academy 2024/2025

Android Networking

Powering Your App with API

Be counted! Scan now. ✔️

Agenda

01 Networking basics

02 Retrofit

03 APIService, Models, Repository

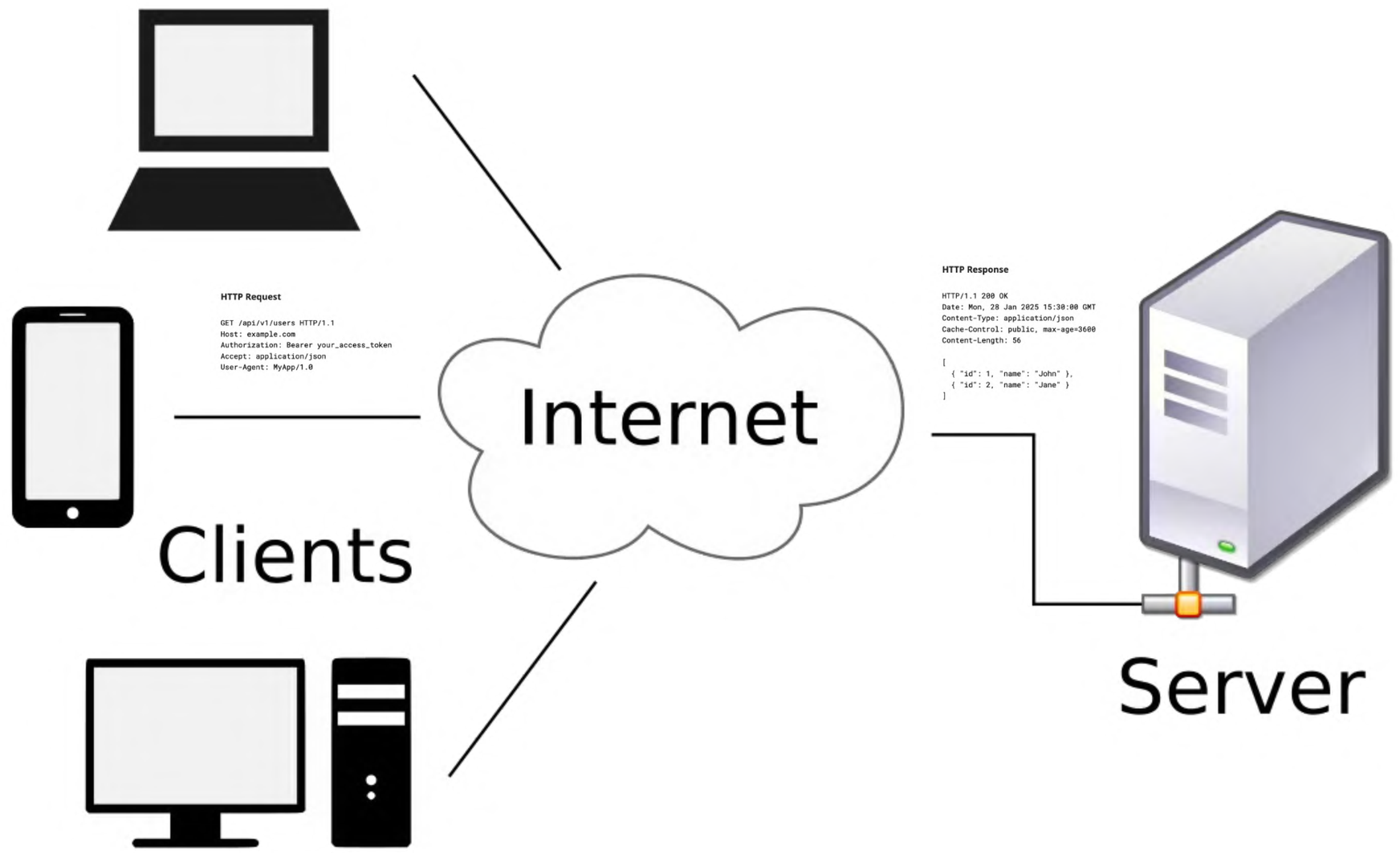
04 ViewModel and State Handling

05 Token Handling, Interceptors, Image Loading

05 Homework, Q&A

1

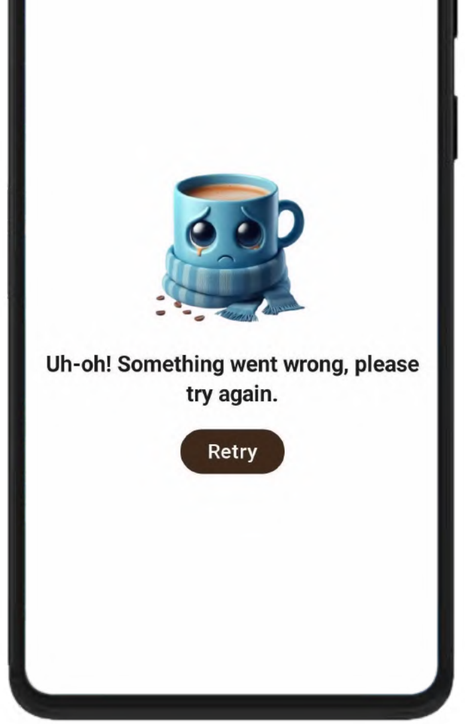
Networking
Back to basics

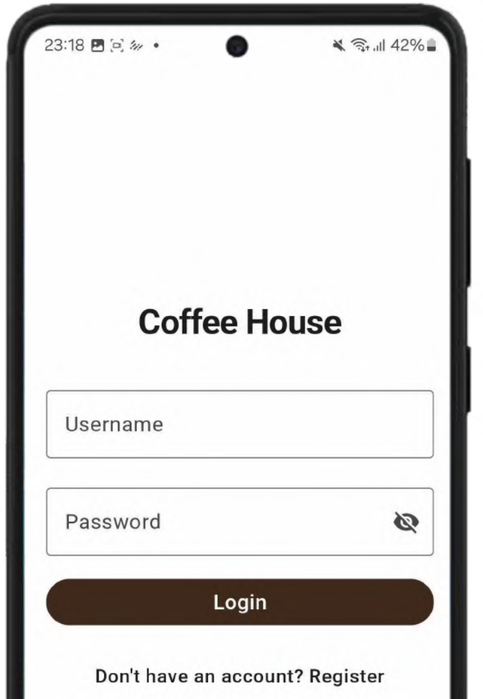

HTTP Request
GET /api/v1/users HTTP/1.1
Host: example.com
Authorization: Bearer your_access_token
Accept: application/json
User-Agent: MyApp/1.0
Internet
HTTP Response
HTTP/1.1 200 OK
Date: Mon, 28 Jan 2025 15:30:00 GMT
Content-Type: application/json
Cache-Control: public, max-age=3600
Content-Length: 56
[
{ "id": 1, "name": "John" },
{ "id": 2, "name": "Jane" }
]
Clients
Server

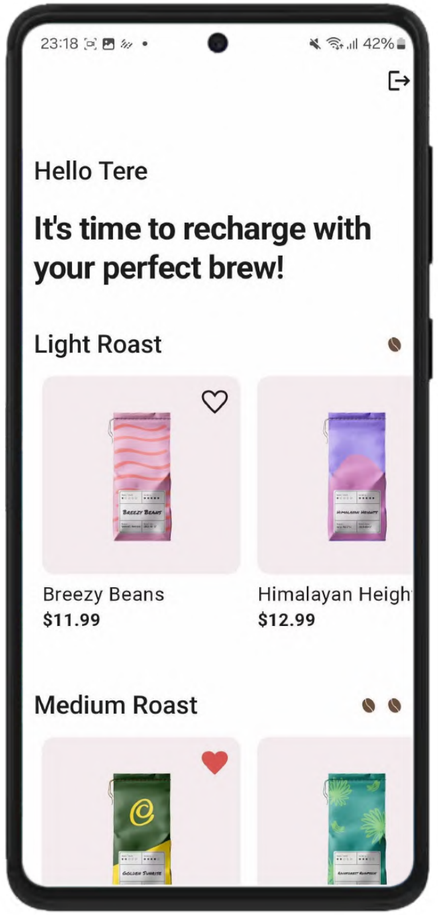
Try Pitch

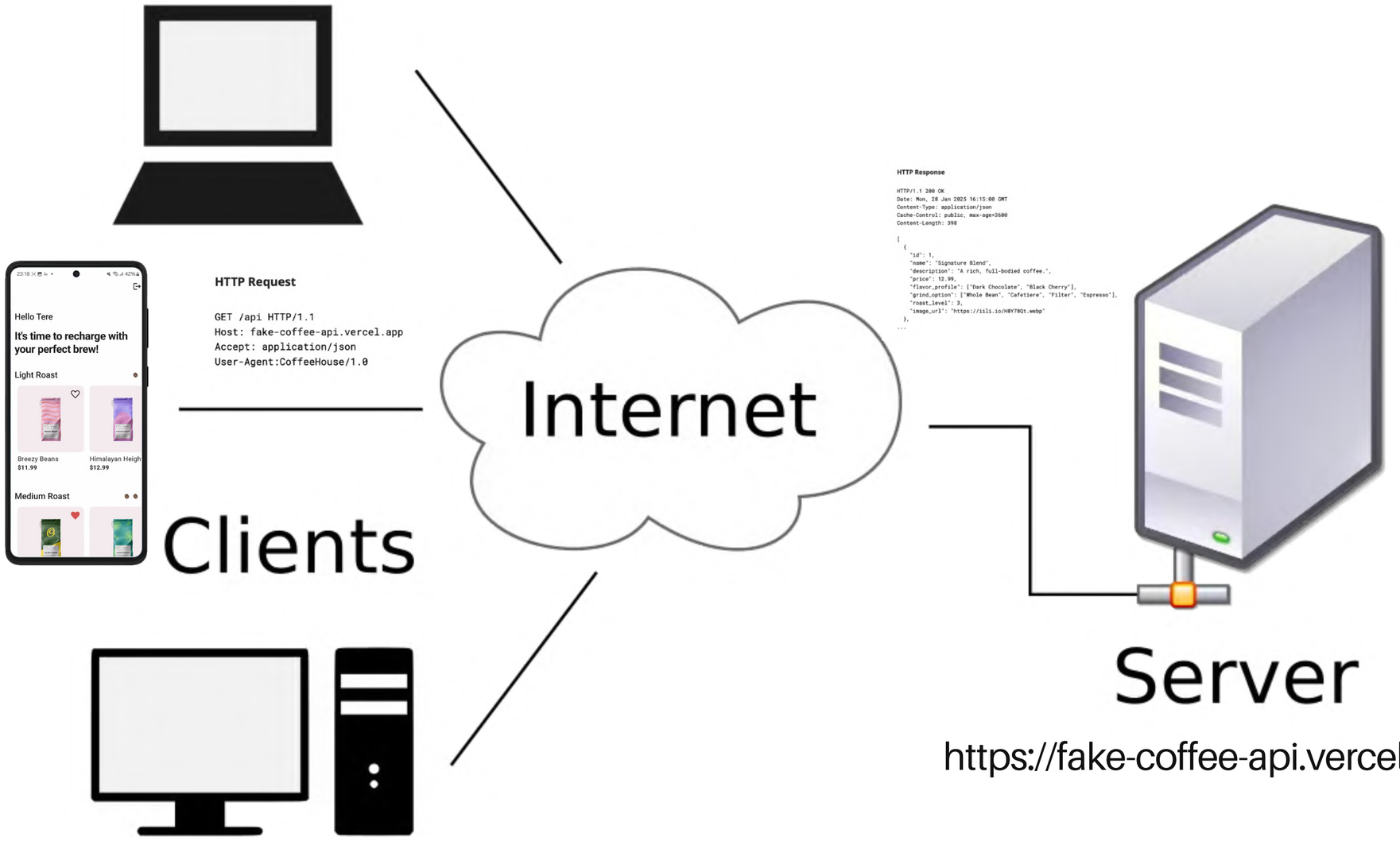
Coffee House

https://github.com/TeaUmily/CoffeeHouse







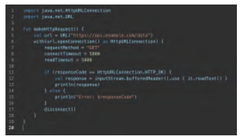

HTTP Request
GET /api HTTP/1.1
Host: fake-coffee-api.vercel.app
Accept: application/json
User-Agent:CoffeeHouse/1.0
HTTP Response
HTTP/1.1 200 OK
Date: Mon, 28 Jan 2025 16:15:00 GMT
Content-Type: application/json
Cache-Control: public, max-age=3600
Content-Length: 398
Hello Tere
It's time to recharge with your perfect brew!
Light Roast
Breezy Beans
$11.99
Himalayan Heigh
$12.99
Medium Roast
Internet
Clients
Server
https://fake-coffee-api.vercel.app

Client part

Low level

HttpURLConnection

- Part of the Java standard library and Android SDK for making HTTP requests
- Complete control over HTTP request
-



Mid level

OkHttpClient

- Open source library developed by Square
- include **interceptors** for monitoring and modifying requests/responses, **connection pooling** for improved performance, **automatic caching** to reduce network calls, timeouts handling...

High level

Retrofit

- **THE Library** for Networking in Android, also developed by Square, well-supported and maintained
- Facade over OkHttpClient
- Support for Coroutines & better threading
- Very easy to integrate
- Highly optimized, reduces a lot of boilerplate code

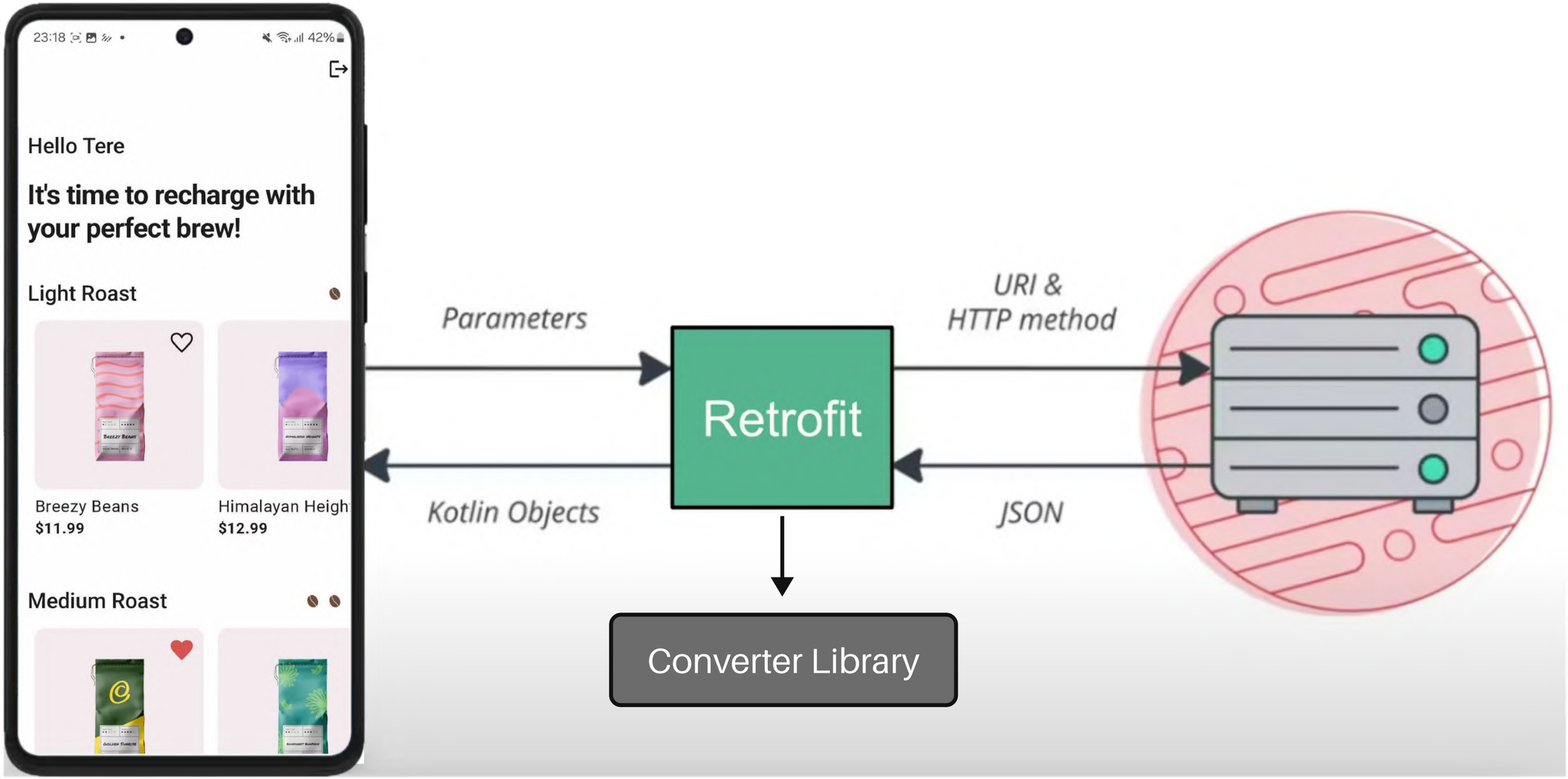
2

Retrofit

libs.version.toml

[libraries]

"com.squareup.**retrofit2**:retrofit"


23:18
42%
Hello Tere
It's time to recharge with your perfect brew!
Light Roast
Breezy Beans
$11.99
Himalayan Heigh
$12.99
Medium Roast
Parameters
Kotlin Objects
Retrofit
URI &
HTTP method
JSON
Converter Library

libs.version.toml

[libraries]

"org.jetbrains.kotlinx:**kotlinx-serialization-json**"

"com.jakewharton.retrofit:**retrofit2-kotlinx-serialization-converter**"

"com.squareup.okhttp3:**okhttp**"

[plugins]

"org.jetbrains.kotlin.plugin.serialization"

```
object RetrofitCoffeeInstance {
    private const val BASE_URL = "https://fake-coffee-api.vercel.app"

    val api: CoffeeApiService by lazy {
        val json = Json {
            ignoreUnknownKeys = true
        }

        val contentType = "application/json".toMediaType()

        val retrofit = Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(json.asConverterFactory(contentType))
            .build()

       retrofit.create(CoffeeApiService::class.java)
    }
}
```

```
object RetrofitCoffeeInstance {
    private const val BASE_URL = "https://fake-coffee-api.vercel.app"

    val api: CoffeeApiService by lazy {
        val json = Json {
            ignoreUnknownKeys = true
        }

        val contentType = "application/json".toMediaType()

        val retrofit = Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(json.asConverterFactory(contentType))
            .build()

       retrofit.create(CoffeeApiService::class.java)
    }
}
```

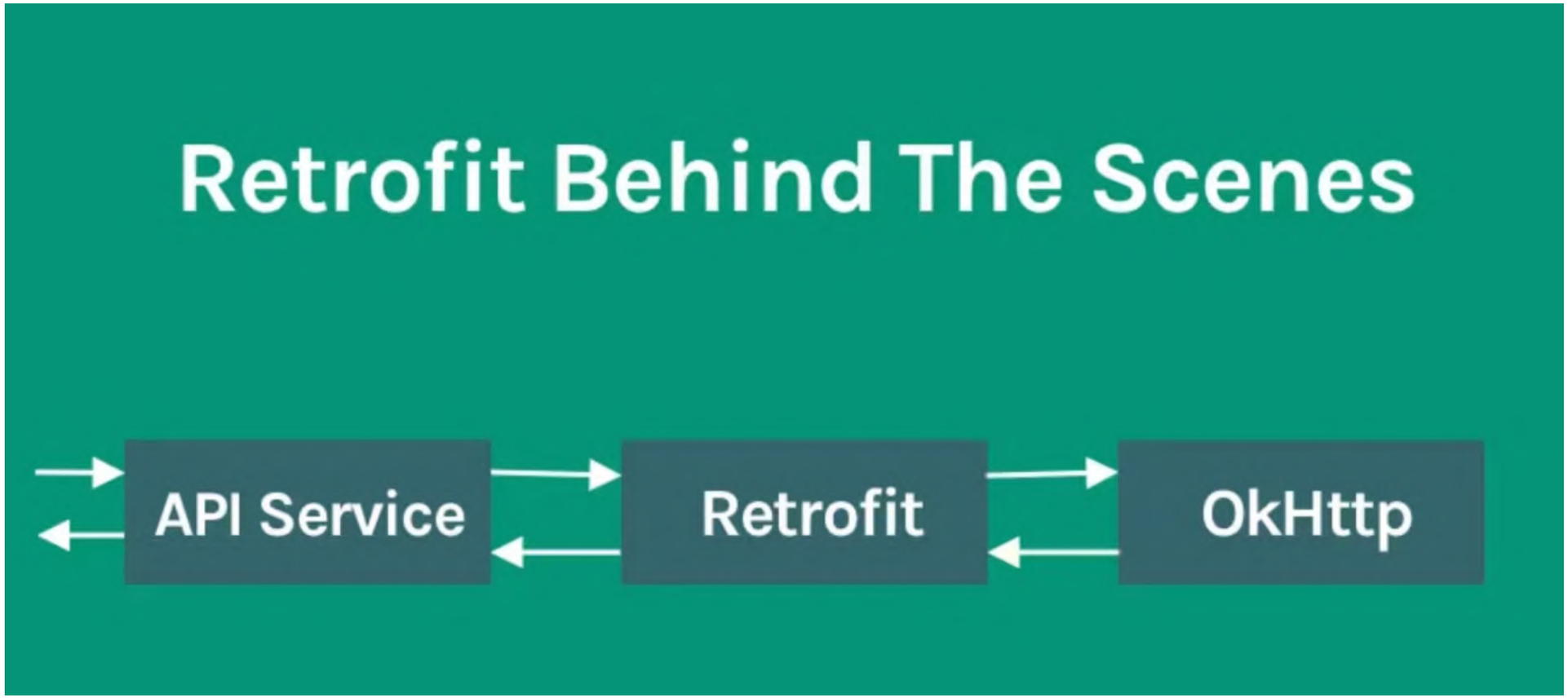
```
object RetrofitCoffeeInstance {
    private const val BASE_URL = "https://fake-coffee-api.vercel.app"

    val api: CoffeeApiService by lazy {
        val json = Json {
            ignoreUnknownKeys = true
        }

        val contentType = "application/json".toMediaType()

        val retrofit = Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(json.asConverterFactory(contentType))
            .build()

       retrofit.create(CoffeeApiService::class.java)
    }
}
```

```kotlin
object RetrofitCoffeeInstance {
    private const val BASE_URL = "https://fake-coffee-api.vercel.app"

    val api: CoffeeApiService by lazy {
        val json = Json {
            ignoreUnknownKeys = true
        }

        val contentType = "application/json".toMediaType()

        val retrofit = Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(json.asConverterFactory(contentType))
            .build()

        retrofit.create(CoffeeApiService::class.java)
    }
}

interface CoffeeApiService { }
```

3

APIService, Models, Repository

Repository



```
interface CoffeeApiService {
    @GET("api")
    suspend fun getCoffees(): List<CoffeeItem>
}
```

```
interface CoffeeApiService {
    @GET("api")
    suspend fun getCoffees(): List<CoffeeItem>
}
```

```
interface CoffeeApiService {
    @GET("api")
    suspend fun getCoffees(): List<CoffeeItem>
}
```

```
interface CoffeeApiService {
    @GET("api")
    suspend fun getCoffees(): List<CoffeeItem>
}
```

```
interface CoffeeApiService {
    @GET("api")
    suspend fun getCoffees(): List<CoffeeItem>
}
```

```kotlin
@Serializable
data class CoffeeItem(
    @SerialName("_id") val id: String,
    @SerialName("id") val order: Int,
    @SerialName("description") val description: String,
    @SerialName("flavor_profile") val flavorProfile: List<String>,
    @SerialName("grind_option") val grindOption: List<String>,
    @SerialName("image_url") val imageUrl: String,
    @SerialName("name") val name: String,
    @SerialName("price") val price: Double,
    @SerialName("region") val region: String,
    @SerialName("roast_level") val roastLevel: Int,
    @SerialName("weight") val weight: Int
)
```

HTTP Response

```
HTTP/1.1 200 OK
Date: Mon, 28 Jan 2025 16:15:08 GMT
Content-Type: application/json
Cache-Control: public, max-age=3600
Content-Length: 398

[
  {
    "id": 1,
    "name": "Signature Blend",
    "description": "A rich, full-bodied coffee.",
    "price": 12.99,
    "flavor_profile": ["Dark Chocolate", "Black Cherry"],
    "grind_option": ["Whole Bean", "Cafetiere", "Filter", "Espresso"],
    "roast_level": 3,
    "image_url": "https://ili1.io/H8Y78Qt.webp"
  },
...
```

```kotlin
interface CoffeeRepository {
    suspend fun getCoffees(): List<CoffeeItem>
}

class CoffeeRepositoryImpl : CoffeeRepository {
    override suspend fun getCoffees(): List<CoffeeItem> {
        return RetrofitCoffeeInstance.api.getCoffees()
    }
}
```

4

ViewModel and State Handling

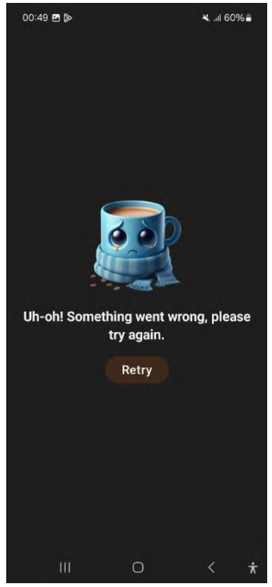
```
class CoffeeCatalogViewModel(
    private val coffeeRepository: CoffeeRepository
) : ViewModel() {

    var coffeeCatalogUiState: CoffeeCatalogUiState by mutableStateOf(CoffeeCatalogUiState.Loading)

    fun getCoffeeCatalog() {
        viewModelScope.launch {
            try {
                val listResult = coffeeRepository.getCoffees()
                coffeeCatalogUiState = CoffeeCatalogUiState.Success(
                    coffees = listResult,
                    isRefreshing = false
                )
            } catch (e: Exception) {
                    CoffeeCatalogUiState.Error
                }
            }
        }
    }
}
```

```
class CoffeeCatalogViewModel(
    private val coffeeRepository: CoffeeRepository
) : ViewModel() {

    var coffeeCatalogUiState: CoffeeCatalogUiState by mutableStateOf(CoffeeCatalogUiState.Loading)

    fun getCoffeeCatalog() {
        viewModelScope.launch {
            try {
                val listResult = coffeeRepository.getCoffees()
                coffeeCatalogUiState = CoffeeCatalogUiState.Success(
                    coffees = listResult,
                    isRefreshing = false
                )
            } catch (e: Exception) {
                    CoffeeCatalogUiState.Error
                }
            }
        }
    }
}
```

```
class CoffeeCatalogViewModel(
    private val coffeeRepository: CoffeeRepository
) : ViewModel() {

    var coffeeCatalogUiState: CoffeeCatalogUiState by mutableStateOf(CoffeeCatalogUiState.Loading)

    fun getCoffeeCatalog() {
        viewModelScope.launch {
            try {
                val listResult = coffeeRepository.getCoffees()
                coffeeCatalogUiState = CoffeeCatalogUiState.Success(
                    coffees = listResult,
                    isRefreshing = false
                )
            } catch (e: Exception) {
                    CoffeeCatalogUiState.Error
                }
            }
        }
    }
}
```

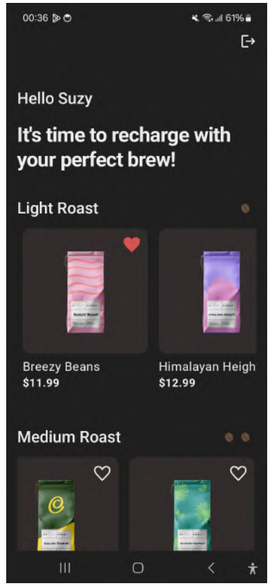
```
class CoffeeCatalogViewModel(
    private val coffeeRepository: CoffeeRepository
) : ViewModel() {

    var coffeeCatalogUiState: CoffeeCatalogUiState by mutableStateOf(CoffeeCatalogUiState.Loading)

    fun getCoffeeCatalog() {
        viewModelScope.launch {
            try {
                val listResult = coffeeRepository.getCoffees()
                coffeeCatalogUiState = CoffeeCatalogUiState.Success(
                    coffees = listResult,
                    isRefreshing = false
                )
            } catch (e: Exception) {
                    CoffeeCatalogUiState.Error
                }
            }
        }
    }
}
```

```
class CoffeeCatalogViewModel(
    private val coffeeRepository: CoffeeRepository
) : ViewModel() {

    var coffeeCatalogUiState: CoffeeCatalogUiState by mutableStateOf(CoffeeCatalogUiState.Loading)

    fun getCoffeeCatalog() {
        viewModelScope.launch {
            try {
                val listResult = coffeeRepository.getCoffees()
                coffeeCatalogUiState = CoffeeCatalogUiState.Success(
                    coffees = listResult,
                    isRefreshing = false
                )
            } catch (e: Exception) {
                    CoffeeCatalogUiState.Error
                }
            }
        }
    }
}
```



Try Pitch

```
class CoffeeCatalogViewModel(
    private val coffeeRepository: CoffeeRepository
) : ViewModel() {

    var coffeeCatalogUiState: CoffeeCatalogUiState by mutableStateOf(CoffeeCatalogUiState.Loading)

    fun getCoffeeCatalog() {
        viewModelScope.launch {
            try {
                val listResult = coffeeRepository.getCoffees()
                coffeeCatalogUiState = CoffeeCatalogUiState.Success(
                    coffees = listResult,
                    isRefreshing = false
                )
            } catch (e: Exception) {
                    CoffeeCatalogUiState.Error
                }
            }
        }
    }
}
```

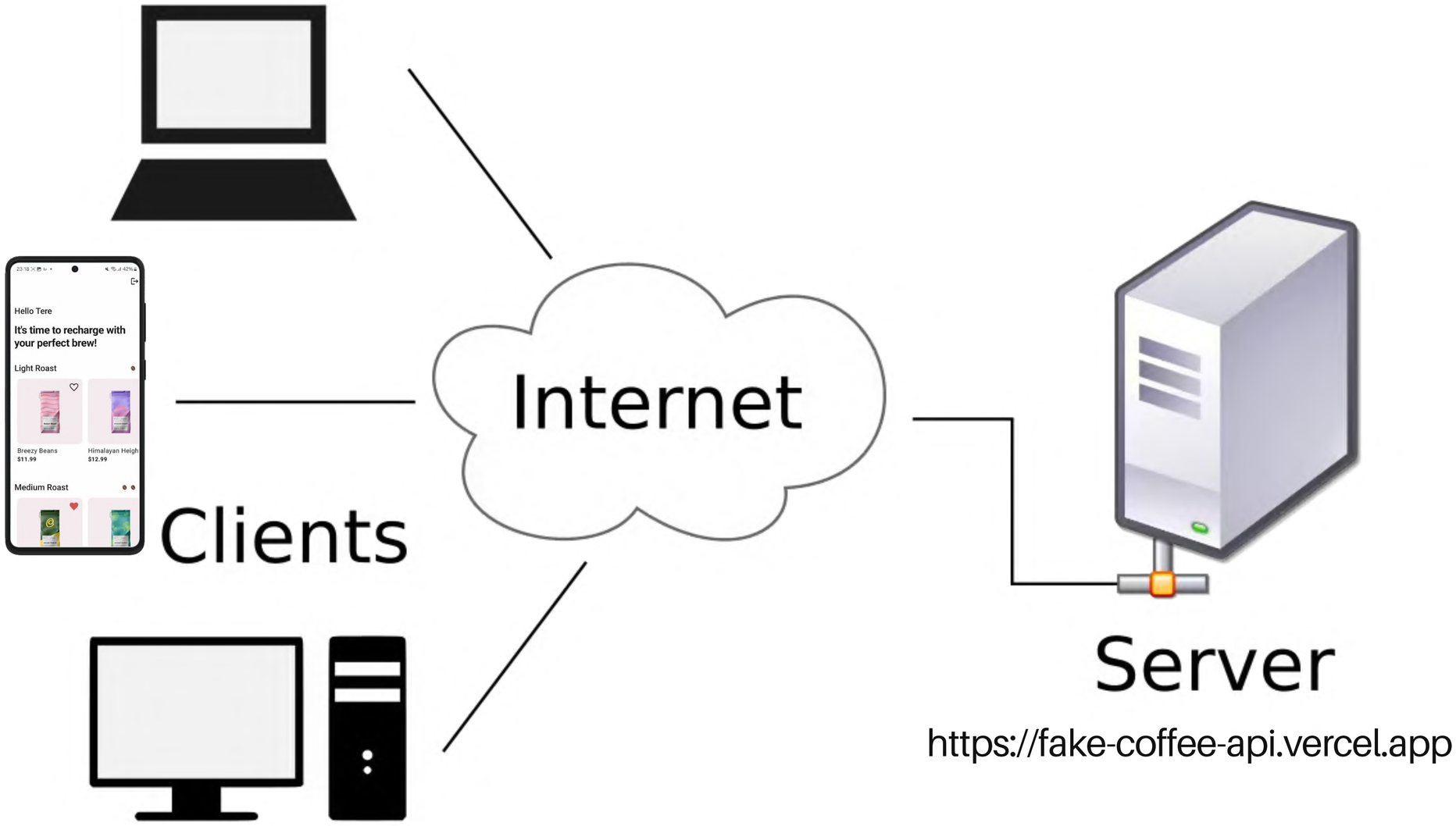


Retrofit? Handled.

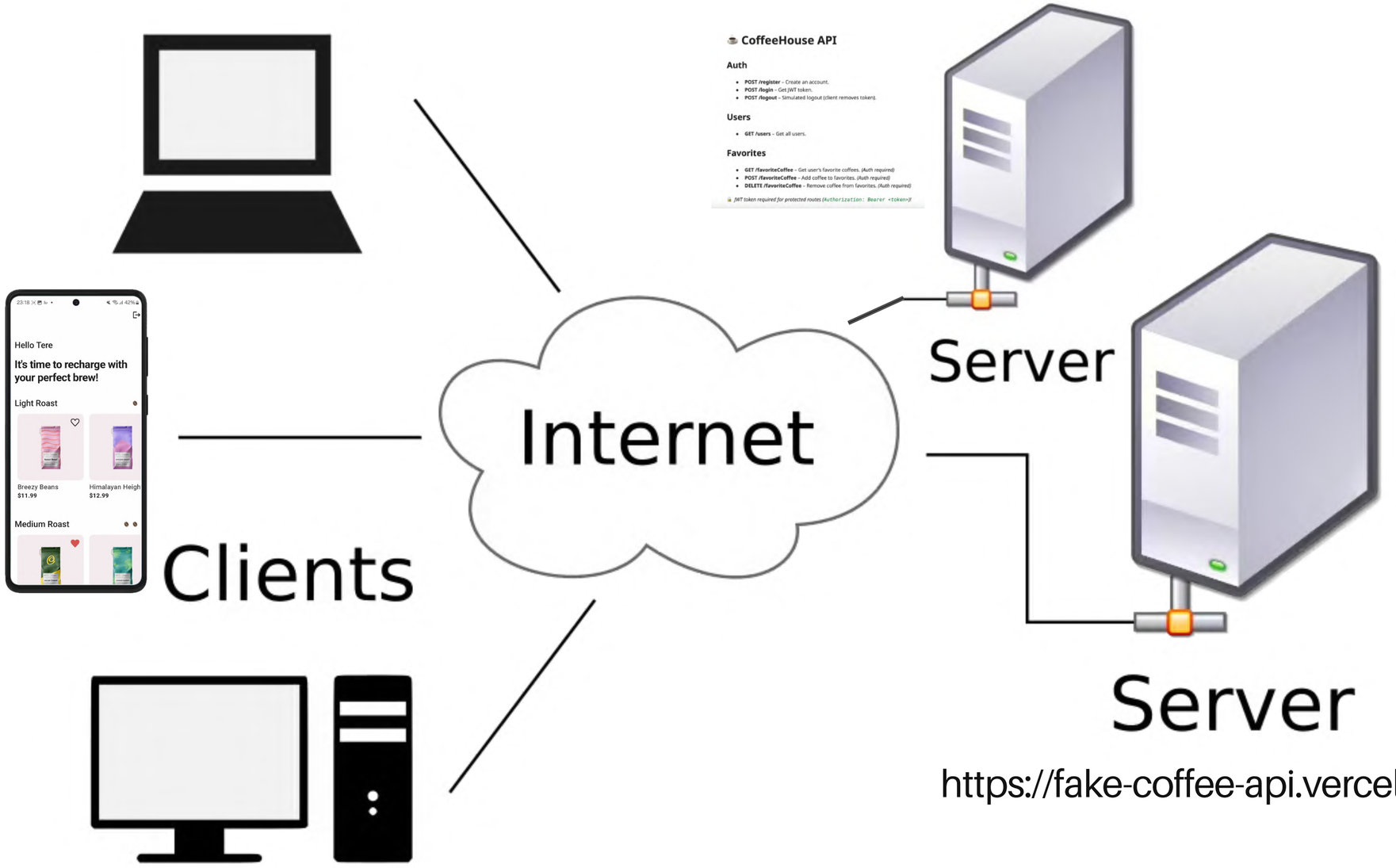
Time for a coffee break!

5

Token Handling, Interceptors, Image Loading


Hello Tere
It's time to recharge with your perfect brew!
Light Roast
Breezy Beans
$11.99
Himalayan Heigh
$12.99
Medium Roast
Internet
Clients
Server
https://fake-coffee-api.vercel.app

https://coffee-house-c8fac7659e47.herokuapp.com



https://fake-coffee-api.vercel.app

This is where I make an uncomfortably long pause.

Now, to Android Studio! 🚀

6

Homework, Q&A

Take it further – Enhance Coffee House App

01

Improve error handling

Improve the error handling by catching specific exceptions and displaying meaningful messages to the user

02

Add a Username Edit Option

Allow users to change their username within the app

03

Token exchange

Exchange a JWT token for a new one when it expires

“Postoji jedna jako lijepa poruka : nema glupih pitanja, nema viška pitanja i nema toga što ne možete kroz pitanja razjasniti. :P"

— Filip Babić, 2018.

OSC:ADA: We know everything!

ADA ×

Filip Babić (via OSC_ADA) <noreply@android.moodlecloud.com>
prima ja

21. tra 2018. 18:49

OSC:ADA » Forums » Obavijesti » We know everything!

We know everything!
by Filip Babić - Saturday, 21 April 2018, 6:19 PM

Pozdrav naši dragi budući Androidaši! :]

Kao što vidim nema nikakvih pitanja za sada. sad

Prije 2.5 godine kad sam krenuo raditi Javu/Android nisam ništa apsolutno znao. U roku od par mjeseci sam jako puno toga pohvatao!

E sad, da ne bi bilo kako se ja samo hvalim, koja je poanta ovog posta? Pa ne bi pohvatao puno stvari u par mjeseci, niti bi u 2.5 godine naučio koliko sam naučio da nisam ispitivao svaku glupost.

Postoji jedna jako lijepa poruka : nema glupih pitanja, nema viška pitanja i nema toga što ne možete kroz pitanja razjasniti. :P

Ideja je da vi stvarno ispitujete što više nas, do mjere da ne možemo disati i odgovoriti vam svima koliko ste nas uspamali.

Rekli ste da ste neke od zadaća riješavali 2-3-5-6 sati? U produkciji, može vam se dogoditi da imate problem ili bug koji morate riješiti. Vjerujte mi na riječ kad vam kažem da smo mi u firmi znali izgubiti 4-5 sati na probleme i gluposti, koje su se na kraju riješile u 5 minuta, samo zato što smo pitali ili razgovarali međusobno kako to srediti.

Q&A

Pitch

Want to make a presentation like this one?

Start with a fully customizable template, create a beautiful deck in minutes, then easily share it with anyone.

Create a presentation (It's free)