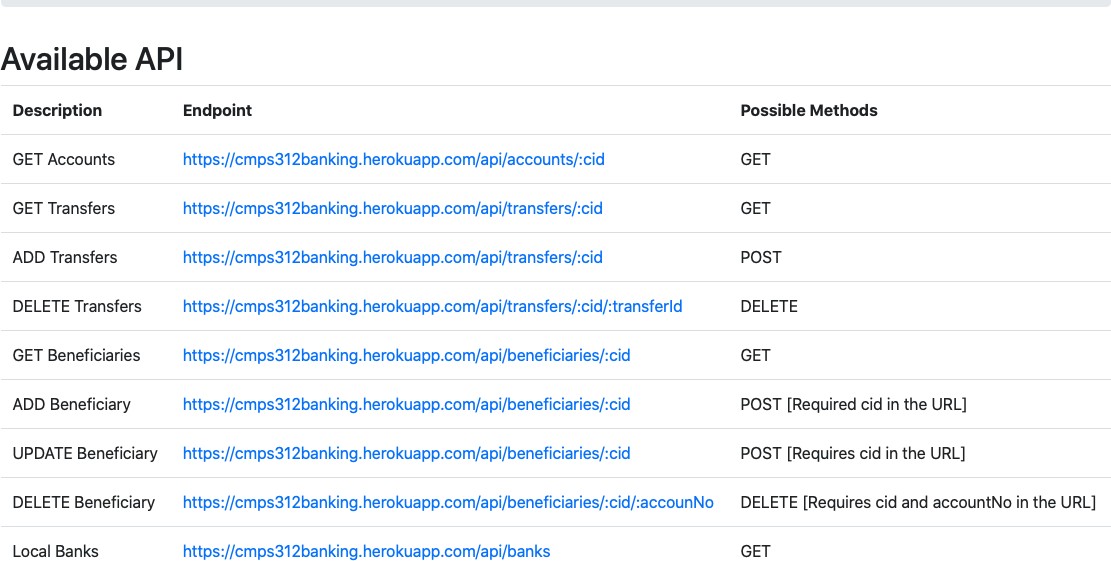
**CMPS 312 Mobile App Development Lab 9 – Web API with Coroutines and Ktor**

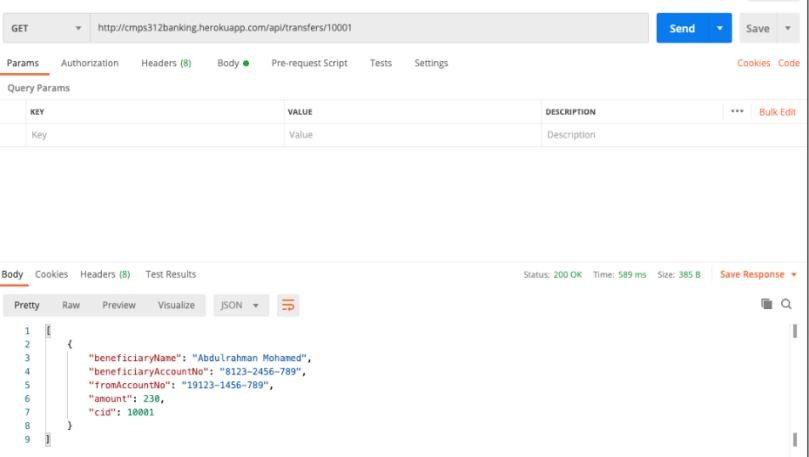
# Objective

In this Lab, you will **extend the Banking App** to communicate with the Bank Web API. You will be using the **Ktor** library, asynchronous suspend functions, and coroutines to get, add, and delete transfers and beneficiaries.

# Implement the Bank Web API using Coroutines

1. Open on Android Studio the ***Banking App*** project from **Lab9-Coroutines** folder in your repository. This project has the complete implementation of **Lab7-BankingApp** with some minor modifications such as delete button and new properties added to the Account class such as cid (i.e., Customer id).
2. Your task is to implement calling the Bank Web API using coroutines to read/write data from/to the remote bank service. You will be using Ktor with coroutines to achieve this.
3. Download postman from <https://www.postman.com/downloads/> and test the following Banking Service Web API available at [https://cmps312banking.herokuapp.com](https://cmps312banking.herokuapp.com/)





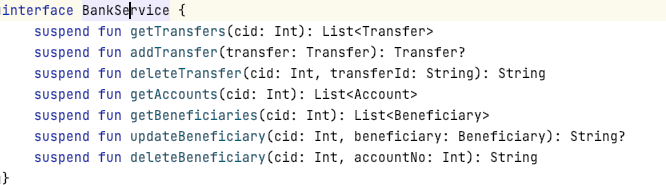
1. Add the following dependencies for ktor and coroutines in your build.gradle app module.

*//For Ktor Client*

*def* ktor\_version = "1.6.4"

implementation "io.ktor:ktor-client-android:$ktor\_version" implementation "io.ktor:ktor-client-serialization:$ktor\_version"

1. Inside the **webapi** package, create an **interface** called **BankService**

List all the interfaces methods that allow the App to communicate with Banking Web API available at [https://cmps312banking.herokuapp.com](https://cmps312banking.herokuapp.com/)

1. Create a QuBankService class under the webapi package that implements all the methods that allow the App to communicate with the Bank Web API.
   * Declare the BASE\_URL constant and set it to <https://cmps312banking.herokuapp.com/api>
   * Create a HTTP client and add JsonFeature auto-parse from/to json when sending and receiving data from the Web API.
   * Implement the **BankService** interface including getTransfers , addTransfer , deleteTransfer , getBeneficiaries , updateBeneficiary , deleteBeneficiary. Create BankServiceTest class and add a main method to test your implementation as you make progress.

For example: the following getTransfers() method sends a get request to url <https://cmps312banking.herokuapp.com/api/accounts/100101>and returns the list of transfers for customer 10001.

override suspend fun getTransfers(cid: Int): List<Transfer> { val url = "$BASE\_URL/transfers/$cid"

*println*(url)

return client.get(url)

}

**PART C: Change the App ViewModels to use QuBankService implemented in Part B**

Your task is to change the app view models to use the **new QuBankService** implemented in Part B.

1. Modify the **TransferViewModel** to use QuBankService and implement the following methods

fun getTransfers() {} fun getAccounts()

fun addTransfer(transfer: Transfer) {} fun getBeneficiaries() {}

fun deleteTransfer(transferId: String) {}

1. Modify the **BeneficiaryViewModel** methods to use QuBankService. You should not change anything else in the App, and it should work as before. **This is the fruit of using MVVM!!!** [ß)Q\*

