# Recruitment task (version 2.0) ENG

## 1. Task Description

Create a website that will provide the REST API endpoint via the HTTP protocol with the parameters indicated in point 3. The purpose of the website itself is to calculate commission for transfers made by a given bank customer. We assume that the data contained in the delivered file correspond to transfers made by a given customer in a given month. For simplicity, the end user should authenticate himself with BASIC on every HTTP request (stateless). The list of users can be predefined in a selected way, e.g. in a properties file with user as key and password as value.
At the entrance, the service accepts the customer's identification number. In response, the user should receive the number of performed transactions grouped by the user (first name, last name and ID), the sum of the transaction amounts, the calculated commission and the date of the last transaction of a given user.
Each request for commission calculation should log this fact into the MongoDb database, containing the client's ID, the amount of the commission and the date of commission calculation. The technical assumptions are described in the following sections.

## 2. Input data sets

**transactions.csv file:** a file containing transactions along with information about their amount, first name, last name and identifier ordering the transfer and the transaction date.

**fee_wages.csv file:** a file containing the commission percentage depending on the sum of the amounts of all transactions.

## 3. Search parameters

a) Identification number of the ordering client-client (id): searching for a user by the identification number (e.g. 5) with the possibility of entering several identification numbers separated by a comma (e.g. 1, 5). Entering multiple client IDs should be

considered as logical expression "or". Failure to provide the value of the client identifier parameter or filling it with the value "ALL" should accept all available identifiers

## 4. Sample request-response

Attention! The requests and responses shown below are not from the data attached to the task. They only show the logic of the search itself

Example:

Request parameters: customer_id = 3

Response: 1 user → 1. (First name: Adam, Last name: Adamowski, Customer ID = 3, Number of transactions: 11, Total value of transactions: 1500,00 Transactions fee value: 5,33, Last transaction date: 11.12.2020 14:54:31

Example 2:

Request parameters: customer_id = ALL

Response: 5 users →

1. (First name: Adam, Last name: Adamowski, Customer ID = 3, Number of transactions: 11, Total value of transactions: 1500,00 Transactions fee value: 5,33, Last transaction date: 11.12.2020 14:54:31).

2. (First name: Marek, Last name: Marecki, Customer ID = 1, Number of transactions: 3, Total value of transactions: 95200,30 Transactions fee value: 0,00, Last transaction date: 14.12.2020 11:00:01).

3. (First name: Anna, Last name: Annowska, Customer ID = 2, Number of transactions: 33, Total value of transactions: 12531,21 Transactions fee value: 0,99, Last transaction date: 31.12.2020 19:21:04).

4. (First name: Cezary, Last name: Czarecki, Customer ID = 4, Number of transactions: 0, Total value of transactions: 0,00 Transactions fee value: 10,00, Last transaction date: null).

5. (First name: Damian, Last name: Damianowski, Customer ID = 5 Number of transactions: 2, Total value of transactions: 1512,00 Transactions fee value: 5,01, Last transaction date: 26.12.2020 14:31:54)

## 5. Technical requirements

The website should be a web application written in Kotlin with the use of Spring Framework (preferably Spring Boot version 2.x). The code can also be partially written in Java (version 8 and higher), but keeping the proportion of classes written in Kotlin at least halfway.

Data delivered as files **should be loaded into the application memory at application startup**.  The database can either be located in a Docker container or as a standard "standalone" database. When designing a commission calculation solution, take into account the speed of operation (assume that the target application has a large number of clients (100,000 I transactions - about several dozen per one client). You can create the structure of the data saved to MongoDb in an arbitrary way.

Preferred tools for building are: Gradle 5 + or Maven.

Instructions for enabling and building the application should be included in the project.

You can also include an application docker image.

The endpoint to which the task relates should have unit tests written in Java or Kotlin or Groovy. The unit test enabling tool is up to you, however, please include instructions on how to run these tests.

The use of good practices such as "clean code" and design patterns where it makes sense will be an additional advantage.

P.S. We have seen from experience that many candidates make several common mistakes. Here are two friendly reminders to avoid them 😉 :

- Load data to the application memory, not to the database
- Note that required endpoint tests are unit ones

The task should not take you longer than 4 hours. 🙂

Good luck!