JAVA Backend Candidate – Home Test

Introduction

The purpose of this test is to evaluate your technical abilities in various aspects of Software Engineering. Also, allowing common grounds for discussion in the interview that takes place after you complete this test.

The test is a home test and you are encouraged to use any resource at your disposal – online etc. You will be asked to explain your solution in the interview, emphasizing on correctness, Design and coding decisions.

You are asked to submit via email, working and tested code.

The code should be in Java 8 and Use Spring Frameworks such as Spring Boot.

There are some files attached to the mail to help you with the Home Test.

For any questions/remarks, please feel free to approach tomer.negbi@motorolasolutions.com or moshe.benabu@motorolasolutions.com

Overview

Congrats! You are about to implement a highly sophisticated program, whose sole purpose is to monitor calling records data from a phone device.

The module should contain several services to receive the data from the devices, validate and save them in a db, and expose abilities to query them and update them.

Business Requirements

Receiving Data

The Program needs to expose a Rest API (a REST Controller) that will get an input of a phone call data.

The phone call data will contain the following fields:

Time, Incoming / Outgoing, Duration, Phone Number

Here is a detailed explanation of the fields:

Time = Date of the phone call record

Incoming / Outgoing = String that indicates if the call is incoming or outgoing

Duration = How much time the phone call took in seconds

Phone Number = what is the other phone number we called or received a call from.

Attached is a list of JSON examples of phone records data (you can use them or create and test with some of your own, just make sure to keep same JSON structure).

The REST API will receive the phone call data and should not return anything back as response.

Validation

We attached a CSV file containing all the list of phone numbers that are saved as balcklist for this device.

For every record we receive from the REST API the program will perform a validation that the record is not a part of a blacklist.

If the phone number exists in the blacklist the program will through an exception with a proper message. (Because it means that from some reason we did got a call from a blocked number)

If it's not part of the blacklist – continue with the program.

Business Logic

Every record that we received from the REST API we have to save.

The program needs to save this record in a DB at you're choosing (MongoDB, Postgrest, or anything else).

You are encouraged to use Spring Data Framework to save it (but not mandatory, you can use JPA/Hibernate or any other Interface that you're comfortable with).

We attached a CSV file containing all the list of phone numbers that are saved as contact for this device.

Before saving the record into DB you need to check if the phone number you received is in the list of contacts given to you, if they are you should save another field, type *boolean*, that is called: *savedContact* with value *true*, if it's not save it with value *false*.

You can use the CSV file in any way you see fit (read it every time, put it in a map on memory, put this data in the DB, something else…)

Get Data

The Program needs to expose a Rest API (a REST Controller) that will allow us to get data from the DB.

We will have 2 API's:

1. Get all records of a specific phone number:

   Input – String of phone number

   Output – list of all the records of that phone number.

2. Get all records that are bigger than some duration

   Input is an integer of duration (example: 600 seconds), output is a list of all the records that took more than that duration (example: all records that duration is bigger than 600).

Update Data

The Program needs to expose a Rest API (a REST Controller) that will allow us to update a data on the DB.

In case somebody has switch he's phone number we want to be able to update that phone number to the new one.

In order to get this ability we need to expose an API that will update all Records of a specific phone number.

Input – String old phone number, new phone number

Output – none

When calling the API the program will update and replace the phone number fields in all the records that has the "old phone number" to the given "new phone number".