**Technical Test for CMF**

The goal of this technical test is to measure the developer's capabilities of producing back-end and front-end solutions around a simple use case.

**:warning: Pre-Requisites:**

* having installed [JDK8](http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html)
* having installed [Maven](https://maven.apache.org/) (version 3 or more)
* having access to internet

## Test content (1:30 hour)

:warning: **15 minutes** are required for **carefully** reading the current document and considering all provided files.

The technical test is split as bellow:

**Back-end**

The back-end test is split into 2 steps which must be considered in the bellow order:

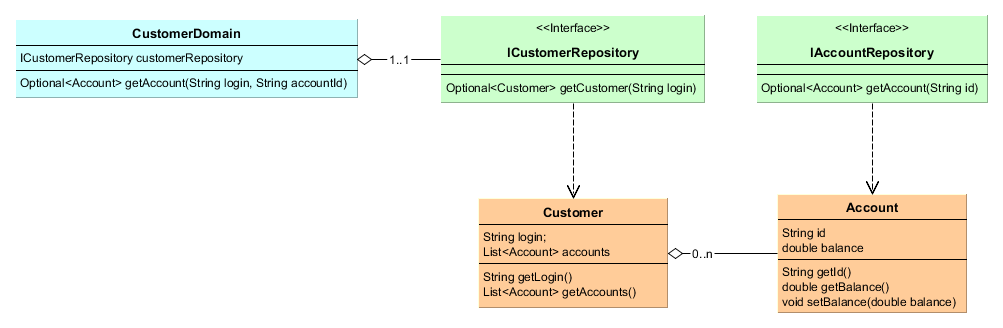
1. developing business logic from a simple feature description with high level of quality (*Java*) **(15 min)**
2. exposing the business logic through a RESTful endpoint (*Spring-Boot*) **(30 min)**

**Front-end**

1. implementing webapp which calls RESTful endpoint (*AngularJS*) **(30 min)**

:warning: back-end and front-end can be done **independently** even if the endpoint is the same.

**Back-End (45 minutes)**



**1. Developing a business logic (15 minutes)**

For this part, we will implement a simple version of hexagonal architecture. CustomerDomain is expected to handle business operations and **must** remain independent from any framework (including spring).

As a customer, I want to be able to get my account's information

**Scenario**: a customer should be able to read his account

**Given** I am a customer with "100.0" on my account "TEST\_002"

**When** I check my account "TEST\_002"

**Then** my balance should be "100.0"

**GOAL**

Implement this feature in CustomerService class with **high level of quality** regarding to this requirement.

**:warning:** Spring-Boot **is not used** at this step.

1. **Exposing this business logic through a RESTful endpoint (30 minutes)**

*SpringBoot Application launcher is Application.class. Sample beans of ICustomerRepository and IAccountRepository are provided. Default port is 8080 (cf. application.yml).*

Given the CustomerDomain which has just been updated, we would like to create a RESTful service in order to access it.

**GOAL**

Implement the endpoint to use CustomerDomain (as remind CustomerDomain **must not be** a spring service)

The endpoint will be mapped to **GET /api/v1/customers/{login}/accounts/{accountId}/balance** and will directly return the balance amount (or 0 if not found).

**:warning:**Spring-Boot **is mandatory** at this step.

**Example**: regarding to provided repositories  
http://localhost:8080/api/v1/customers/steve.jobs/accounts/STEVE\_001/balance 🡪 100.0

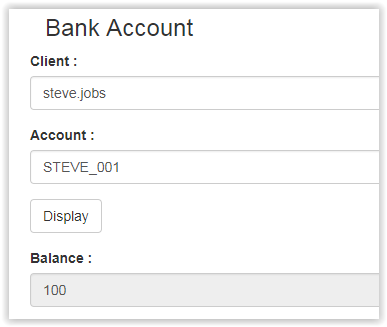
**Front-End (30 minutes)**

**NB**: there is no front-end server for this test. When the index.html page is launched through a browse, the page to be tested is displayed.

As an admin, I want to be able to display account's balance for a given client's login and its account's id.

* When I click on “Display” button, the back-end endpoint is called and the account’s balance should be displayed in the correct field
* When no customer nor account are found, the balance field is set to 0.

**Example**:



**GOAL**

Implement index.html, controller.js and service.js to fit to this requirement.  
As remind, the endpoint should be:   
**GET http://localhost:8080/api/v1/customers/{login}/accounts/{accountId}/balance**