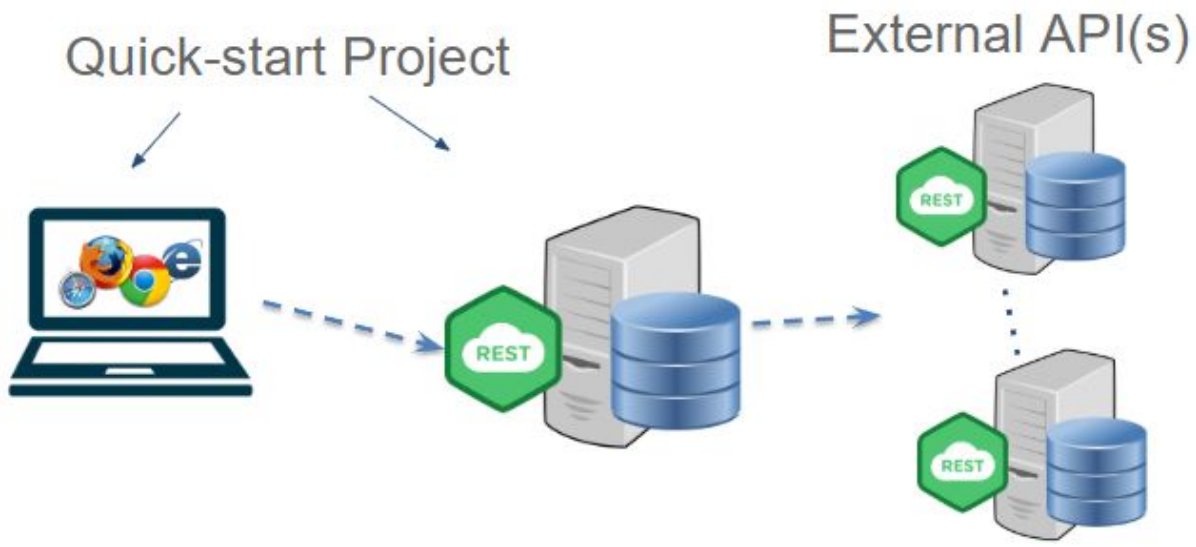
CA-3 Spring 2020



The purpose of this CA is to create a *Quick-start Project*, which will allow developers to get a quick start with applications having an architecture as sketched above. That is applications where the server needs to communicate with other servers (Momondo or similar sites) and includes a web-client (SPA) that communicates with the backend via a REST-API

The CA will also ensure that you get a setup ready for the semester project and VERY IMPORTANT for the exam, which will ensure you can do testing (on the backend) and quickly (as in minutes) deploy your backend code with REST and database access.
You should use the **security branch** of the usual setup project, changed to handle passwords the right way, as the start code for this exercise (The security exercise from last week)

This CA will strengthen your skills and knowledge related to:

- Continued use of JPA, JAX-RS and Java
- Continued use of maven, Testing, Continuous Integration and Continuous Delivery
- Handling Authentication/Authorization and Passwords for SPA's
- Creating REST endpoints that must delegate work to other servers in order to make a response
- React and React Router
- Build and deploy full-stack SPA-applications

IMPORTANT: *Info Before you start*

The requirements for the Quick Start Project is as follows

It must include two separate projects, backend + a web-client, and all must be “runnable” immediately when cloned, after the database(s) is set up, and URLs are adjusted.

Backend:

- Must provide an initial setup with entity classes for *users* and *roles*
- The backend must provide a JWT-based authentication/authorization mechanism.
- The backend must provide unit and integration tests, and must be “testable” by Travis
- The backend must be deployable using a single maven command
- Passwords must be protected using a hash/salt-strategy
- The backend must provide (at least) the following Dummy REST-endpoints:
    - An endpoint that requires the user-role
    - An endpoint that requires the admin-role
    - An endpoint that fetches data from a minimum of five remote servers (use https://swapi.co/ or similar free API's for the start code) (see hints and requirements at the end of this document)

WEB-Client:

- Must be implemented as a Single Page Application with React and provide:
    - A login/logout option
    - It must render the username, and if used, role(s), for a logged-in user (in any way you like)
    - A React Router Based setup with initial pages/routes that renders data fetched from the three endpoints mentioned above and a welcome page with initial instructions on how to use the Quick Start Project.
    - Must be styled (use bootstrap unless you have knowledge from somewhere else) to be immediately “presentable”
    - URLs used by the client-projects must be read from a file `settings.js`

Documentation and Proof of Functionality

- The Backend project on GIT must include a README.MD file with instructions on how to use (all parts of) the Quick-Start Project
- The Client Project must also include a README page with documentation similar to what you wrote for the backend.

Hand-in and Deployment

You need to add the following to the hand-in document:

- A link to the Github Repo(s) for the GROUP Version
- **Each member** of the group must (personally) demonstrate a use scenario of the start code, that is: set it up on Git, Travis and on a personal Droplet accessible only via https. Each group member decide how to deploy the Client App (Surge, Nginx, Tomcat or whatever)
- The “Default page” on each of these deployed versions must include a PERSONAL description written by the specific group member with reflections related to how the start code was used

How and When to hand-in

Before the end of Tuesday April 14th. Check that your group names and group members are as sketched in this document.

Before Sunday April 19th 16.00, make sure to update the document with personal links to everyone's solutions.

We assume that you all have a complete CI-setup ready BEFORE the end of day-1. Generally, we assume this to be the simplest CA, given this semester, so if you finish before the given deadline, use it to get an early start on your semester project.

Meeting schedule for CA3 Reviews April 20 will come her

The review of CA3 will be online over zoom and each student must prepare a presentation:

1. Group presentation: An introduction/demonstration from **the group**, explaining how far you came, problems you might have had, and importantly, who did what.
2. After that, we expect **each member** to introduce their own personal copy of the implemented start code. This must include a demonstration of the Travis log-file, the actual running application on your OWN droplet + a short talk about what YOU DID using your own GitHub repo as the starting point.

Hints: Calling an external API from your backend (GET)

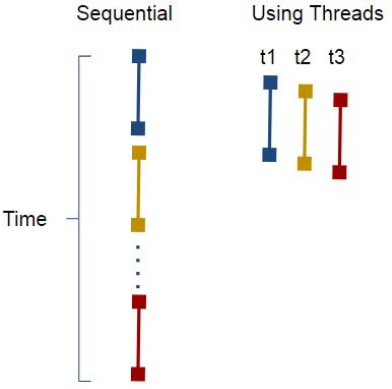
The requirements states you have to make a request up against five external servers. Just use Swapi or a similar open API for all five requests, but your task will be to “assemble” the responses into one final response which you return from to your clients.

You can use this small utility function to make a request from your server up against the Swapi-api:

```
public String fetchData(string _url) throws MalformedURLException, IOException{
    URL url = new URL(_url);
    HttpURLConnection con = (HttpURLConnection) url.openConnection();
    con.setRequestMethod("GET");
    //con.setRequestProperty("Accept", "application/json;charset=UTF-8");
    con.setRequestProperty("Accept", "application/json");
    con.setRequestProperty("User-Agent", "server"); //remember if you are using SWAPI
    Scanner scan = new Scanner(con.getInputStream());
    String jsonStr = null;
    if (scan.hasNext()) {
      jsonStr = scan.nextLine();
    }
    scan.close();
    return jsonStr;
  }
```

External requests and performance

The method given above blocks several places, waiting for “the other end” of the communication. The effect of this is, that if you just call it five times from one of your end-points, it could take almost **five times longe**r than necessary (could be a serious problem if the servers are “slow”) as sketched in the first part of this figure.



This solution is acceptable if your group is VERY green

All other groups should go for a “parallel solution”. Use Alexander's suggestion (found on Slack) as inspiration for how to do it in parallel, as sketched in the second part of this figure.