Code.Hub

# Technico
# The technicians visit Jakarta and Angular islands

Requirements specification document

**September 2024**

Code.Hub

## General description of the assignment

The project is an individual assignment for a full-stack application, the Technico web application. The implementation will take place in two parts, namely the back and front end.

In the appendix, the functional and non-functional requirements for the project are given. They are typically the same as in the previous assignment. Therein a console application was required. Now, an integrated application is required. In the first part, the previous code has to be converted to a REST API using Jakarta EE and the WidlFly Application Server. In the second part, the front end will be implemented as a web client with complete coverage of all requested endpoints.

## Milestones

The milestones are given as a logical separation of the various tasks and will help us to allocate the various stages of implementation in time. It is not obligatory to observe them as ordered, but it is suggested that they be followed so that there is an indication of the progress of the deliverables.

**Back end**

**Final push 27th September 2024 EOBD**

- Refactor the project that you have already produced and check if it conforms to the requirements. Create your personal private repository and assign the instructors as collaborators
- Prepare and configure the application server
- Introduce dependency injections when needed
- Implement the endpoints with the appropriate Jakarta EE classes
- Implement the login functionality and security features
- Test the artifact with unit testing and postman collections
- Implement the nice-to-have functionalities if there is time

**Front end**

**Final push 4th October 2024 EOBD**

- Implement the home pages of Admin and Property Owner / User
- Design the User and Admin pages and navigation
- Implement import, search, modification, deletion • Test the front end
- Implement the nice-to-have pages if there is time

Code.Hub

# Appendix

## 1. Functional and non-functional requirements Project general description

A Renovation Contractor Agency, **Technico**, within the framework of its operation, needs a web application that will enable the employees - managers of this platform to access information concerning customers and repairs. It will also enable its customers to oversee the progress of repair/renovation work on their property.

## User roles

- Admin (**Employee of the Agency**)
- Property Owner, User of the platform, customer

## Functional requirements

### Description of the application

### Login Page

Everyone will enter her/his email and password and depending on her/his type/role, will be redirected to one of the following 2 pages

1. Property Owner (User) Home Page
2. Administrator Home Page

On this page a self-registration page for the user will be linked.

### The self-registration page for Property-Owner

This page requires the following fields:

- VAT number (ΑΦΜ), which is a unique identifier that characterizes users)
- Name
- Surname
- Address
- Phone number
- E-mail (used as Username )
- password
- type of user

Code.Hub

## The user Homepage

This page shows only the repairs that concern the current user. This page is expected to align with the GDPR requirements.

Also, it will allow her/him to navigate to the following pages:

- Property Owner details page
- Property details page
- Repairs page

## Property Owner details page (user view)

The pages for property owners include the following options: Update, Delete.

**Update (nice to have)**: On this page, the complete details of a user will be displayed, and the administrator will be able to modify them. The navigation on the page can be done with a corresponding link on the search page.

**Delete (self-deactivation)**: Through the update page, a special button will be provided that will allow this function. It is recommended to press the JavaScript window to confirm the action at the touch of a button. Consider the soft delete option.

## Property Page (user view)

It will contain the following functions: View, Create, Delete, and Update.

**View**: Displays all the details of the property.

**Create**: Form with the following fields:

- A Property Identification Number, which coincides with the corresponding number of E9 and will uniquely characterize the property,
- Property address,
- Year of construction,
- Type of property (Detached house, Maisonet, Apartment building), (Note: this field is now removed by the owner
- VAT number of its owner.

**Update (nice to have)**: All the details of the property can be edited.

**Delete**: The property can be permanently deleted or deactivated.

## Property Repair Page (user view)

It will contain the following functions: Search, Create, Delete, and Update.

**Create:**

- Date (datetime) of the scheduled repair
- Type of repair (Painting, Insulation, Frames, plumbing, electrical work)
- Repair description as a free-text field for the work to be done (e.g., installation of a solar water heater)

- Repair address
- Status of the repair (Pending, In progress, Complete - default is the pending status)
- Cost of repair
- Owner for whom the repair is made

**Update**: As in the Owner page, where it will be possible to update the repair details (e.g., status)

**Delete**: Through the update page, a special button will be provided that will allow this function. It is recommended to press the JavaScript window to confirm the action at the touch of a button.

Nice to have

- Validations via JavaScript for data entry forms. For example, a user with a blank VAT number cannot be entered.
- Validations in the Backend when entering data: For example, you cannot enter a user with an existing email or VAT number
- Owner has more than one property.
- Authentication & Authorization: After their identification, depending on their type, ordinary users should not have access to the administrators' pages (i.e., changing the URL).

## The administrator Homepage

The page will present the active repairs of the day (ongoing repairs). Also, it will allow administrators to navigate to the following pages:

- Property Owners and Properties page
- Repairs page

## Property Owners and properties page (admin view)

The page will display a list of the properties using a paging view. Also, it will contain links for the users and property management to the following pages:

1. Create a property owner with the corresponding form (create-owner-page)
2. Owner data processing page. Stylistically it will look like creating an owner just the fields will be pre-filled. In addition, there will be a button that when pressed will delete the corresponding Owner. (edit-owner- page)
3. User search page: A form with the search criteria (VAT number, e-mail) and below a table with the results (initially empty on the 1st visit to the page). (search-ownerpage)
4. Create-property- page (nice to have)
5. Edit-property- page (nice to have)
6. Search-property- page (nice to have). The administrator will be able to search for properties based on various criteria, but mainly

- Property ID Number
- VAT number

**Note**: Update and Delete options can be found next to the search results for each owner/ user.

## Repairs page (admin view)

The page will display a list of the pending repairs. Also, it will contain links to the pages for the repair management as follows

1. Create-repair- page
2. Edit-repair- page (nice to have)
3. Search repair page: Based on
    - Date or Range of dates
    - User ID in case we want to display all the repairs made for a property owner.

## Non-functional Requirements

1. The programming environment for the back end
    1. Java SE 17
    2. Database: MySQL Server
    3. Jakarta EE
    4. Application Server WildFly 27.0.1
2. Programming techniques
    1. Suitable Object Relation Mapping library
    2. Dependency Injection is required
3. The programming environment for the front end
    1. HTML, CSS (and/or Bootstrap, templates, etc.)
    2. JavaScript (ES5/ES6)
4. Coding and architectural standards and conventions
5. VCS: Project's source code must be delivered on GitHub. Each commit should be depicted with clear and descriptive messages. The use of branches for each major feature development is mostly appreciated.

Code.Hub

## 2. Rubrics

## Rubric for assignment evaluation of the back end

| # | Topic | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 1. | JPA model, repository | | | | | |
| 2. | Dependencies in POM | | | | | |
| 3. | Logging | | | | | |
| 4. | Services design, interfaces, exceptions | | | | | |
| 5. | Proper validation | | | | | |
| 6. | Dependencies injection when needed | | | | | |
| 7. | REST API design | | | | | |
| 8. | Implementation of JAX-RS | | | | | |
| 9. | Correctness of functionality | | | | | |
| 10. | Conventions, Formatting, Java Doc | | | | | |
| 11. | Testing | | | | | |

**Levels**

1 Beginning: very few expectations/standards have been met; work demonstrates misconceptions.

2 Developing, Average: some expectations/standards have been met; major improvements are required.

3 Competent, Satisfactory: most expectations/standards have been met; minor improvements are required.

4 Advanced, Very Good: meets expectations/standards effectively and efficiently.

5 5 Proficient, Professional: fully meets and exceeds expectations/standards; adds innovative/insightful elements.

Code.Hub

## Rubric for assignment evaluation of the front end

| # | Topic | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 1. | Design and easiness of the pages | | | | | |
| 2. | Correctness of functionality | | | | | |
| 3. | Proper validation | | | | | |
| 4. | Conventions, Formatting, Separate script files | | | | | |
| 5. | AJAX calls implementation | | | | | |
| 6. | Effective DOM manipulation | | | | | |
| 7. | Security implementation | | | | | |
| 8. | Testing | | | | | |

**Levels**

1 Beginning: very few expectations/standards have been met; work demonstrates misconceptions.

2 Developing, Average: some expectations/standards have been met; major improvements are required.

3 Competent, Satisfactory: most expectations/standards have been met; minor improvements are required.

4 Advanced, Very Good: meets expectations/standards effectively and efficiently.

5 Proficient, Professional: fully meets and exceeds expectations/standards; adds innovative/insightful elements.