Software Design Document for the Web Savings Calculator

September 2021, Skopje

Revisions

Version	Author (s)	Description	Date
Version 1.0	Marija Nikolova	Initial version of the technical documentation for the web savings calculator	25.09.2021
Version 2.0	Marija Nikolova	Added details for the frontend part of the calculator	10.10.2021

Approval history

Approving party	Version approved	Signature	Date

Review history

Reviewer	Version reviewed	Signature	Date

Table of figures *5*
I Introduction *7*
Document scope and purpose **7**
Target audience **7**
Acronyms / Abbreviations **7**
Reference documents **7**
System environment **7**
Design approach **8**
Data flow design **8**
Architecture design **8**
Backend component 8
Frontend component 9
APIs overview **9**
Version control **10**
II Overview of modules *11*
Config data **11**
Customer **11**
CashFlows **11**
Goals **11**
Accounts **11**
Projection **12**
III Entity diagram *12*
IV Class diagram *12*
Domain model **12**
Controllers **13**
Services **14**
V Sequence diagrams *15*
Config data **15**
Customer **16**
Goal **17**
Cash flow **18**
Account **20**
Projection **21**
VI Algorithms / formulas for annual values calculation *22*
Cash Flow **22**
Account **23**

Regular account ... 23
Deposit account ... 23
Second pillar account ... 23
Investment account ... 23
VII Graphical interface ... 23
Homepage ... 23
Sign-up ... 24
Dashboard ... 25
Personal data ... 26
Accounts ... 28
Goals ... 29
Cash flows ... 31
Projection ... 33
Page not found ... 34
User not logged in ... 34
VIII Project setup ... 35
Backend component ... 35
Cloning the project ... 35
Building the project ... 35
Starting the web application ... 36
Checking the services in the browser ... 37
Frontend component ... 38
Cloning the project ... 38
Installing dependencies ... 39
Local webpack configuration ... 39
Prettier setup ... 39
Checking the local H2 database ... 40
IX API generation and making changes ... 41
X Production deployment ... 42
Team members ... 42

Table of figures

Figure 1 FE - BE components 8
Figure 2 Overview of the architecture of the system 9
Figure 3 Swagger - UI for the web savings APIs - 1 10
Figure 4 Swagger - UI for the web savings APIs - 2 10
Figure 5 Entity diagram of the Web Savings Calculator API 12
Figure 6 Class diagram for the domain models 13
Figure 7 Class diagram for the controllers 14
Figure 8 Class diagram for the services 15
Figure 9 Sequence diagram for getConfigData 15
Figure 10 Sequence diagram for saveConfigData 16
Figure 11 Sequence diagram for retrieving customer data 16
Figure 12 Sequence diagram for the saveCustomerData method 17
Figure 13 Sequence diagram for saving goals 17
Figure 14 Sequence diagram for retrieving the goals 18
Figure 15 Flow of deleting goal 18
Figure 16 Sequence diagram for saving cash flow 19
Figure 17 Sequence diagram for retrieving the cash flows 19
Figure 18 Flow for deleting cash flow 20
Figure 19 Sequence diagram for saving accounts 20
Figure 20 Sequence diagram for retrieving accounts 21
Figure 21 Sequence diagram for deleting account operation 21
Figure 22 Sequence diagram for retrieval of annual values part 1 22
Figure 23 Sequence diagram for retrieval of annual values part 2 22
Figure 24 Homepage 24
Figure 25 Scenario with invalid credentials 24
Figure 26 Sign up form 24
Figure 27 Alert after successful register 25
Figure 28 Login page after successful registering 25
Figure 29 Dashboard 26
Figure 30 Personal data 27
Figure 31 Saving personal data 28
Figure 32 Accounts page 28
Figure 33 Form for adding new account 29
Figure 34 Editing already existing account 29
Figure 35 Goals page 30
Figure 36 Add new goal 30
Figure 37 Update goal 31
Figure 38 Cash flow page 32
Figure 39 Adding new cash flow 32
Figure 40 Update of cash flows 33
Figure 41 Projection page - part 1 33
Figure 42 Projection page - part 2 34
Figure 43 Page not found 34
Figure 44 Login to enter page 34

Figure 45 Importing the project from VCS control in Intellij 35
Figure 46 Clean install on the project 36
Figure 47 Web application start configuration 37
Figure 48 Swagger UI for the Web Savings Calculator 37
Figure 49 Example for executing the saving config data query 38
Figure 50 Cloning the FE component 38
Figure 51 Installing fe dependencies 39
Figure 52 Local run configuration 39
Figure 53 Prettier configuration 40
Figure 54 Login into the h2 local db 41
Figure 55 H2 db UI 41
Figure 56 Maven plugin for code generation 42

I Introduction

The introduction part provides overview of the Software design document for the Web Savings Calculator Calculators. It includes the purpose, scope, target audience and design approach of the system. This document describes the technical documentation for the implementation of the Web Savings calculator.

Document scope and purpose

This document provides description of the technical design for the Web Savings Calculator Web application. The main purpose of the document is to describe the technical vision of how the business requirements will be realized on the backend side. The document also provides an architectural overview of the system to depict different aspects of the system. This document is meant to work as foundational reference point for developers.

Please note that this document is constantly being updated during development.

Target audience

This document is mainly targeting technical stakeholders:

- Development team
- Consumers of the APIs
- IT management

It is not limited to the above-mentioned group, although it is assumed that the reader has technical background in software design and development.

Acronyms / Abbreviations

Term	Definition
API	Application programming interface
RESTful service	Representation state transfer services
Open API specification	Provides a standard, language-agnostic interface to RESTful APIs readable for humans and computers

Reference documents

- System requirement document for the Web Savings Calculator

System environment

- Development: Intellij Idea 2021.1.1
- Unit test: Junit
- Dependency management: Maven
- Frameworks: Spring
- API design: Open API 3.0

- Diagrams: VisualParadigm / Drawio.com
- Database management: H2 in memory db for development, My SQL workbench
- Server: Embedded Tomcat
- Discussions: Microsoft teams
- Version control: Github
- Lombok
- React: 17.0.2
- React-final-form
- React-router-dom
- Highcharts

Design approach

The design approach is based on the following.

Data flow design

The Web application is created as two micro services – the backend serving the data through the endpoint and the frontend which is React application and it is consuming the data as well as containing the Graphical interface of the application.
The services are created as RESTFul services where the data is consumed / provided by the consumers and saved to database through My Sql / H2 (local development). Saving is done through Hibernate and JPA. The APIs are generated based on the Open API 3.0 specification yaml file.
The frontend application is created as React application with help of create-react-app scripts and it is consuming the backend services. Figure 1 shows how these applications are communicating.

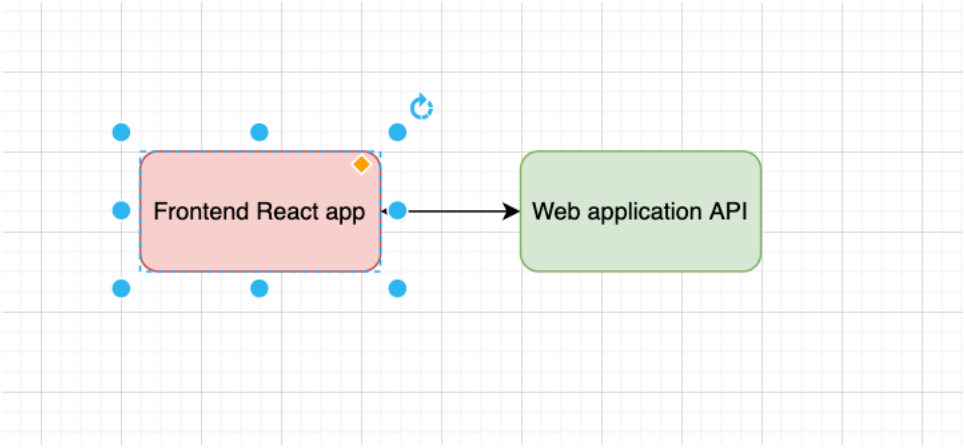

Figure 1 FE - BE components

Architecture design

Backend component

The Web Savings Calculator API service is following Four Layer Architecture so that the system can be easily organized with separation of concerns and preparation for distribution and reuse. The advantage of this approach is that due to separation of concerns, the components can be easily changed without affecting bigger changes, as well as code readability.

- The presentation layer: Contains the API defined in Open API 3.0 specification file and controllers from typical MVC. The consumer can check all the available APIs through configured swagger UI.
- The domain model: Models representing the logic with OO design for reuse. The objects in this layer are application independent and can be reused.
- The persistence layer: Hibernate and JPA entities which are mapping the domain model to and from the database.
- The data layer: The data is managed in H2 for development purposes, in production, will be switched to My SQL.

Figure 2 shows an overview of the architecture of the system.

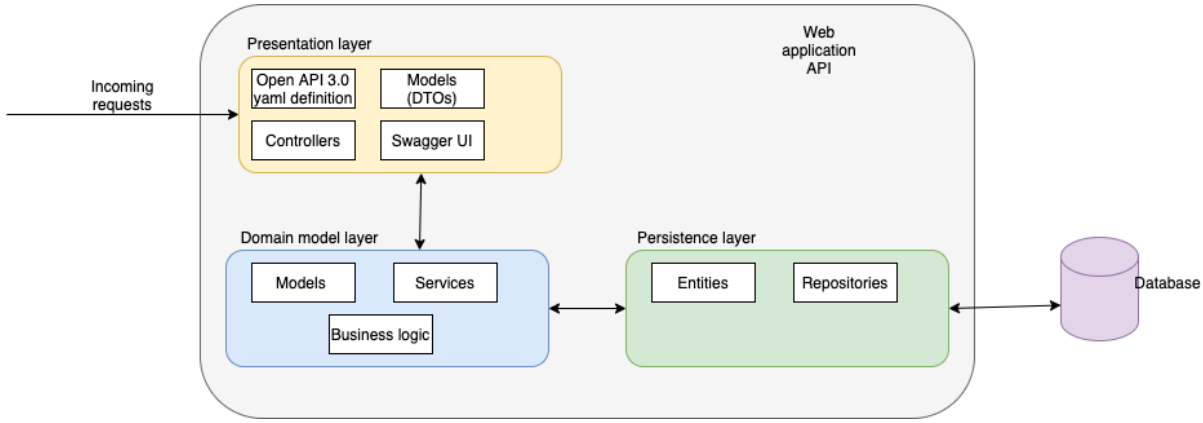

Figure 2 Overview of the architecture of the system

Frontend component

The frontend component is created as React application consuming the APIs defined in the previous section. All the UIs are listed in the sections below. Here is a detailed list of all libraries used for creating the ui:

- React
- React-router-dom – for routing
- React-bootstrap – for styling of the components
- Styled-components – for styling of the components
- React-final-forms – easier manipulation with forms
- Prettier – automatic code reformatting
- Highcharts – for the charts in projection

APIs overview

The complete definition of the APIs is done through Open API 3.0 specification, so it is easy to show a graphical interface representing the UIs through Swagger UI. The swagger UI is automatically generated based on the yaml definition, using the spring doc library. Figures 3 and 4 show how the swagger UI looks. This UI is helper UI made to easily understand what is behind the service and how to use the APIs.

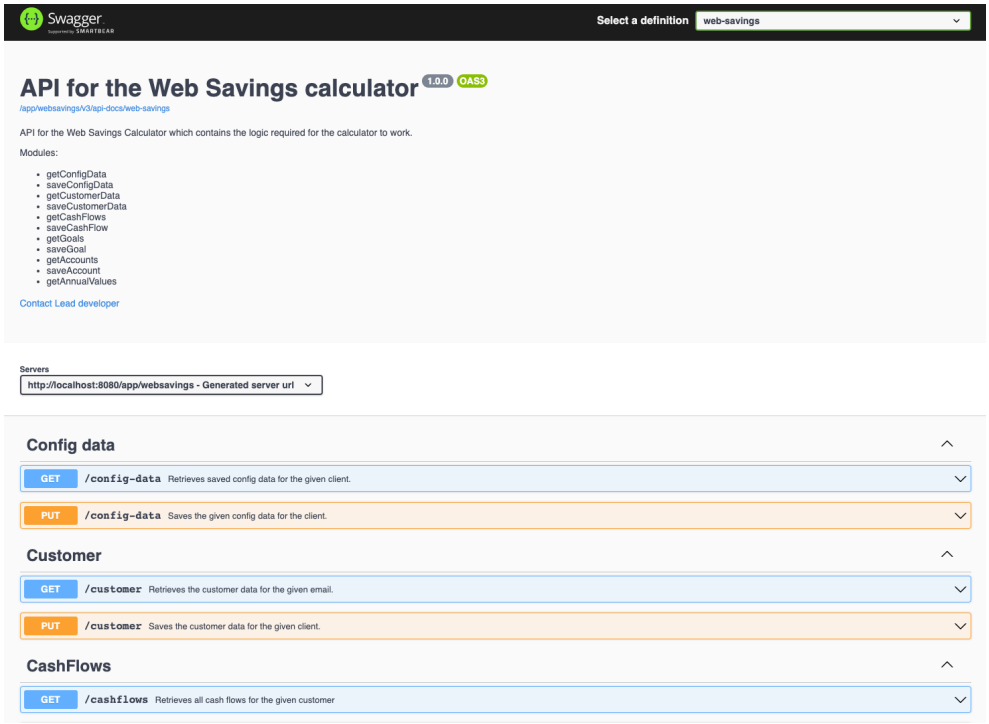

Figure 3 Swagger - UI for the web savings APIs - 1

Figure 4 Swagger - UI for the web savings APIs - 2

Version control

Github is used for version control and the complete source code is available on the following links:

- Backend component: https://github.com/MarijaNikolova/WebSavings.git.
- Frontend component: https://github.com/MarijaNikolova/WebSavingsFE.git

II Overview of modules

This section contains overview of the modules / functionalities implemented with the Web Savings API web application.

Config data

Module used to save and retrieve the configuration data for the customer plan. The configuration data is saved per customer, and it contains the language in which the customer chose to check the application (MK or EN) as well as the number of years for which the projection of the assets needs to be done.

Two endpoints are provided for the configuration data handling: getConfigData and saveConfigData.

Customer

Module used to save and retrieve the data for the given customer. One endpoint used for login, the other one for sign up. For sign up, the customer needs to enter email, date of birth, password, name and surname. For login, the customer sends the email and the password.

Two endpoints are provided for this customer functionality: getCustomerData and saveCustomerData.

CashFlows

Module used to save and retrieve the data for the cash flows. As per the definition in the requirements, the cash flows are recurring or one – time incomes or expenses that the client receives. The get method retrieves the list of all cash flows, whether the save method saves a cash flow for the given customer.

Three endpoints are provided for this functionality: getCashFlows, saveCashFlow, deleteCashFlow.

Goals

Module used to save and retrieve the data for the goals. As per in the definition in the requirements specification, the goals are planned expenses for the client. Endpoint for retrieving all goals is provided as well as endpoint for saving a goal.

The three endpoints provided for this functionality: getGoals, saveGoal, deleteGoal.

Accounts

Module used to save and retrieve the data for the saving accounts of the customer. There are three endpoints for this module as well – getAccounts, deleteAccount and saveAccount.

Projection

The module for projection has only one endpoint called getAnnualValues and it is used to return the complete projection for the accounts, cash flows and goals which should be shown on the timeline in the UI application. Different calculations based on the type of the assets is done and value is retrieved for each year.

III Entity diagram

Figure 5 shows the Entity diagram for the Web Savings calculator API.

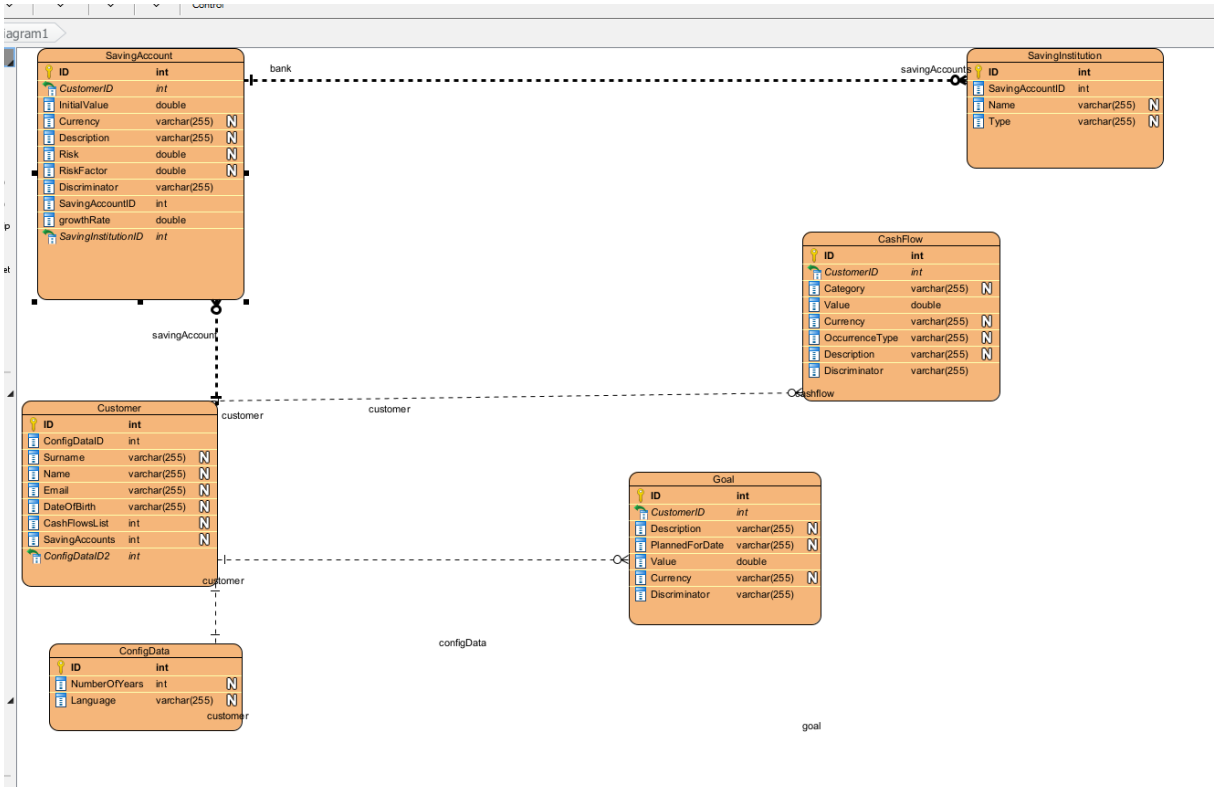

Figure 5 Entity diagram of the Web Savings Calculator API

As we can see, the customer has 1:m relations with the goals, cashflows and saving accounts, where 1:1 relation with the configuration data entity. The annual values are not saved into the database since they are calculated dynamically as of today until the chosen number of years that the client has chosen.

IV Class diagram

This section contains the class diagrams for the domain models, controllers, and services.

Domain model

Figure 6 shows the class diagram for the domain models. The customer is the aggregate domain model and all the content inside of the application is dependent on it.

The Savings account are represented with their types using inheritance from the base class Saving Account. The same approach is done for the goals with the Goal class and for the Cash flows via the CashFlow class.

The Cash flows and Accounts have a common method to retrieve the calculation for the given number of years and there will be a follow up on the exact calculations.

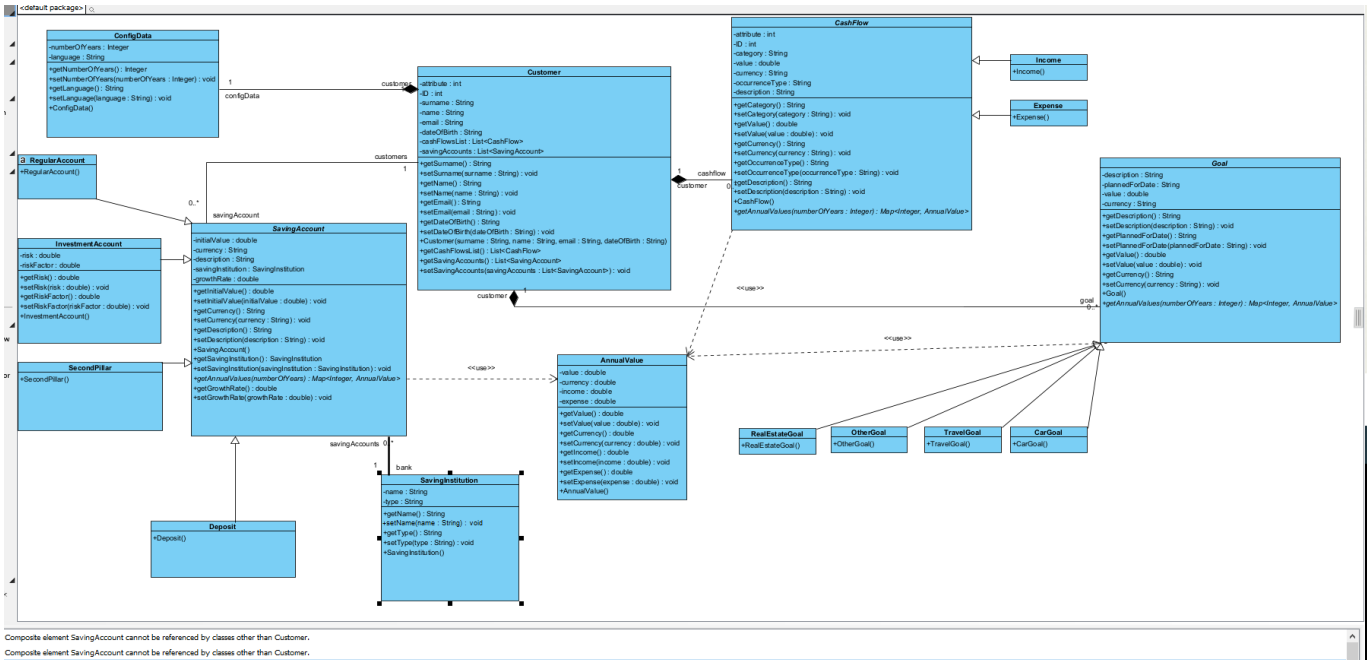

Figure 6 Class diagram for the domain models

Controllers

Figure 7 shows the Class diagram for the controllers. They are all implementation of the API generated by the Open api 3.0 definition.

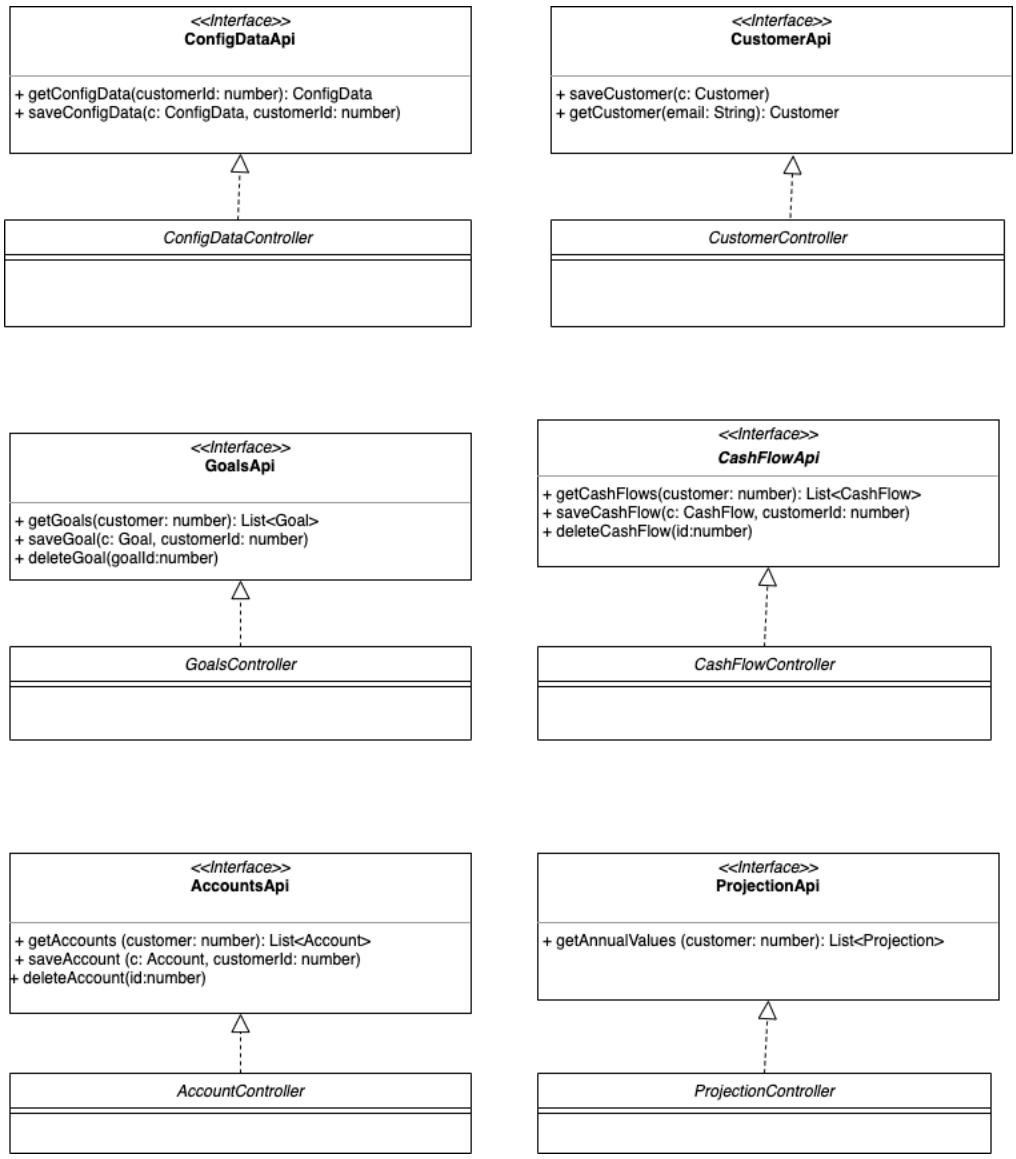

Figure 7 Class diagram for the controllers

Services

Figure 7 shows the class diagram for the services.

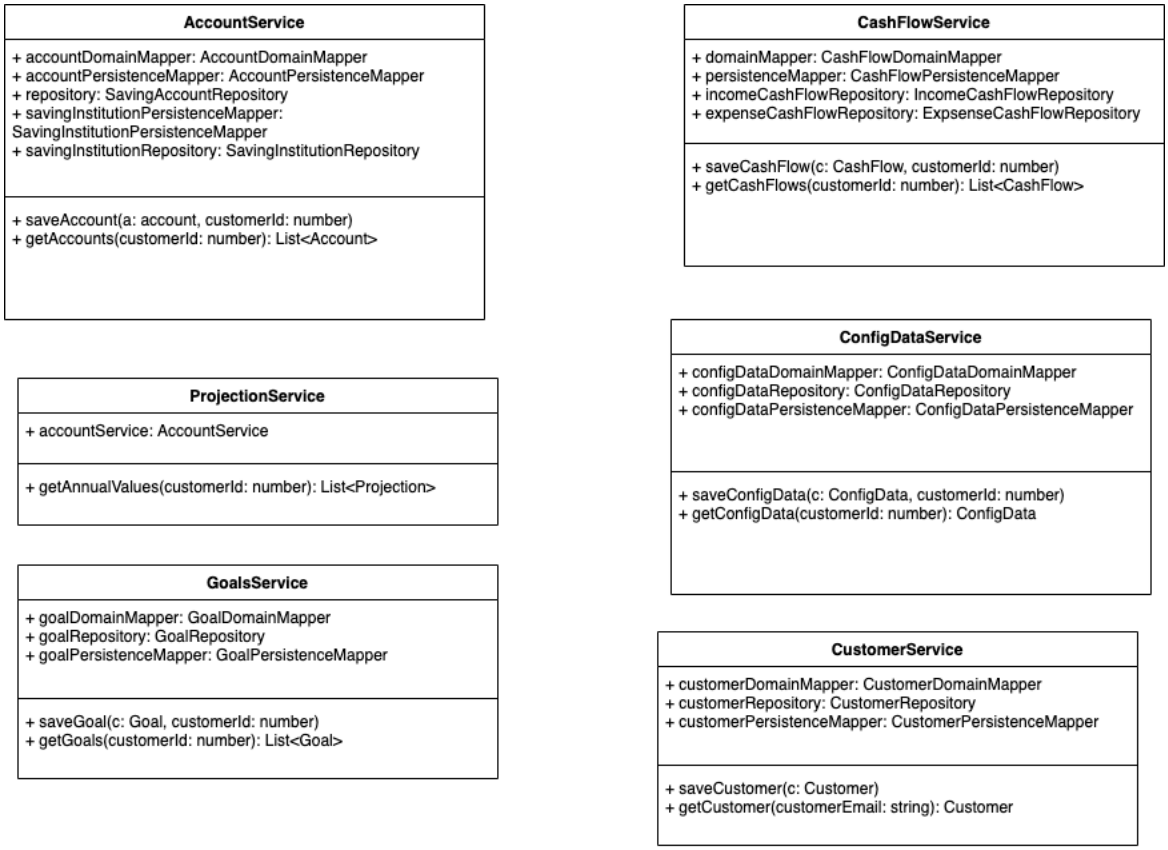

Figure 8 Class diagram for the services

V Sequence diagrams

Config data

Figure 9 shows the sequence diagram for retrieving the configuration data.

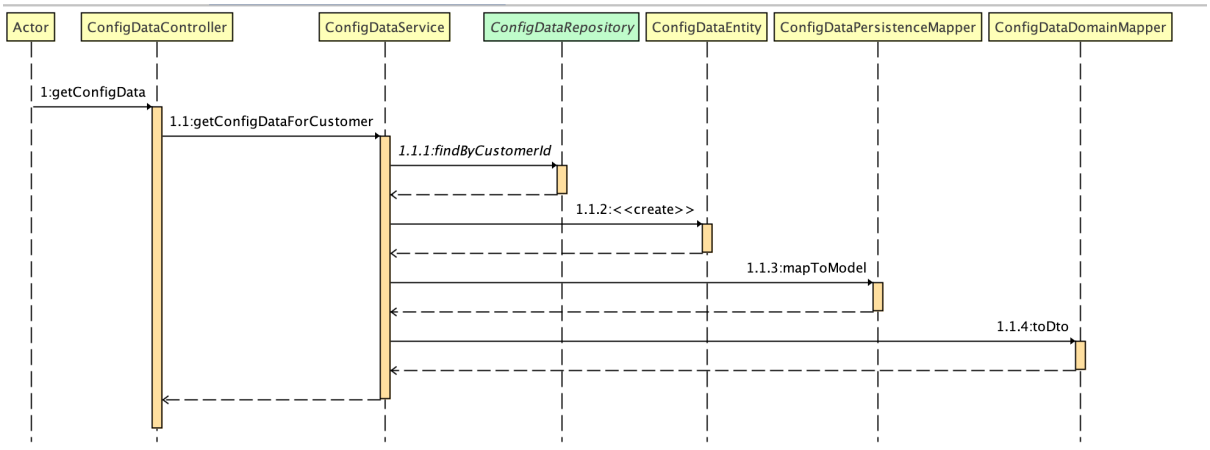

Figure 9 Sequence diagram for getConfigData

Figure 10 shows the sequence diagram for saving the configuration data.

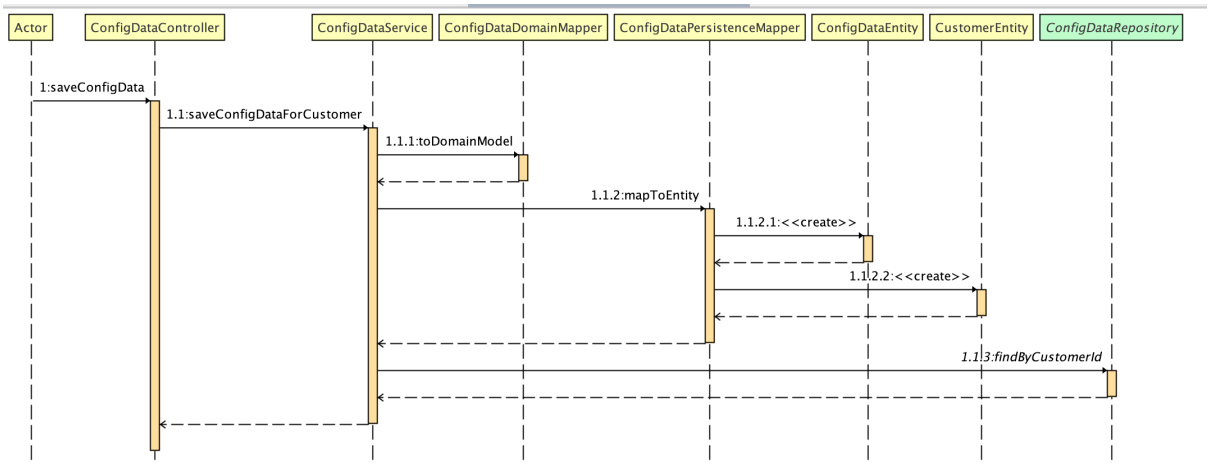

Figure 10 Sequence diagram for saveConfigData

Customer

Figure 11 shows the sequence diagram for retrieving the customer data.

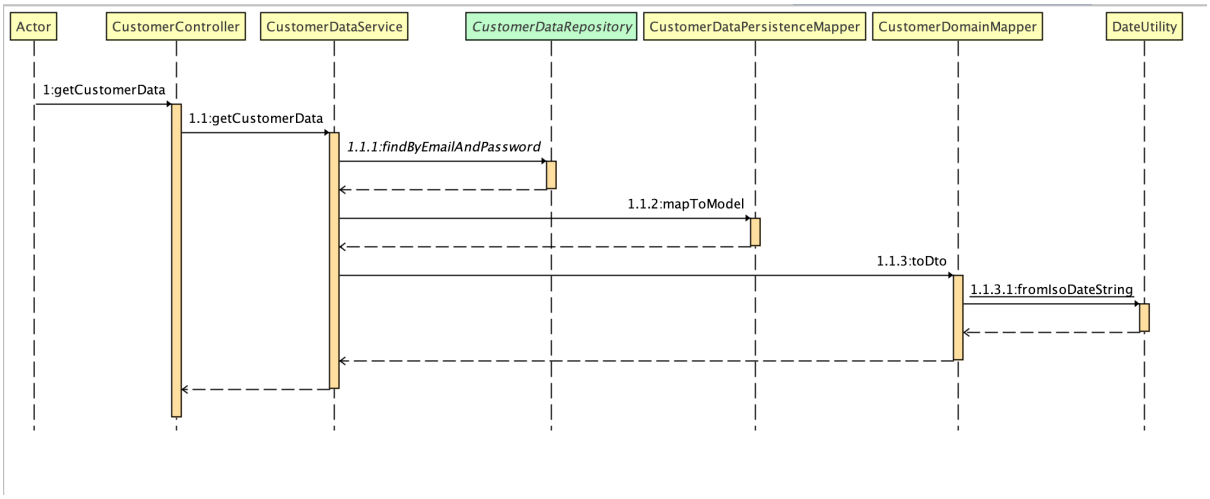

Figure 11 Sequence diagram for retrieving customer data

Figure 12 shows the sequence diagram for saving customer data.

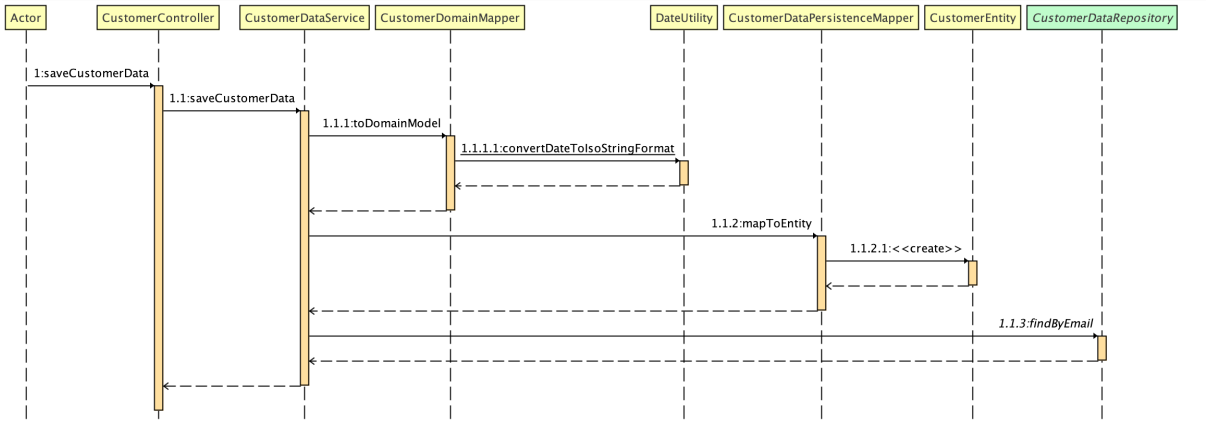

Figure 12 Sequence diagram for the saveCustomerData method

Goal

Figure 13 shows the sequence diagram for the saving of the goals.

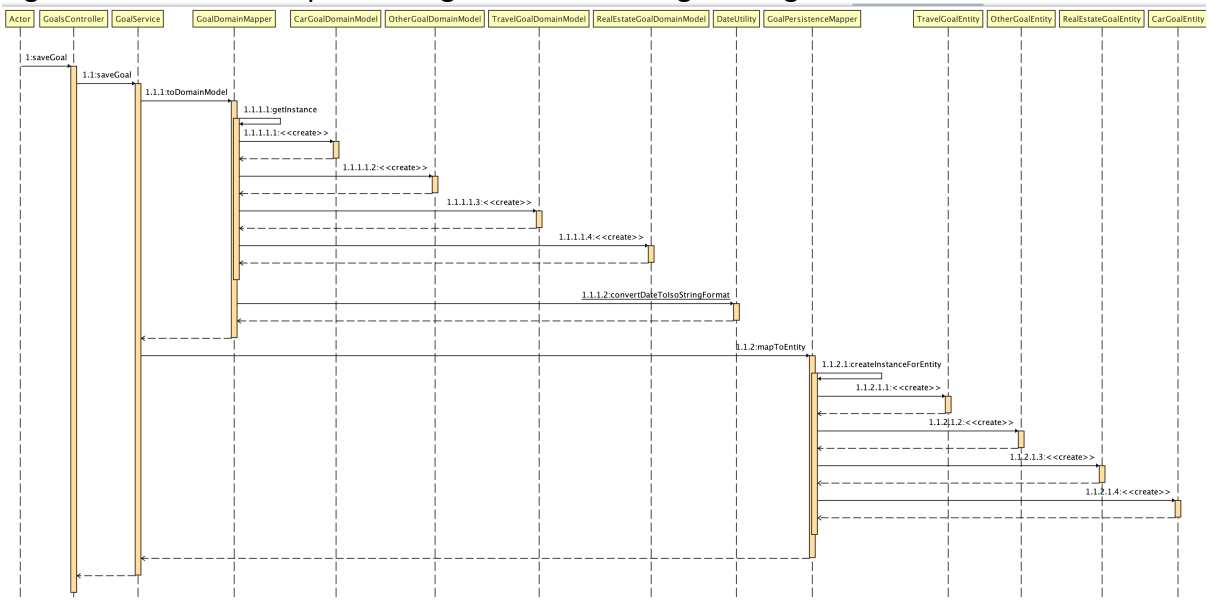

Figure 13 Sequence diagram for saving goals

Figure 14 shows the sequence diagram for the retrieval of the goals.

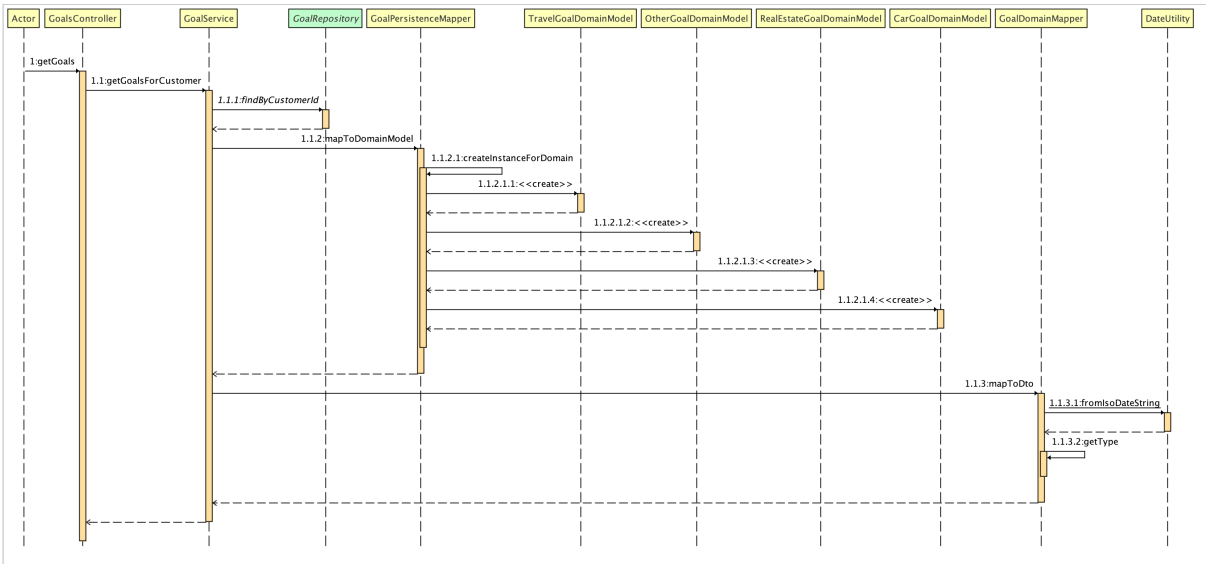

Figure 14 Sequence diagram for retrieving the goals

Figure 15 shows the flow for deleting goal.

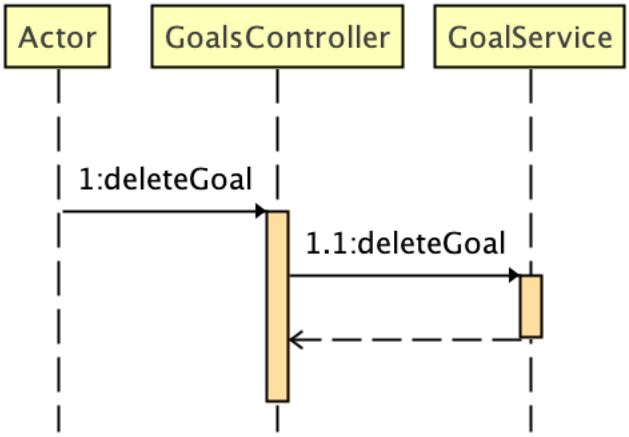

Figure 15 Flow of deleting goal

Cash flow

Figure 16 shows the sequence diagram for saving of the cash flows, where figure 17 shows the flow of retrieving the cash flows. Figure 18 shows the flow for deleting the cash flows.

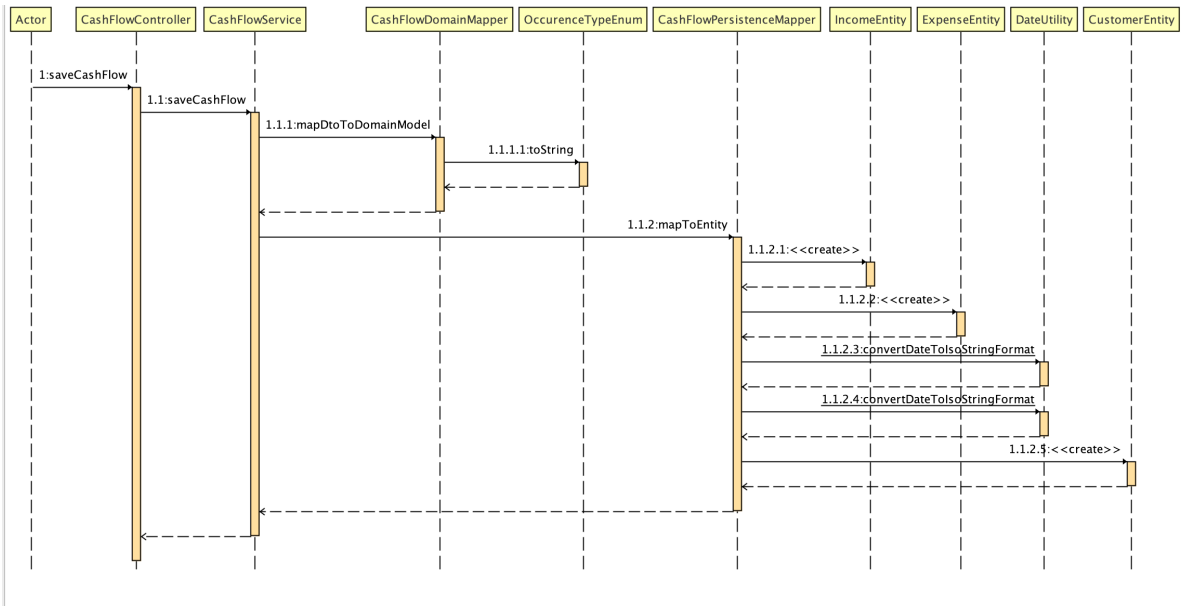

Figure 16 Sequence diagram for saving cash flow

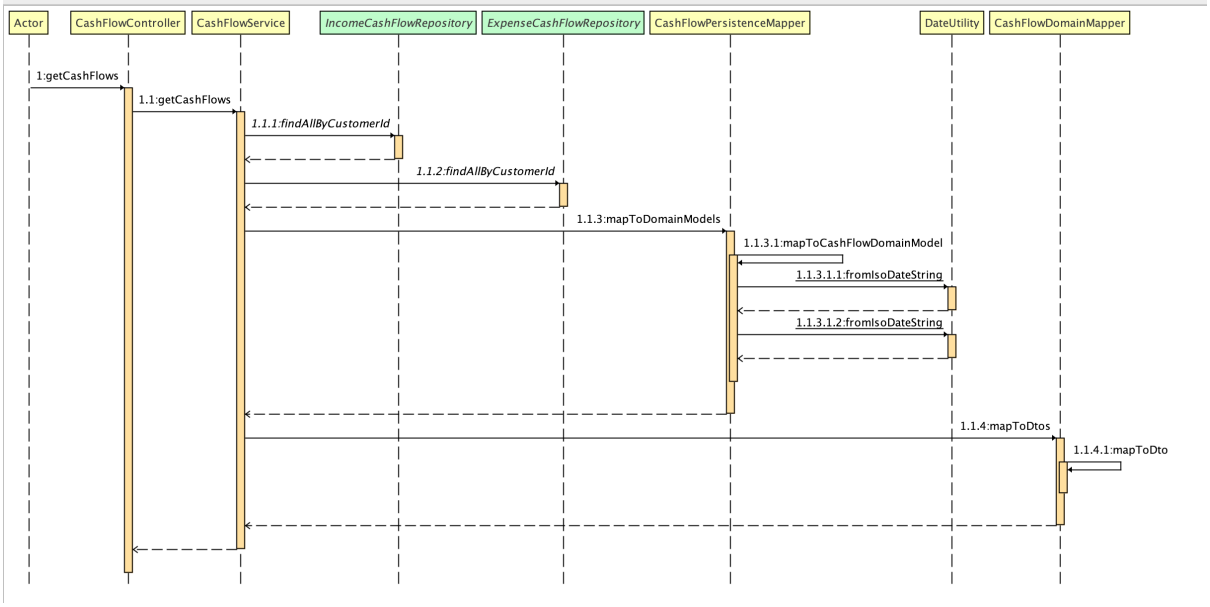

Figure 17 Sequence diagram for retrieving the cash flows

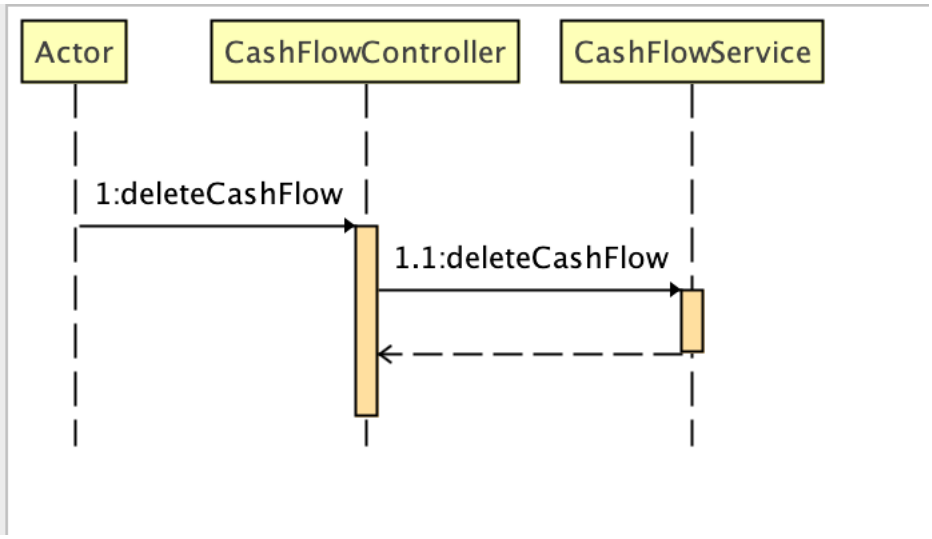

Figure 18 Flow for deleting cash flow

Account

Figure 19 shows the sequence diagram for saving the accounts, and Figure 20 shows the sequence diagram for retrieval of the accounts.

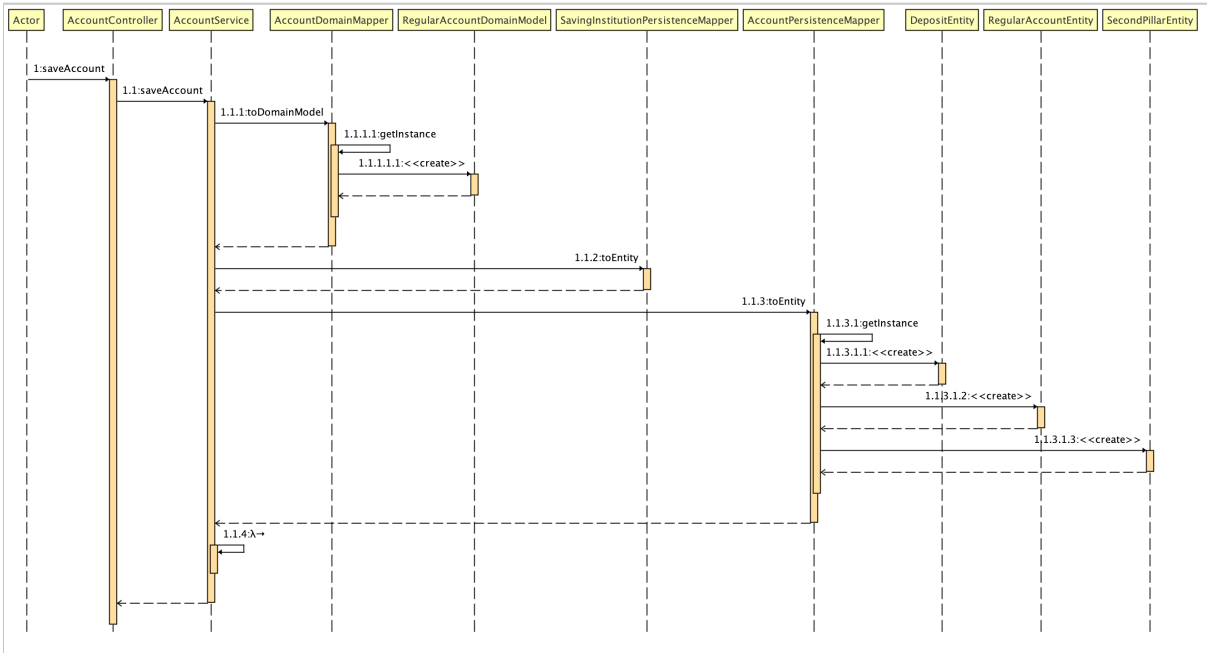

Figure 19 Sequence diagram for saving accounts

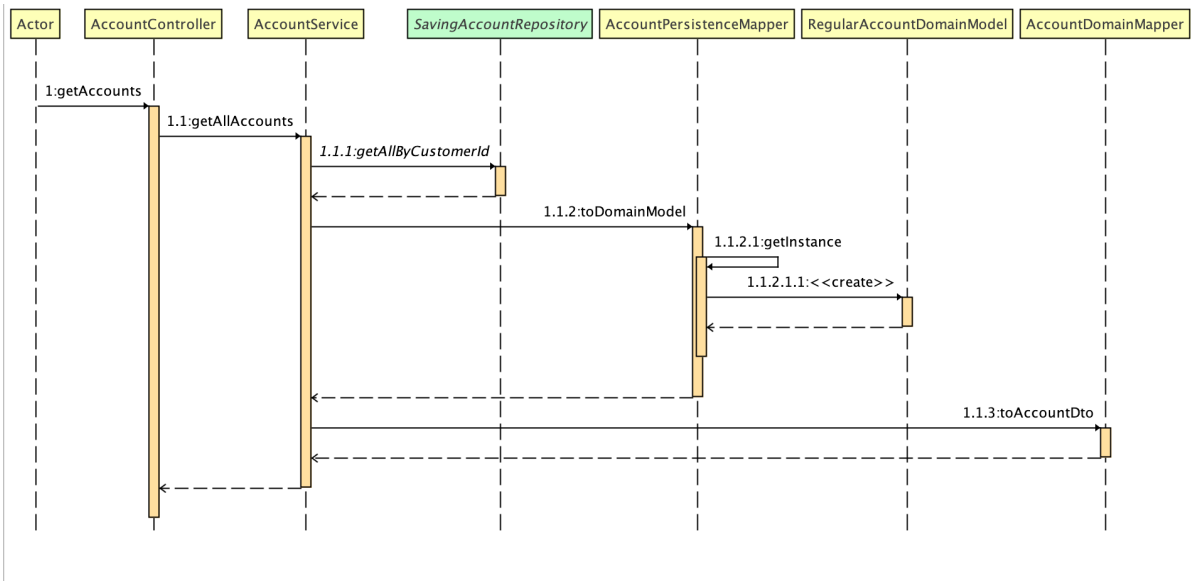

Figure 20 Sequence diagram for retrieving accounts

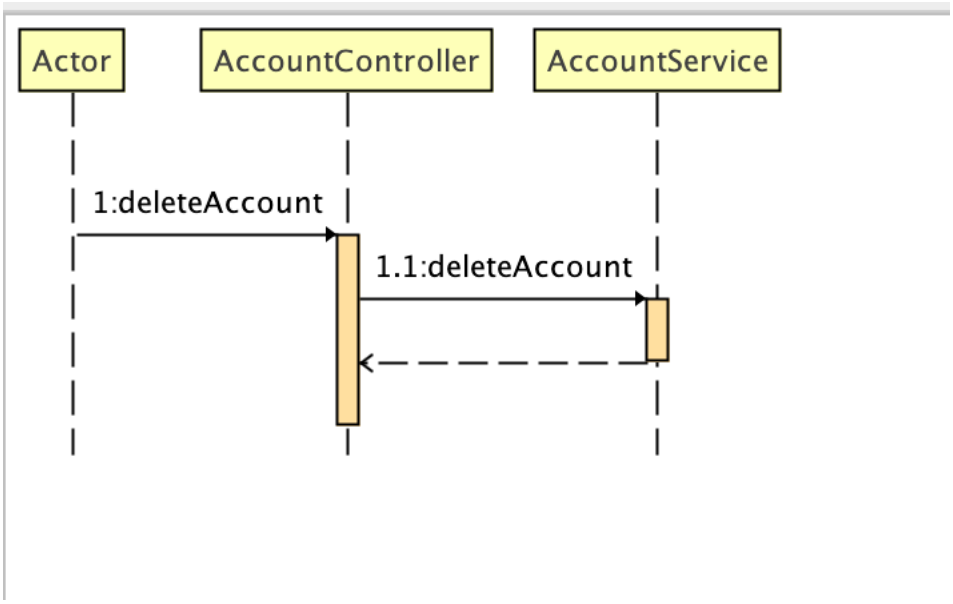

Figure 21 Sequence diagram for deleting account operation

Projection

The method for the projection annual values retrieval is probably the most complex one since it gathers the annual projections from each account and cash flow and amalgamates the response. Figures 22 and 23 show the sequence diagram of this method.

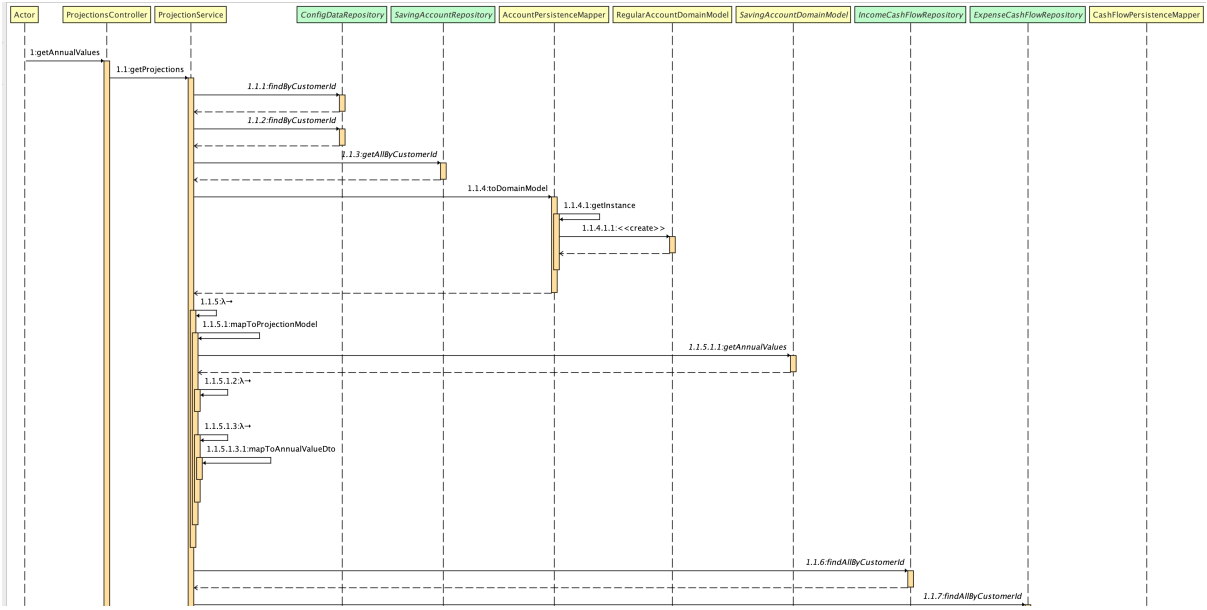

Figure 22 Sequence diagram for retrieval of annual values part 1

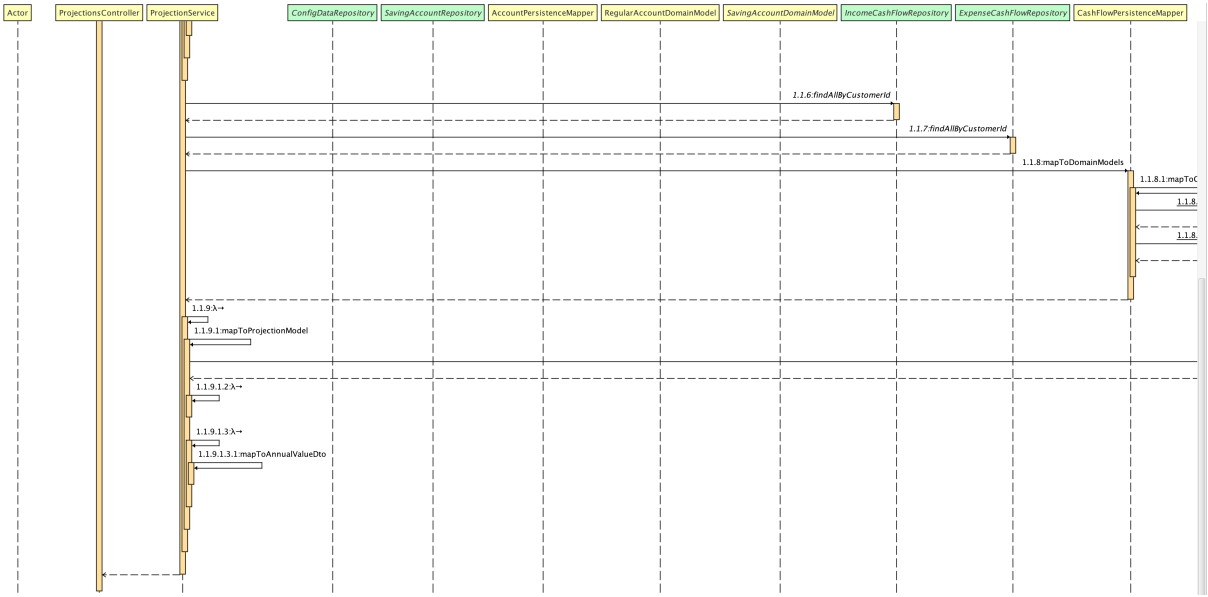

Figure 23 Sequence diagram for retrieval of annual values part 2

VI Algorithms / formulas for annual values calculation

This section contains the logic for the annual values calculation for the accounts and the cash flows.

Cash Flow

The cash flows are one – time or recurring payments / expenses that the client has.
The pseudo code for the calculation is the following:

```
* Check whether the cash flow from and to date are in the range of the
current date until the number of years.
* If the cash flow is one time, then create annual value with the complete
value and add it to the income / expense
```

```
* section.
* If the cash flow is recurring, iterate through the range of today until
the number of years and add the recurring
* income / expense in the annual section.
```

Account

The accounts are separated into several types so we will check the calculation for each one of them.

Regular account

For the regular account there is no growth rate so the value will be always the same. The annual values will be filled with value equal to the initial value of the account.

Deposit account

Value for a given year is calculated as:

Value (year) = value (year – 1) * growth rate

Because the growth rate is on yearly level. If the payout date is not in the projection range, then the calculation will be ignored, and the values will be empty.

Second pillar account

The value for year x is the pension per year or the planned lump sum if it is the retirement year. Of course, the retirement year must be in the projection scope.

Investment account

The initial value is multiplied with the growth rate and then multiplied the risk factor so that the risk is applied.

VII Graphical interface

This section contains detailed overview of the graphical interface of the application.

Homepage

Figure 24 shows the look of the homepage. The Homepage allows the user the option to enter the login credentials and login to the application or he / she can continue to the sign-up form.

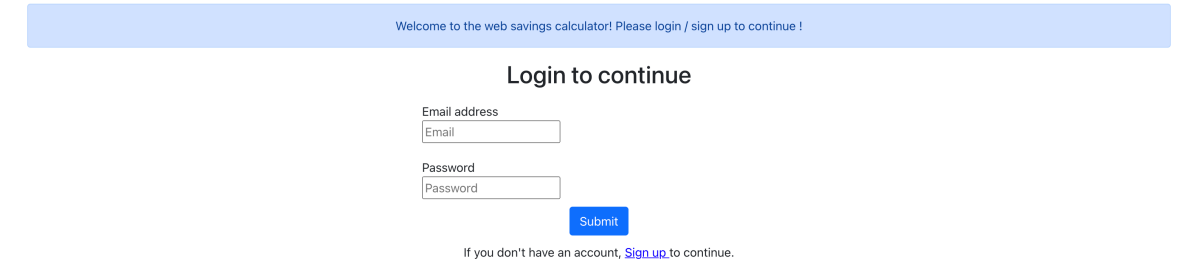

Figure 24 Homepage

In case of invalid credentials, an alert is shown. This scenario is captured on figure 25.

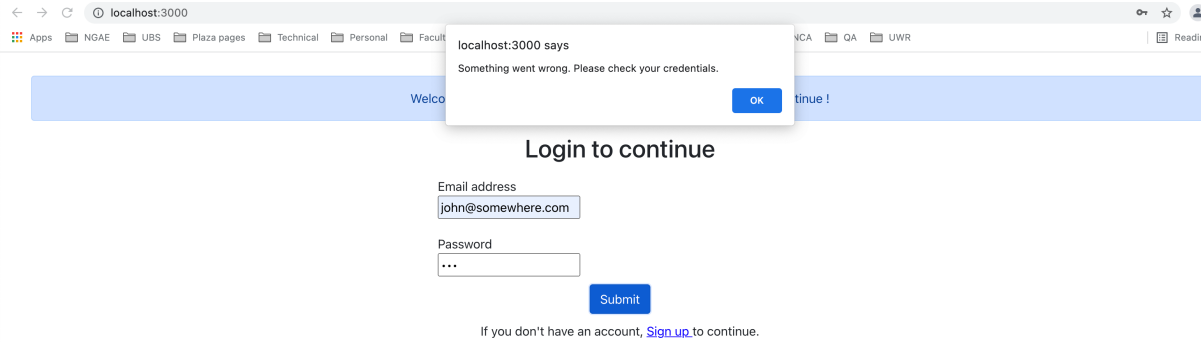

Figure 25 Scenario with invalid credentials

Sign-up

Figure 26 shows the sign-up form. The user is requested to enter his name, surname, date of birth, email and password. After successful registering, the user receives a notification that he / she is registered and they are redirected to the login page. Figure 27 and 28 are capturing this scenario.

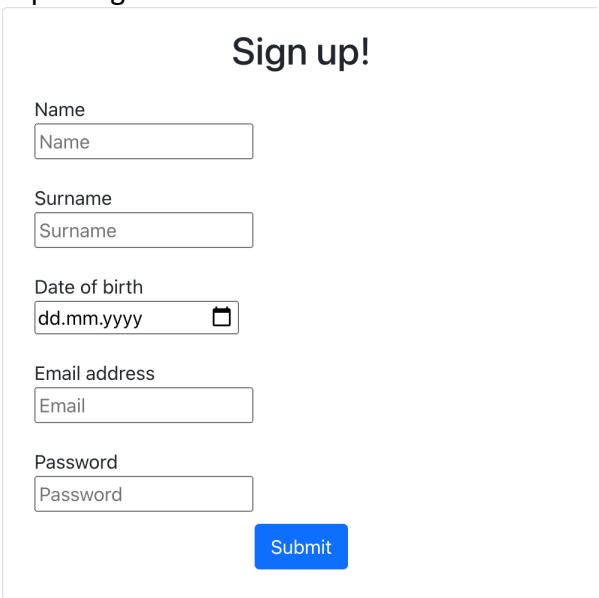

Figure 26 Sign up form

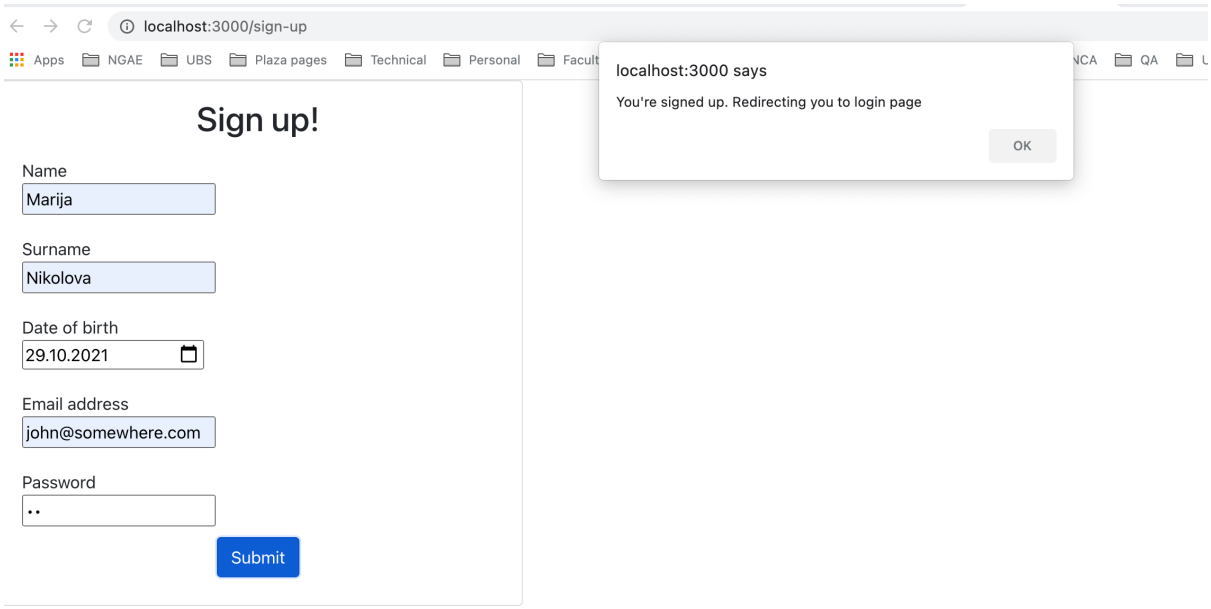

Figure 27 Alert after successful register

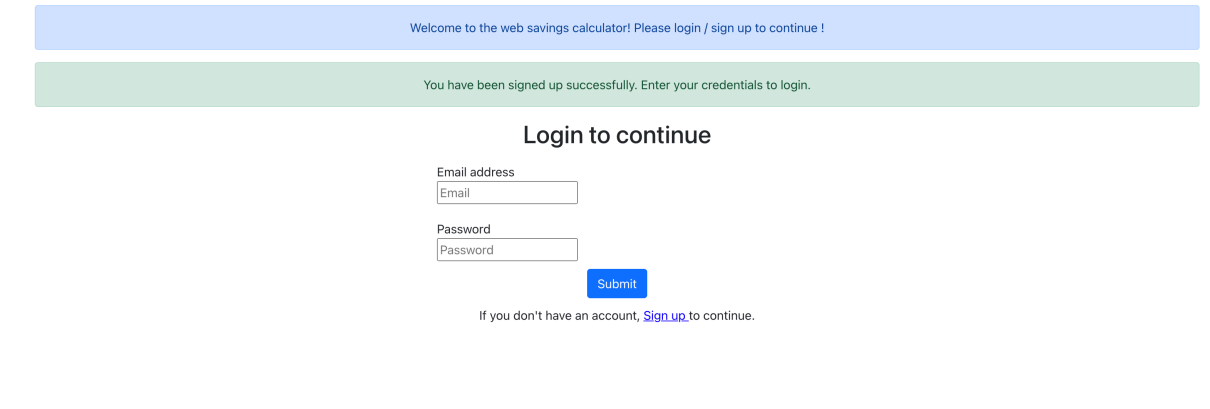

Figure 28 Login page after successful registering

Dashboard

After successful login, the user sees the dashboard page which is shown on figure 29. The dashboard page contains links to enter all submenus in the application: Personal data, goals, accounts, cash flows and projection page.

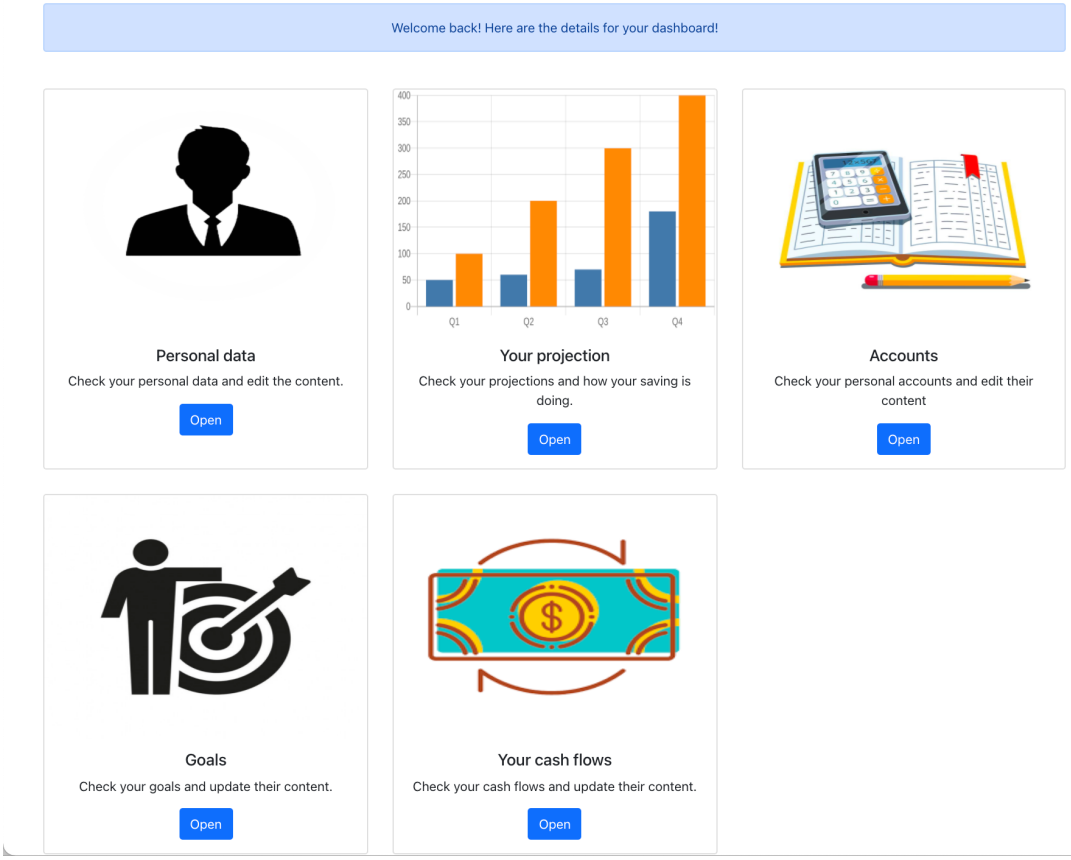

Figure 29 Dashboard

Personal data

In the Personal data page, the client can edit some of his / her private data as well as to enter the configuration data like number of years for projection and the language in which the application will be served. The look is shown on figure 30. As it can be seen, the UI offers the possibility to update data or to navigate back to the dashboard.

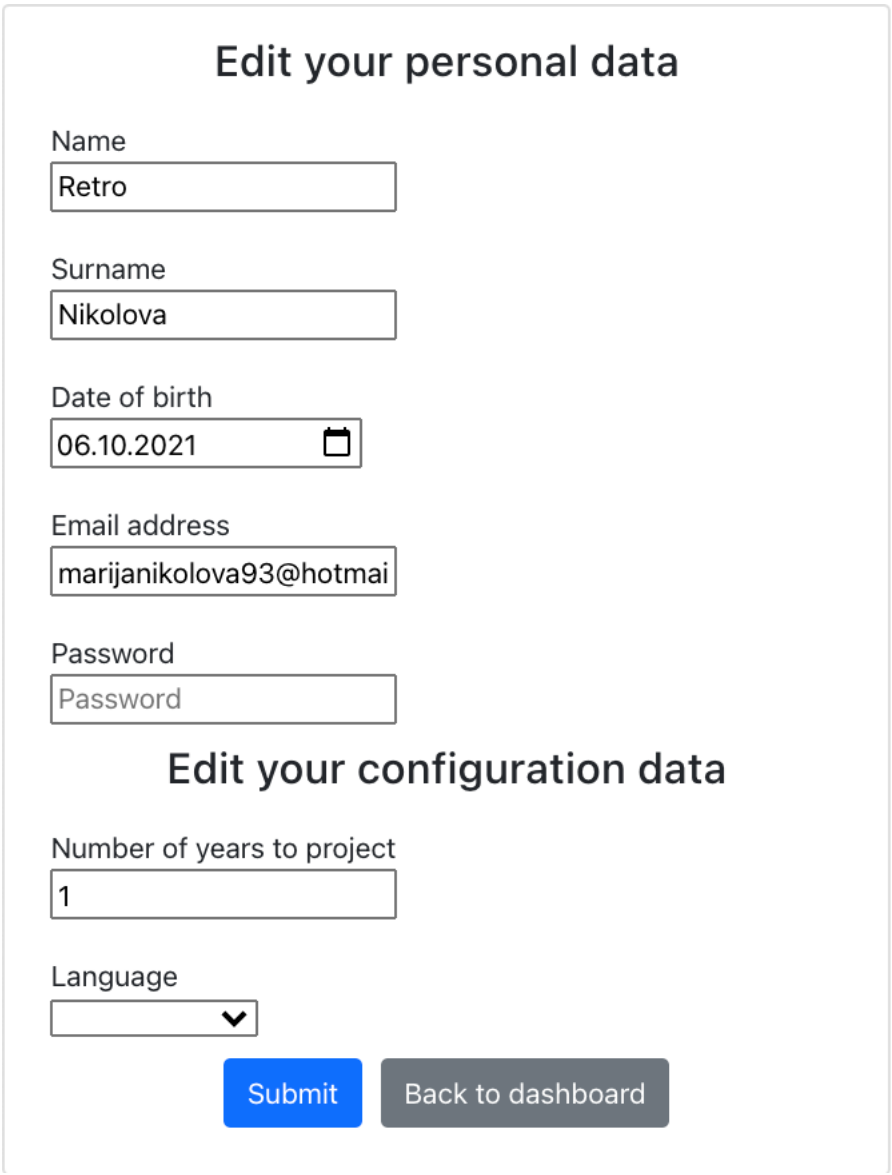

Figure 30 Personal data

In case the user edits something, after the successful saving, a message is triggered, as shown on figure 31.

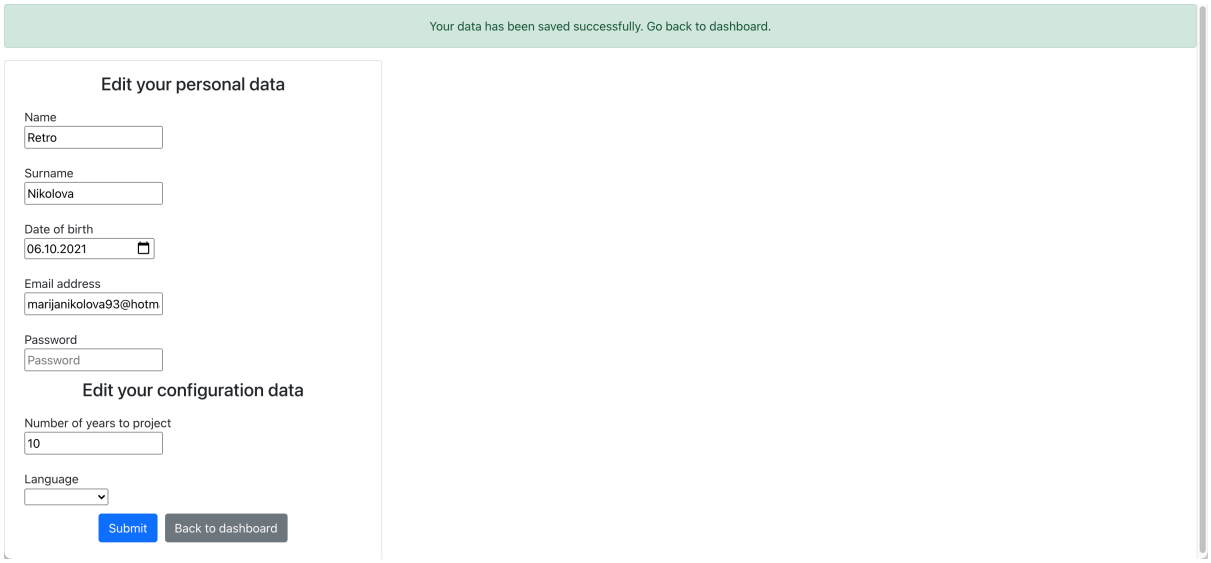

Figure 31 Saving personal data

Accounts

The accounts page is listing all existing accounts and it offers possibility to edit existing as well as adding new accounts. There is also navigation back to the dashboard. Figure 31 shows the view of this page.

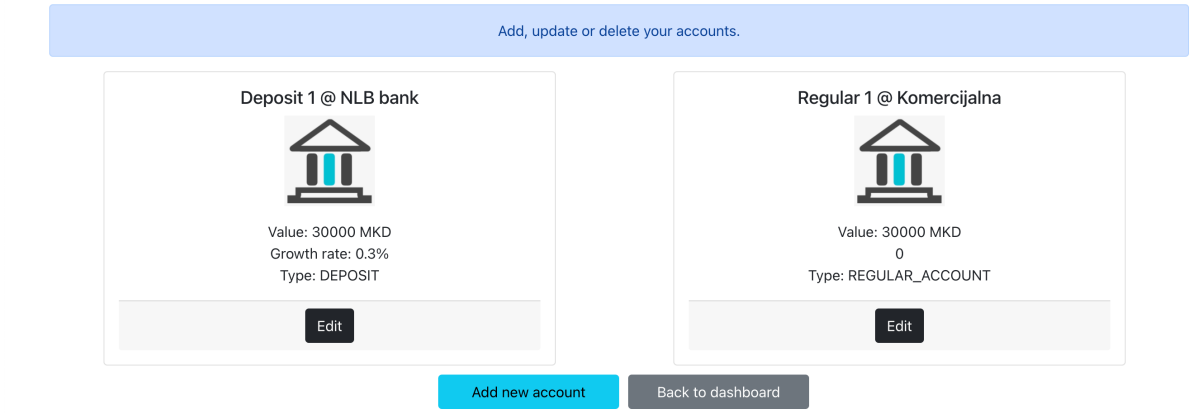

Figure 32 Accounts page

Figure 33 shows the form for entering an account. After saving, the user is automatically redirected to the accounts page.

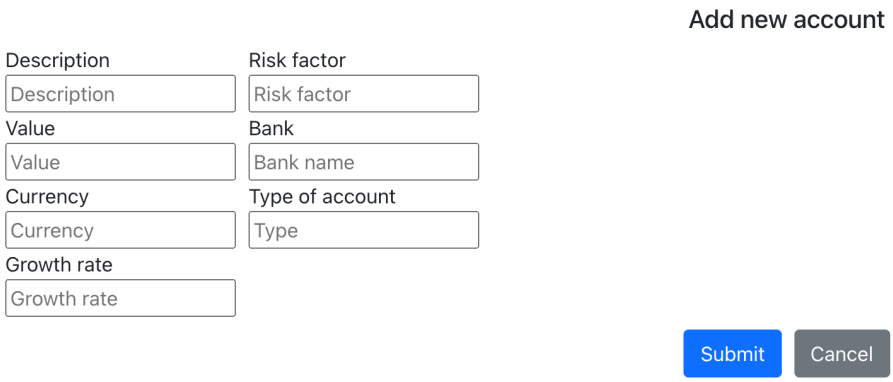

Figure 33 Form for adding new account

Figure 34 shows how the user can edit an already existing account. As we can see, the client has the possibility to update some of the entry fields or to delete the account. The cancel option is showing the view mode of the account.

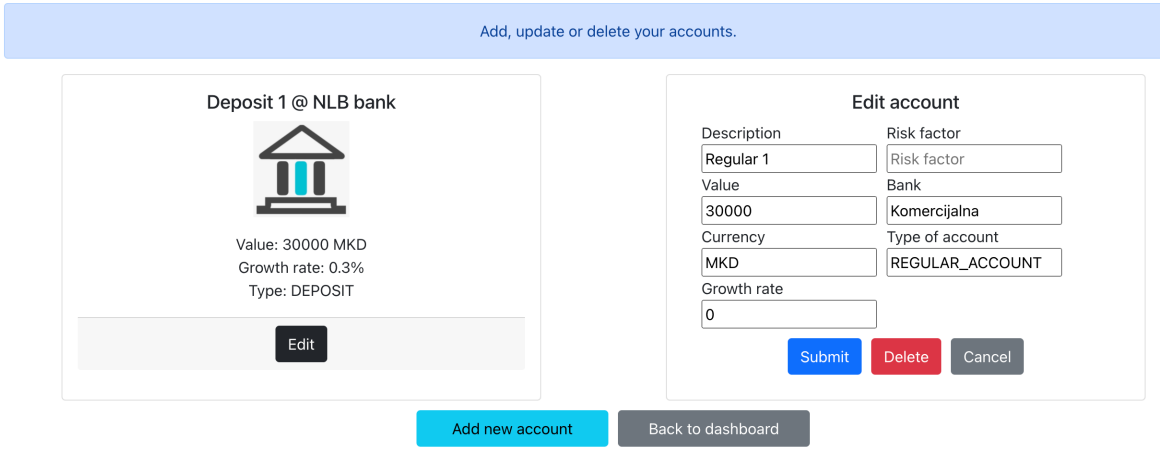

Figure 34 Editing already existing account

Goals

Figure 35 shows the content of the goals page. It lists all already existing goals, offers possibility for saving new goal or updating the existing ones. We can also go back to the dashboard.

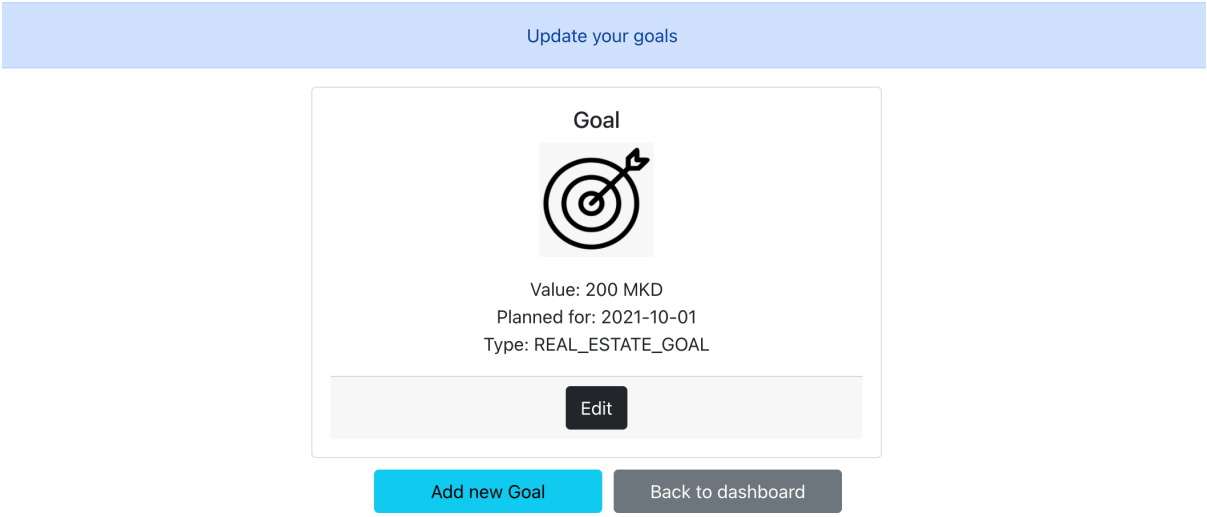

Figure 35 Goals page

Figure 36 shows the form for adding new goals.

Figure 36 Add new goal

Figure 37 shows the option for editing / deleting the goal.

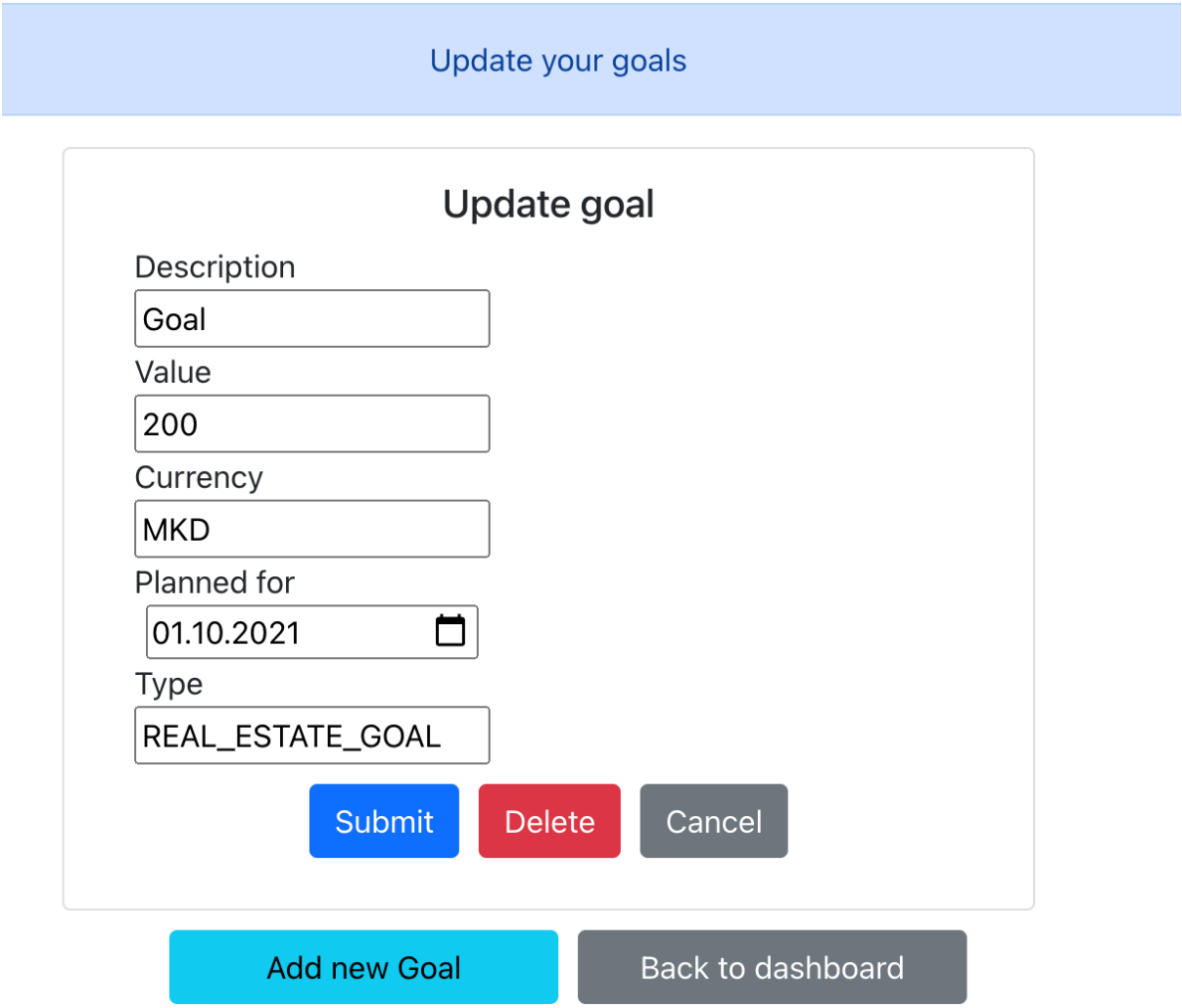

Figure 37 Update goal

Cash flows

Figure 38 shows the cash flows page. It shows all existing cash flows with possibility to enter new ones and update the existing ones.

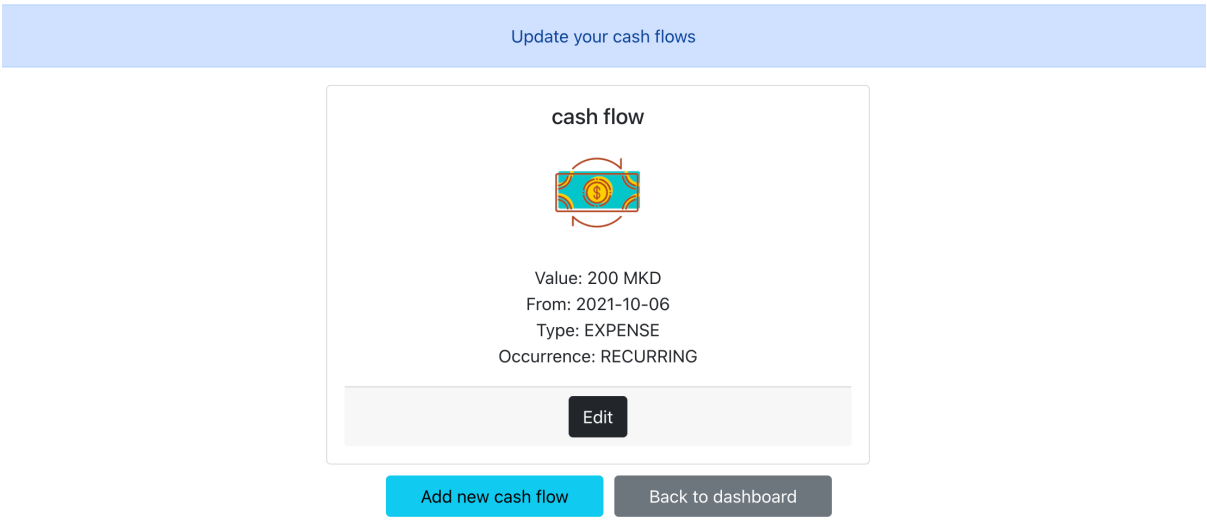

Figure 38 Cash flow page

Figure 39 shows the page for adding new cash flows.

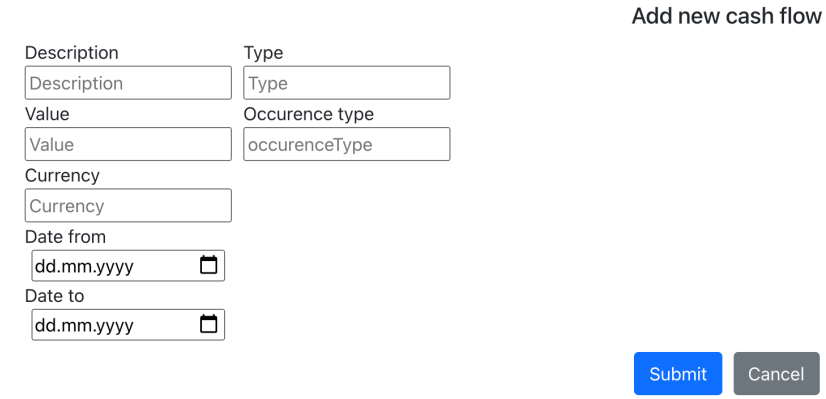

Figure 39 Adding new cash flow

Figure 40 shows the option for updating existing cash flow.

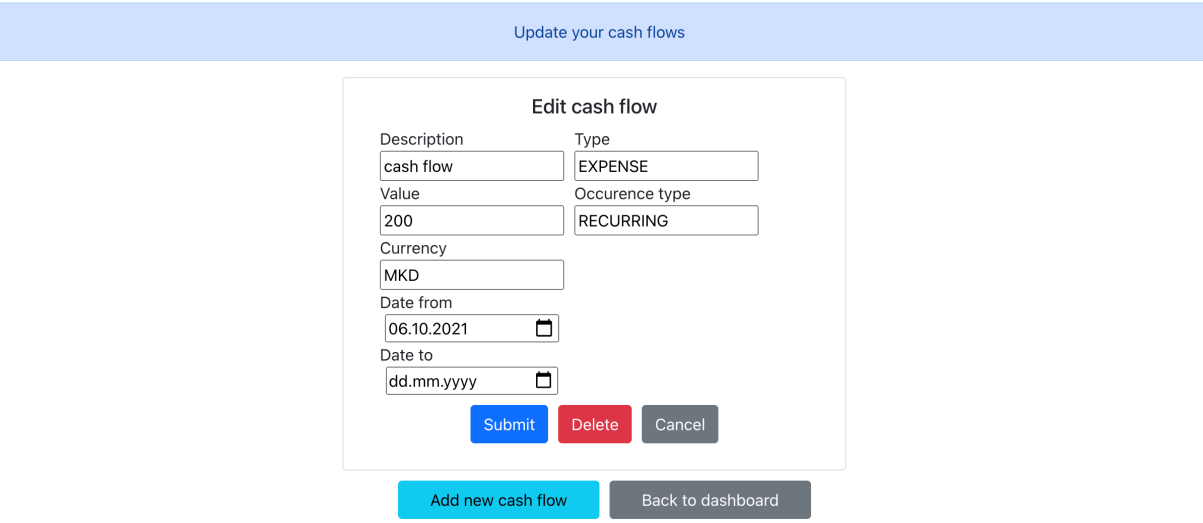

Figure 40 Update of cash flows

Projection

The projection page shows an overview of all accounts and cash flows, how they are developing through the years as well as the goals which the client has to keep in mind. This is shown on figure 41 and 42.

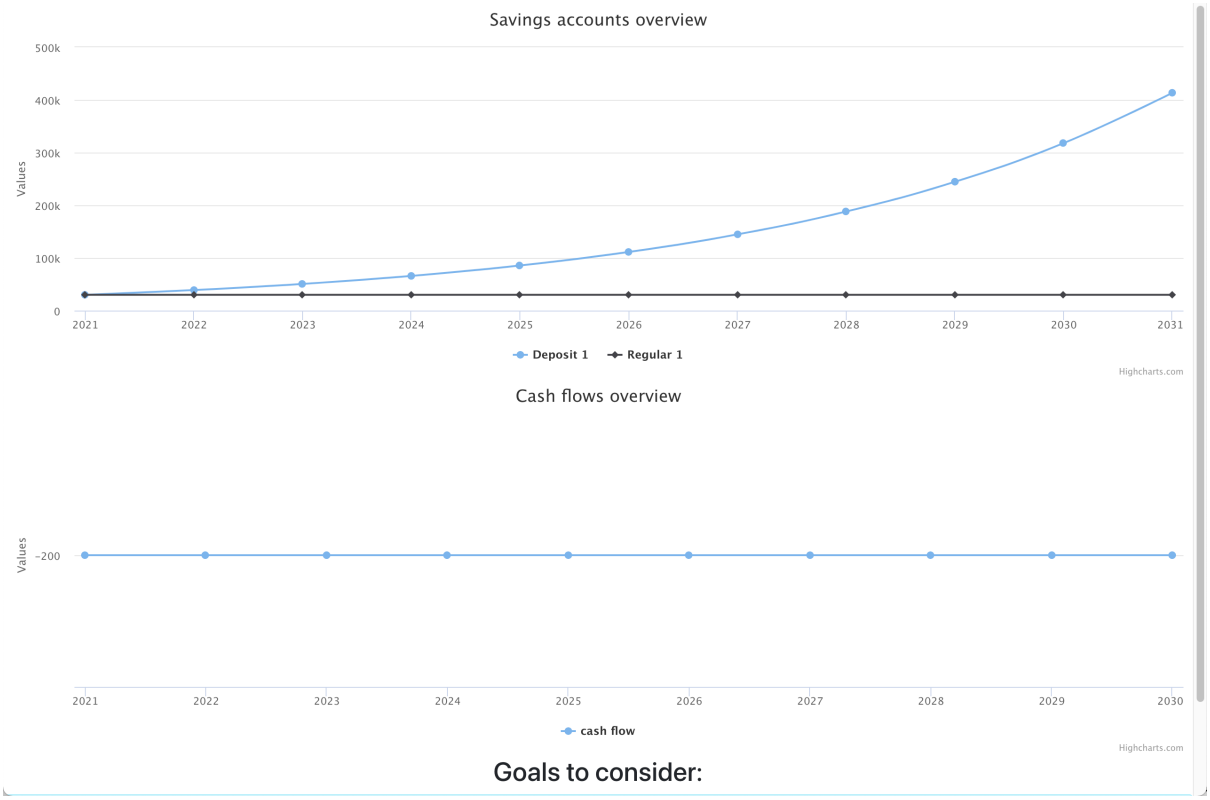

Figure 41 Projection page - part 1

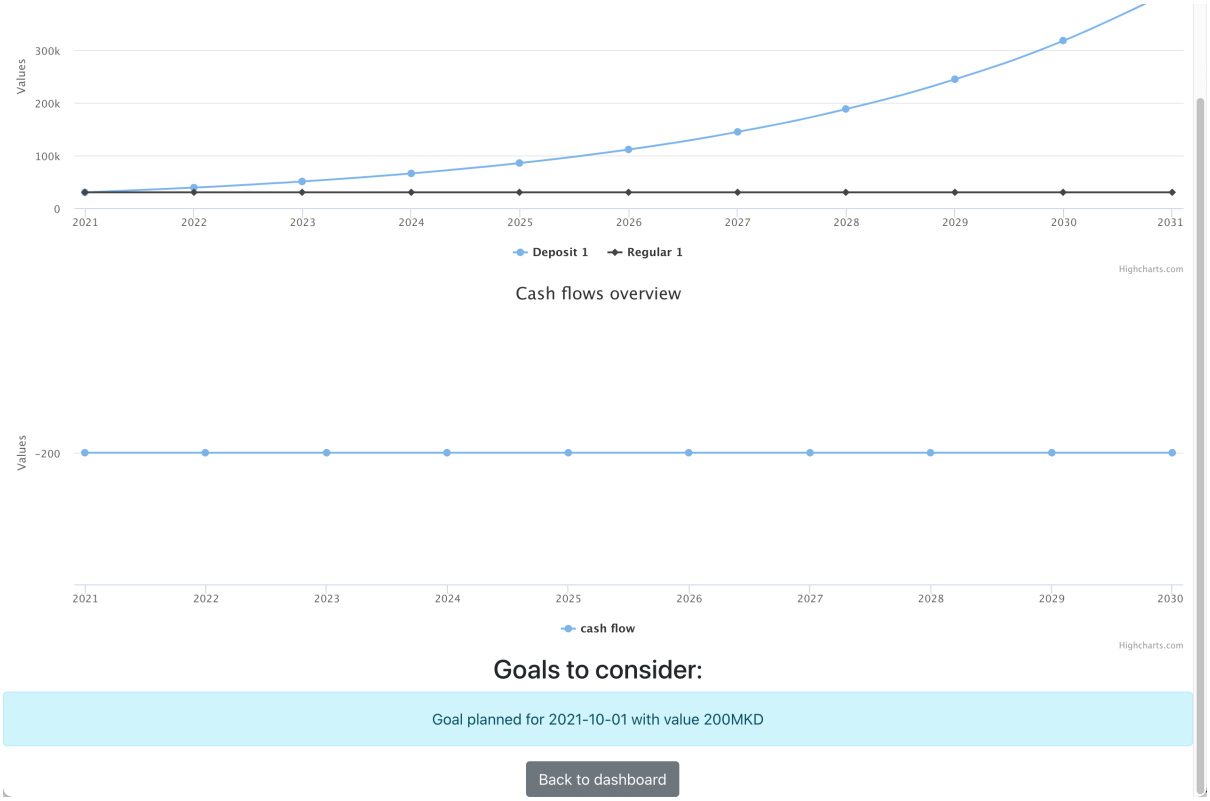

Figure 42 Projection page - part 2

As we can see, the goals are listed at the bottom and there is also the option to go back to the dashboard.

Page not found

In case someone tries to access a page, which is not existing in the app, we will show warning that the page was not found, as shown on figure 43.

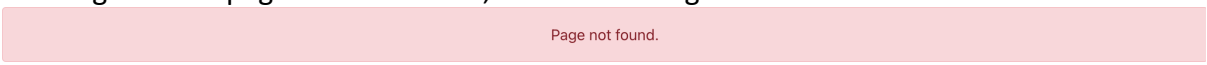

Figure 43 Page not found

User not logged in

In case the user tries to access an existing page and he is not logged in, then we show the page shown on figure 44.

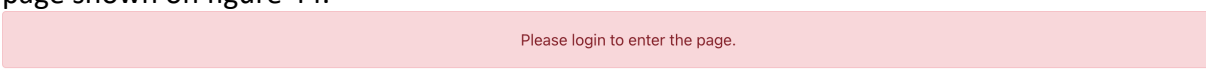

Figure 44 Login to enter page

VIII Project setup

Backend component

This section contains instructions to setup the project and start with development. This is useful for developers which just joined the project.

Cloning the project

The project is available on https://github.com/MarijaNikolova/WebSavings.git. Import the project with the option from VCS control as shown on figure 45.

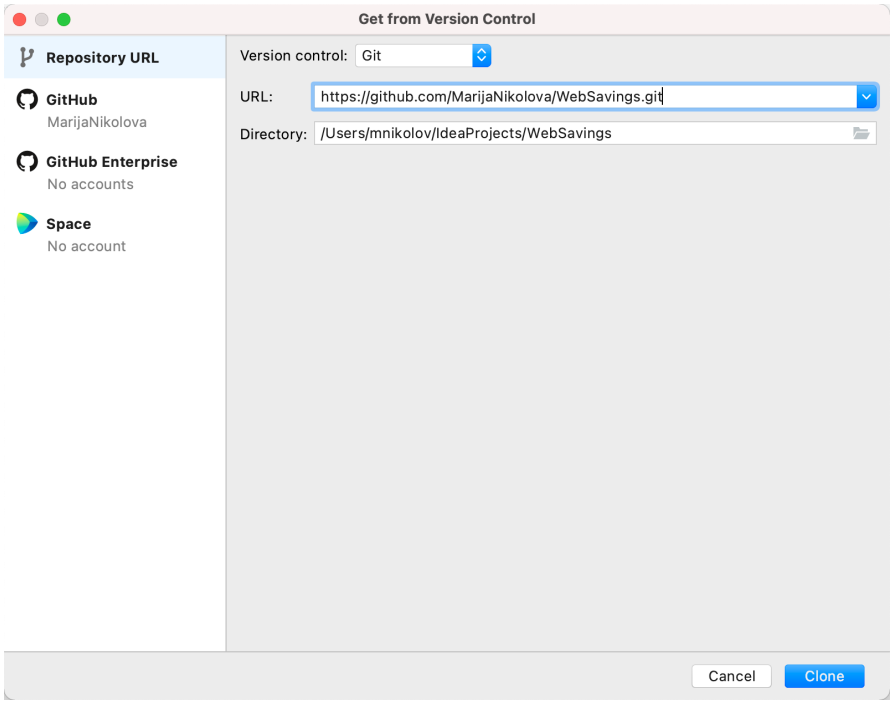

Figure 45 Importing the project from VCS control in Intellij

Building the project

After the project is successfully imported, since it is a maven project, we need to build it so that we resolve all dependencies and generate all files. For this, we need to run clean:install from maven, as shown on figure 46.

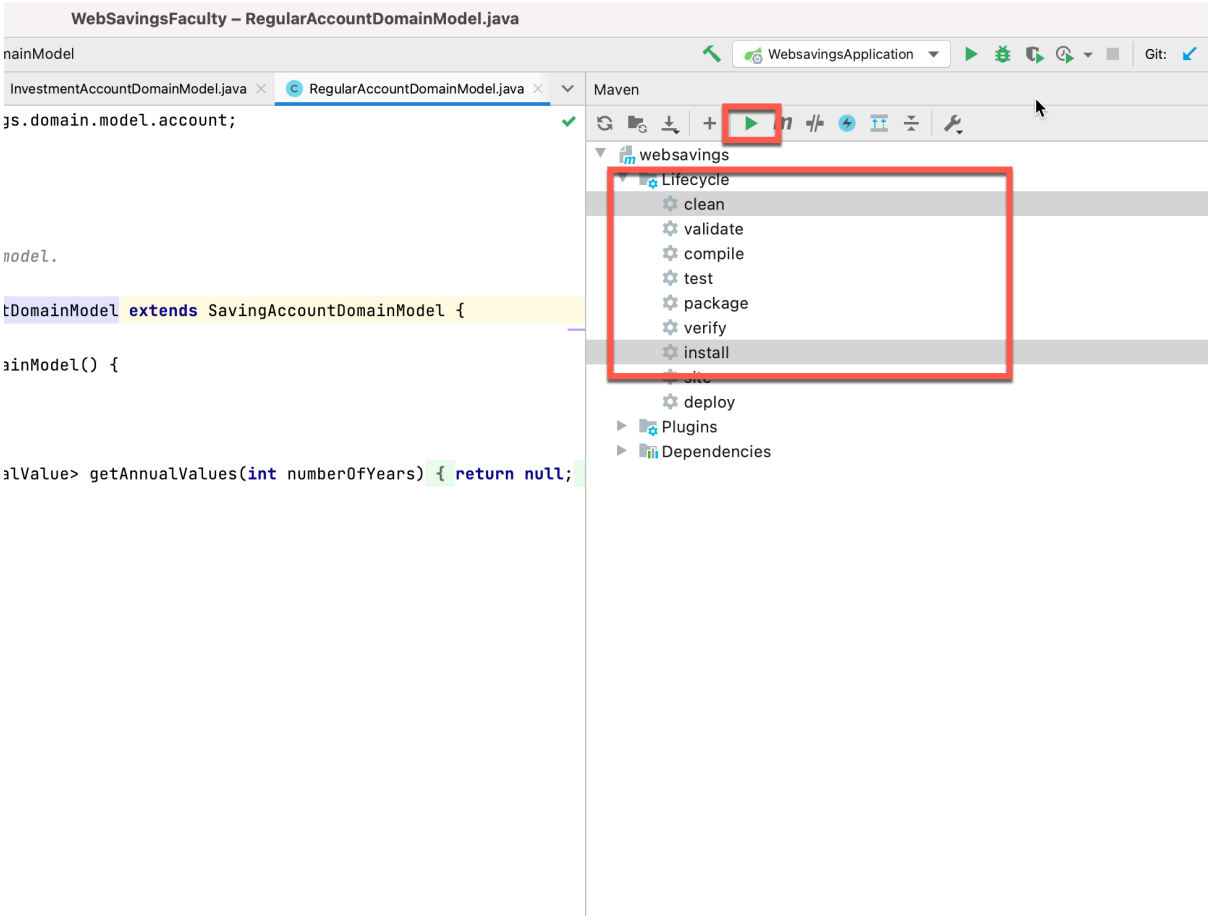

Figure 46 Clean install on the project

After the successful finishing of this operation, the project is ready to be started.

Starting the web application

For starting the web application, we need the run configuration shown on figure 47.

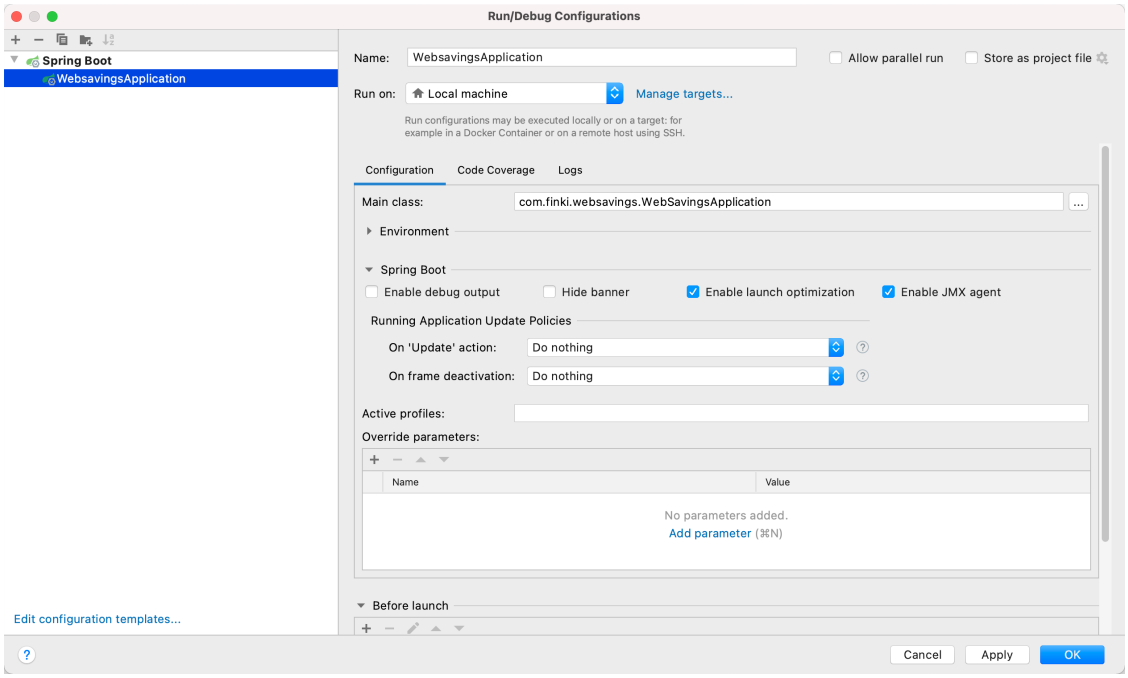

Figure 47 Web application start configuration

Checking the services in the browser

In the browser, we have to open the following URL: http://localhost:8080/app/websavings/swagger-ui/index.html. This will open the swagger UI shown on figure 48.

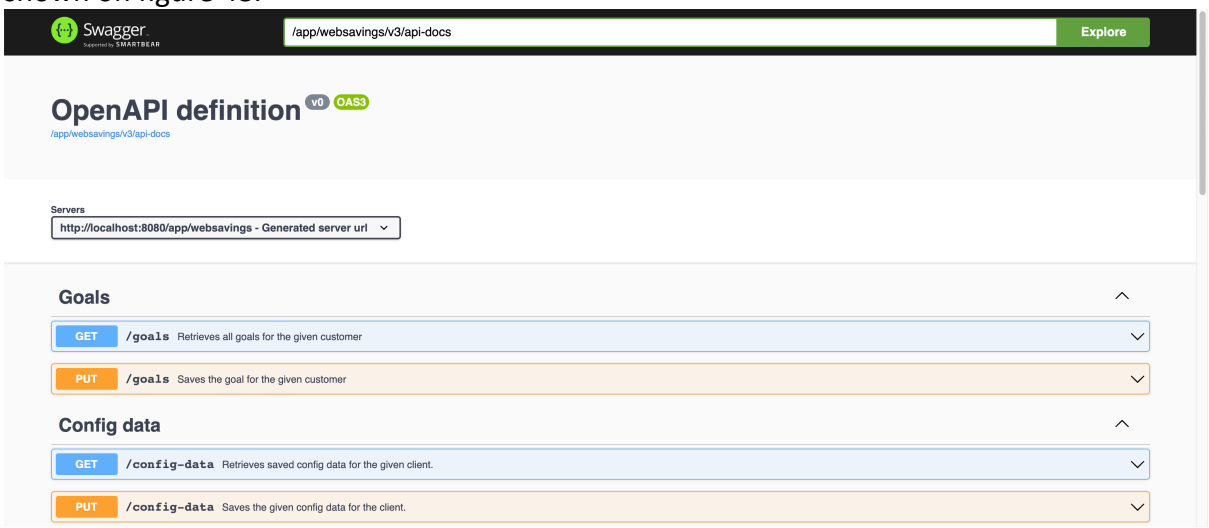

Figure 48 Swagger UI for the Web Savings Calculator

Figure 49 shows how we can execute the queries through this UI.

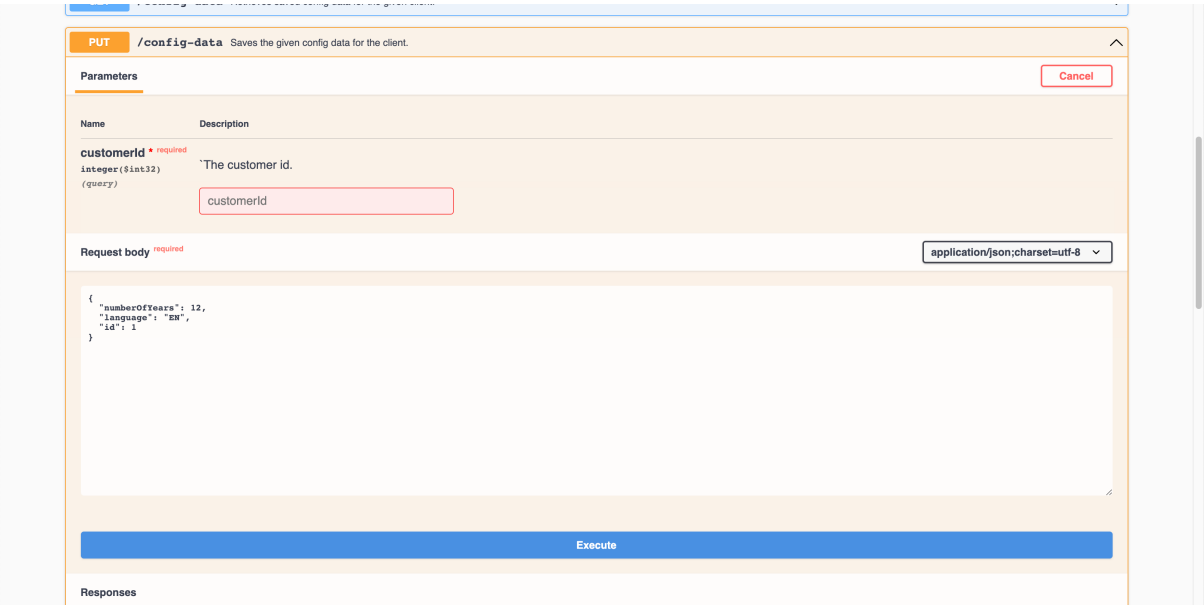

Figure 49 Example for executing the saving config data query

Frontend component

Cloning the project

First we have to import the project in Intellij as shown on figure 50.

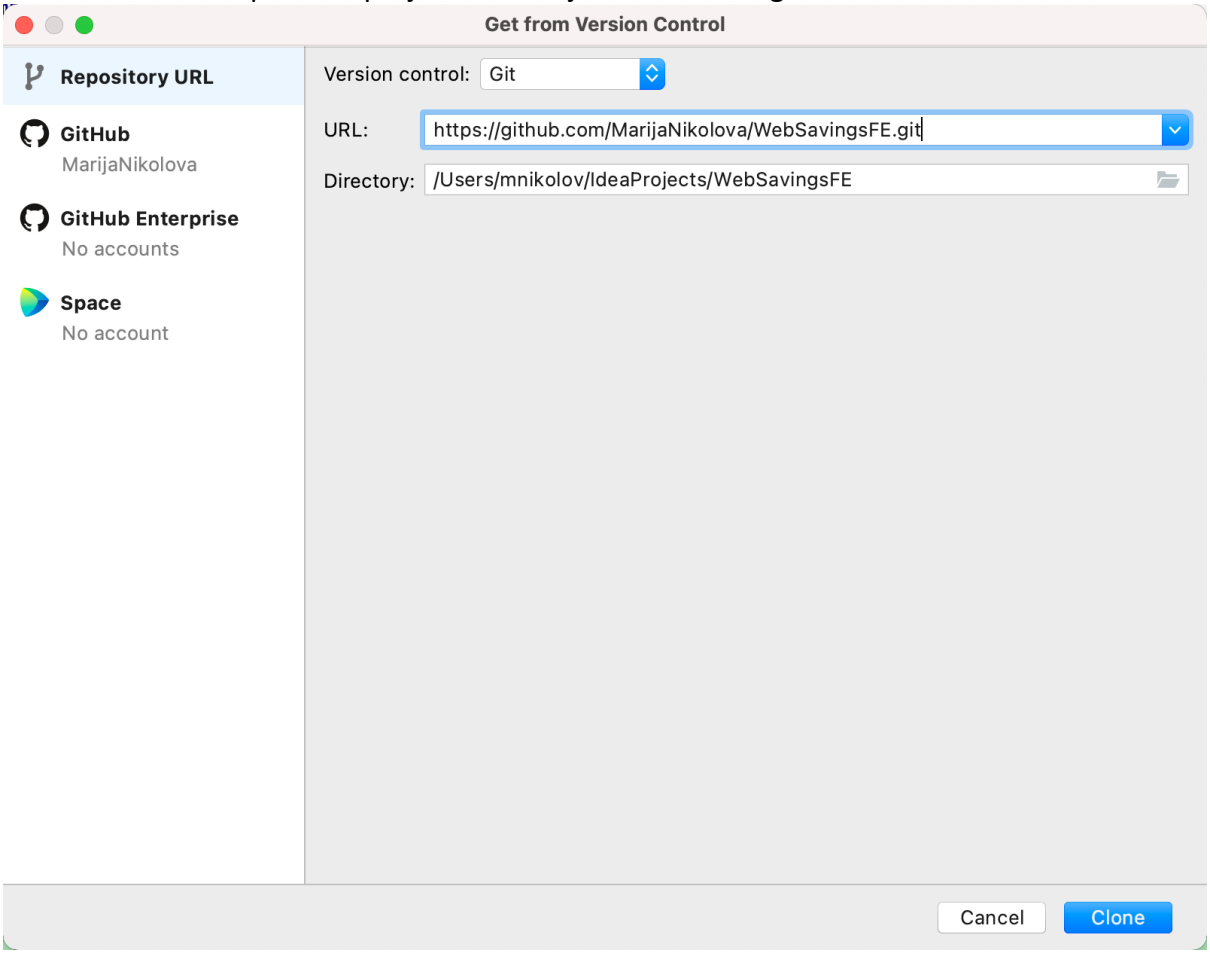

Figure 50 Cloning the FE component

Installing dependencies

Make sure that you have installed npm and yarn on your machine. After the checkout, open the terminal and execute yarn install, as shown on figure 51.

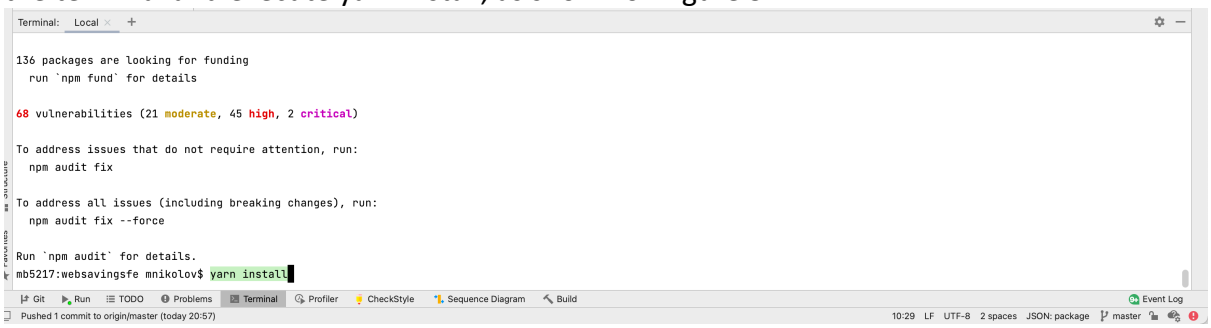

Figure 51 Installing fe dependencies

Local webpack configuration

Add webpack configuration as shown on figure 52.

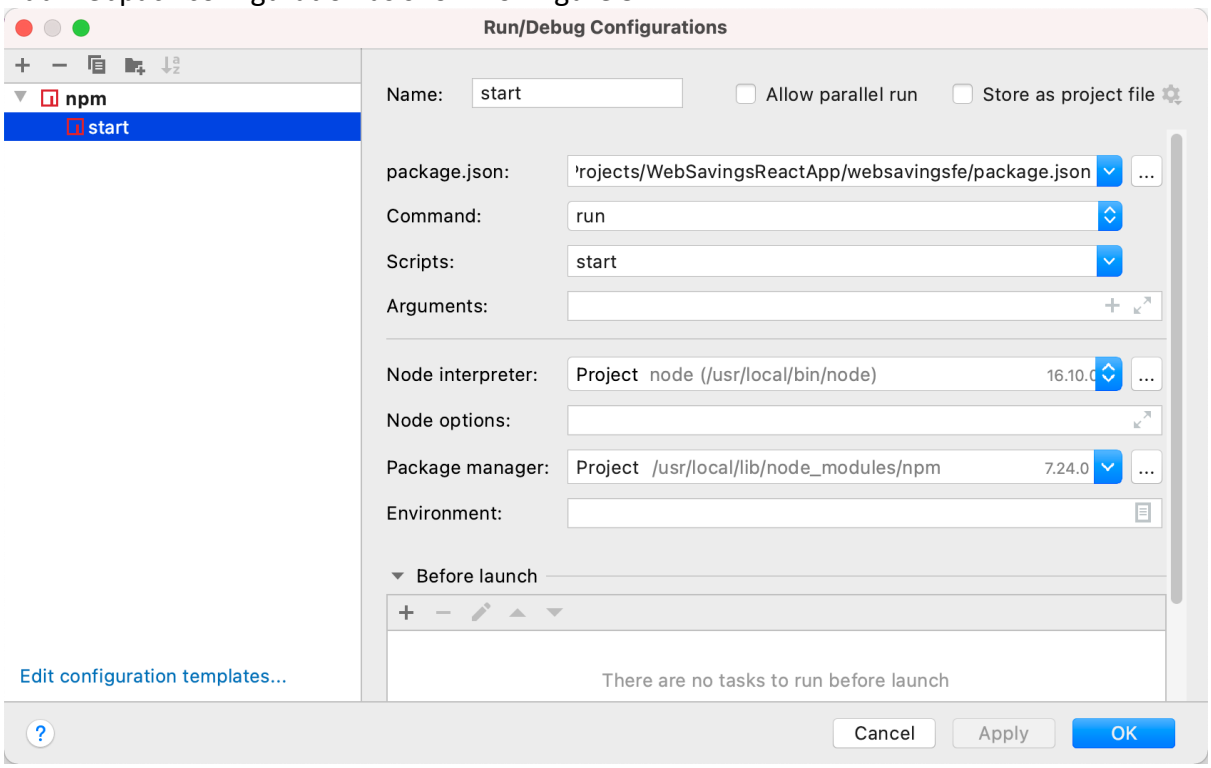

Figure 52 Local run configuration

Prettier setup

In order to be able to automatically reformat files, install the prettier plugin for Intellij and add the configuration shown on figure 53.

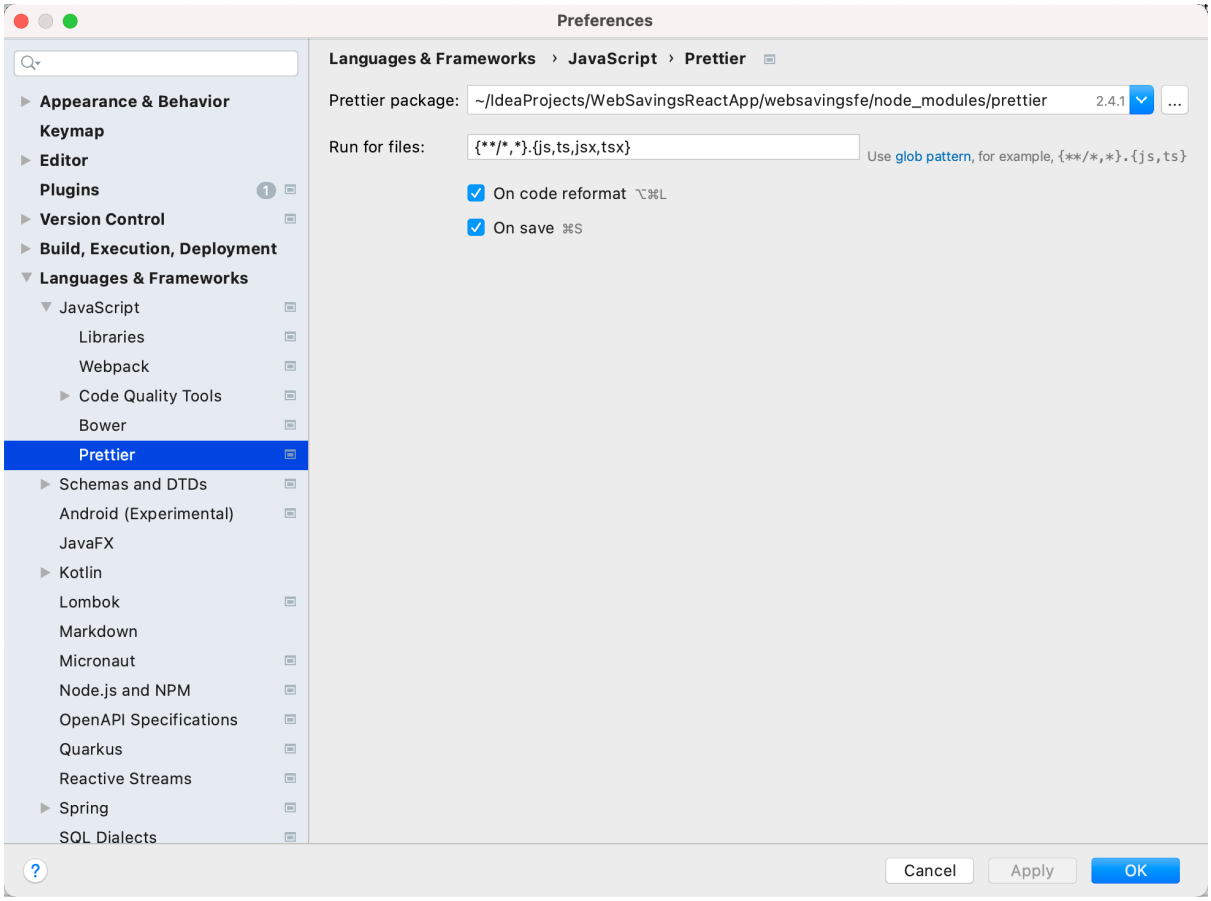

Figure 53 Prettier configuration

Checking the local H2 database

The data from the in – memory H2 database is available on: http://localhost:8080/app/websavings/h2-console. This URL shows UI for login into the database. For development purposes, the credentials are admin / admin and after entering the details for the schema, we can explore the database, as it can be seen on figure 54.

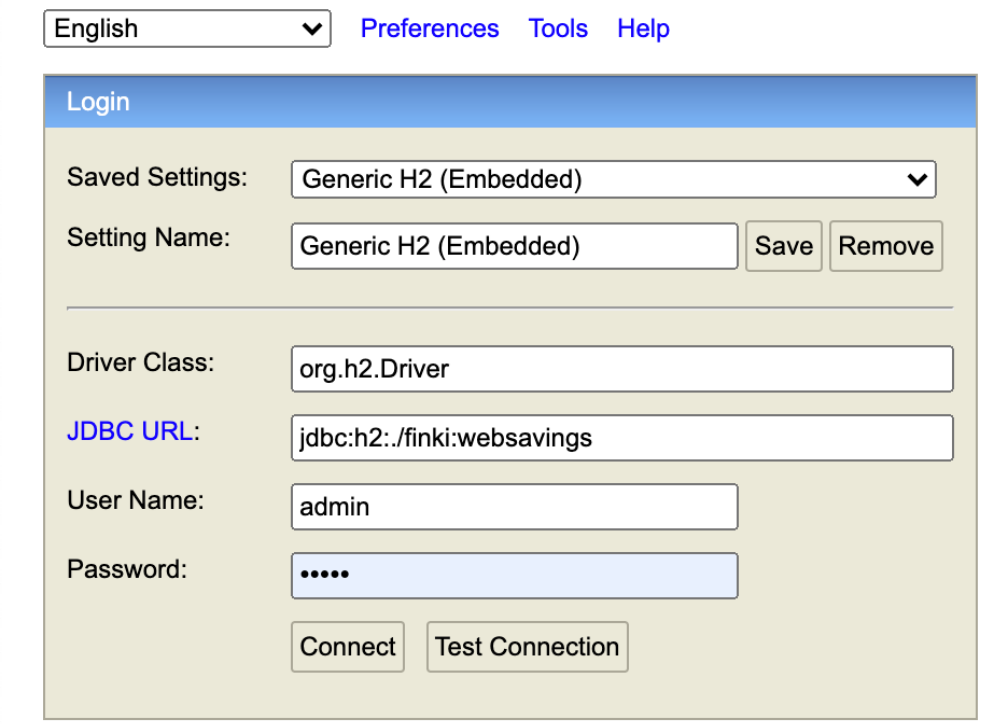

Figure 54 Login into the h2 local db

Figure 55 shows how the UI for the database administration looks, and where we can execute the queries.

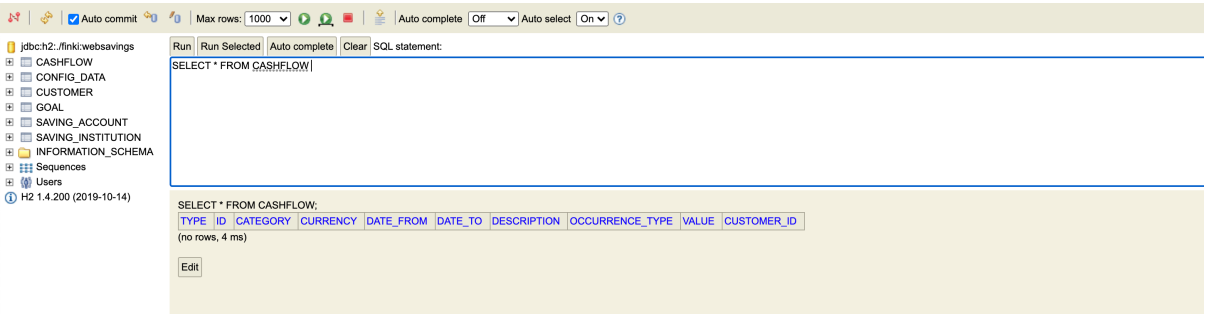

Figure 55 H2 db UI

IX API generation and making changes

This section contains details of the API generation and making changes to the API definitions.

All API definitions are added to the web-savings.v1.yaml. The build is using maven plugins to generate the API from the yaml file as shown on figure 56.

```
        </plugin>
        <plugin>
            <groupId>io.swagger.codegen.v3</groupId>
            <artifactId>swagger-codegen-maven-plugin</artifactId>
            <version>${codegen.version}</version>
            <executions>
                <execution>
                    <id>generate-web-savings-api</id>
                    <goals>
                        <goal>generate</goal>
                    </goals>
                    <phase>generate-sources</phase>
                    <configuration>
                        <inputSpec>${project.build.directory}/generated-sources/api/web-savings.v1.yaml</inputSpec>
                        <language>spring</language>
                        <addCompileSourceRoot>false</addCompileSourceRoot>
                        <apiPackage>com.finki.websavings.service</apiPackage>
                        <modelPackage>com.finki.websavings.model</modelPackage>
                        <configOptions>
                            <interfaceOnly>true</interfaceOnly>
                            <java8>true</java8>
                            <dateLibrary>java8</dateLibrary>
                            <library>spring-boot</library>
                            <useBeanValidation>true</useBeanValidation>
                            <useTags>true</useTags>
                        </configOptions>
                        <generateModels>true</generateModels>
                        <generateApis>true</generateApis>
                        <generateApiTests>false</generateApiTests>
                        <generateApiDocumentation>false</generateApiDocumentation>
                        <generateModelTests>false</generateModelTests>
                        <generateModelDocumentation>false</generateModelDocumentation>
                        <generateSupportingFiles>false</generateSupportingFiles>
                    </configuration>
                </execution>
            </executions>
        </plugin>
        <plugin>
```

Figure 56 Maven plugin for code generation

After making changes in the yaml, running clean install will rebuild the APi once again.

X Production deployment

This part will be defined after the complete flow for the prod deployment is implemented. It should also contain a detailed explanation for the CI / CD process.

Team members

Role	Name	Contact
Lead developer	Marija Nikolova	Marijanikolova93@hotmail.com
Project Manager	John Smith	jonhsmith@smith.com
Developer	John Doe	johndoe@doe.com
Business analyst	Angelina Jolie	jolie@an.com