PROJECT Design Documentation

Team Information

- Team name: 6f
- Team members
  - Kelly Feke
  - Jack Sebben
  - Oliver Vinneras
  - Kerri Walsh

Executive Summary

This project involves a backend and frontend implementation of a flag E-store, where users can purchase and customize flags and the owner can edit their inventory. The backend, titled Flags_API, uses model, controller, and persistence tier structure to implement the workings on the E-store. In these tiers are classes that implement the inventory and deal with flag objects. The entire backend is written in Java. The frontend, titled Tour-of-Flags, utilizes Angular and Maven to create the connection to the backend and implement the site interface. It utilizes component and service structure by Maven. The frontend is written in Typescript, HTML, and Cascading Style Sheets.

Purpose

> *Provide a very brief statement about the project and the most important user group and user goals.*

Glossary and Acronyms

> *Provide a table of terms and acronyms.*

Term	Definition
TS	Typescript
CSS	Cascading Style Sheets

Requirements

This section describes the features of the application.

> *In this section you do not need to be exhaustive and list every story. Focus on top-level features from the Vision document and maybe Epics and critical Stories.*

Definition of MVP

> *Provide a simple description of the Minimum Viable Product.*

MVP Features

> *Provide a list of top-level Epics and/or Stories of the MVP.*

Roadmap of Enhancements

> *Provide a list of top-level features in the order you plan to consider them.*

Application Domain

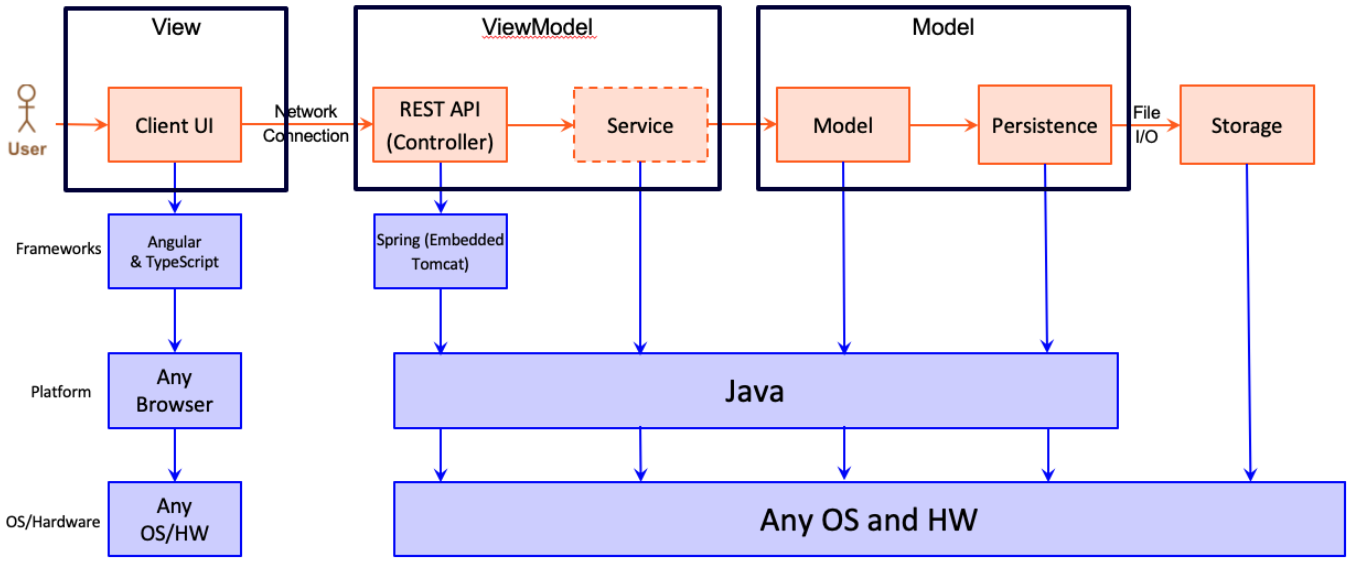
The domain model starts with the user in the Account class. The user interacts with the server and can login as either the admin of the E-store or a customer. The server stores the products in the inventory of the E-store, which in our project are flags. The admin can edit this inventory, adding or removing products as desired. The customer can search through the products in the inventory. The customer also has a shopping cart that they can add products to or remove products from.

Architecture and Design

This section describes the application architecture.

Summary

The following Tiers/Layers model shows a high-level view of the webapp's architecture.



The e-store web application, is built using the Model–View–ViewModel (MVVM) architecture pattern.

The Model stores the application data objects including any functionality to provide persistance.

The View is the client-side SPA built with Angular utilizing HTML, CSS and TypeScript. The ViewModel provides RESTful APIs to the client (View) as well as any logic required to manipulate the data objects from the Model.

Both the ViewModel and Model are built using Java and Spring Framework. Details of the components within these tiers are supplied below.

Overview of User Interface

This section describes the web interface flow; this is how the user views and interacts with the e-store application.

> *Provide a summary of the application's user interface. Describe, from the user's perspective, the flow of the pages in the web application.*

View Tier

> *Provide a summary of the View Tier UI of your architecture. Describe the types of components in the tier and describe their responsibilities. This should be a narrative description, i.e. it has a flow or "story line" that the reader can follow.*

> *You must also provide sequence diagrams as is relevant to a particular aspects of the design that you are describing. For example, in e-store you might create a sequence diagram of a customer searching for an item and adding to their cart. Be sure to include an relevant HTTP reuqests from the client-side to the server-side to help illustrate the end-to-end flow.*

ViewModel Tier

> *Provide a summary of this tier of your architecture. This section will follow the same instructions that are given for the View Tier above.*

> *At appropriate places as part of this narrative provide one or more static models (UML class diagrams) with some details such as critical attributes and methods.*

Model Tier

> *Provide a summary of this tier of your architecture. This section will follow the same instructions that are given for the View Tier above.*

> *At appropriate places as part of this narrative provide one or more static models (UML class diagrams) with some details such as critical attributes and methods.*

Static Code Analysis/Design Improvements

> *Discuss design improvements that you would make if the project were to continue. These improvement should be based on your direct analysis of where there are problems in the code base which could be addressed with design changes, and describe those suggested design improvements.*

> *With the results from the Static Code Analysis exercise, discuss the resulting issues/metrics measurements along with your analysis and recommendations for further improvements. Where relevant, include screenshots from the tool and/or corresponding source code that was flagged.*

Testing

> *This section will provide information about the testing performed and the results of the testing.*

Acceptance Testing

> *Report on the number of user stories that have passed all their acceptance criteria tests, the number that have some acceptance criteria tests failing, and the number of user stories that have not had any testing yet. Highlight the issues found during acceptance testing and if there are any concerns.*

Unit Testing and Code Coverage

> *Discuss your unit testing strategy. Report on the code coverage achieved from unit testing of the code base. Discuss the team's coverage targets, why you selected those values, and how well your code coverage met your targets. If there are any anomalies, discuss those.*