Architecture Design The First Order

The First Order

28 apr 2016

1 Introduction

1.1 Purpose

This document describes the architectural software design of the system by providing different architectural views. It's purpose is to explain the architectural decisions made on the system.

1.2 Design goals

While designing the system, we've maintained the following design goals:

1.2.1 Component independence

To maintain modularity and extensiblity, it's important for components for be independent of each other. This should allow us to easily replace, change or extend components in the system. Several design principles will be applied for this.
We will apply the model-view-controller pattern to decouple the representation, logic and the models.

1.2.2 Code quality

To keep our code maintainable, code quality is an important design aspect. A new member should be able to quickly understand what the code does. This means all methods should be commented with Java-doc. Testing is also an important aspect. Not only does it verify the workings of a class, it also provides a form of documentation.
To enforce this, we use several static code analysis tools. These have to pass before any code may be merged with the main branches.

1.2.3 Cross-platform compatibility

The application should be able to run not only on desktops, but on tablets as well. The UI should adjust itself to provide a smooth experience for both desktop and tablet users.

1.2.4 Scalability

The system should not only work for small projects, but also for larger projects. The performance should be consistent, which means importing a large project shouldn't slow the application too much.

1.2.5 Reliability

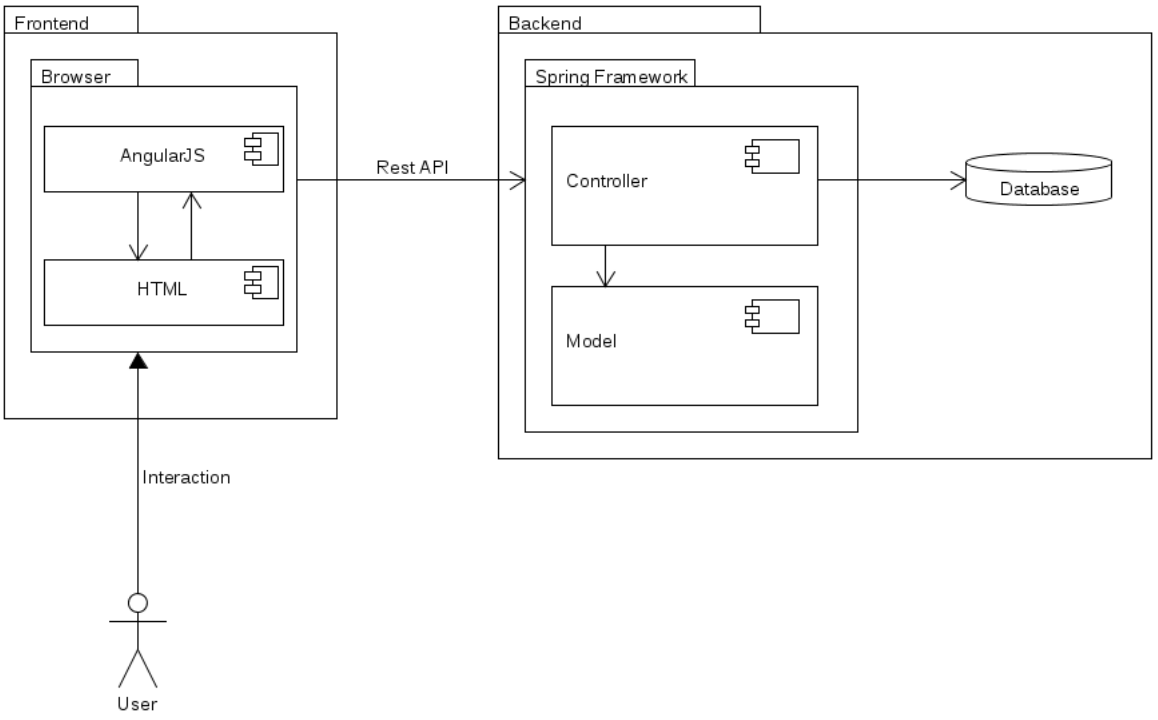
The system should not crash or produce unknown errors. If an exception is thrown, it should either be handled transparently or, if it is fixable by the user, shown to the user.

2 Software architecture

2.1 Programming languages and frameworks

The application is written in Java. To allow cross-platform compatibility, we chose to make this a web application.
Out of all the Java web frameworks, Spring Boot seemed the best choice. Spring is a dependency injection framework at first, but has been extended with support for numerous features. It has it's own model-view-controller framework, support for object-relational mapping frameworks and more.
Spring boot is basically a pre-configured suite, built on the Spring framework. It also provides the convention-over-configuration paradigm for Spring, making it fairly simple to setup a Spring web application without too much configurations.
For the frontend, we use AngularJS. This provides an easy way to configure the web interface.

2.2 Subsystem decomposition



2.3 Hardware/software mapping

The Spring backend is a server which is decoupled from the view. This means the UI can not only be viewed on the same machine as the backend, but also on a separate machine. The UI is accessed through the browser, which means other devices could also access it, such as tablets.

2.4 Persistent data management

The Spring framework has support for relational databases. In deployment, we will most likely use a MySQL database, but for development we currently use an in-memory database. We use object-relational mapping to store Java object in a relational database.

2.5 Concurrency

Multiple users might be using the application simultaneously. To accommodate for this, we designed the application to be RESTful.