High Level Design

Agile Express

Mert Koca

1. Introduction

1.1. Scope

This High Level Design application describes the goal and the design of the Agile Express application built for the JCP Internship programme. Document gives detailed information about the design choices made during the development, application architecture, data flow and database design.

1.2. Purpose of The Application

Agile Express is a lightweight Agile tracking application. The application provides basic necessities for agile management such as product board and tasks & sprints for project planning

1.3. Definitions

- JCP - Java Challengers Programme, An internship program where the participant expected the design and built the given project.
- Agile - A methodology used in software development.
- RESTful API - An architectural style for an application program interface
- Endpoint - An address which services a set of REST resources.
- LDAP - Lightweight Directory Access Protocol
- NoSQL - A database that models data with means other than the tabular relations used in relational databases.
- JSON - Javascript Object Notation, Common data transfer format

2. General Overview

2.1. Product Goal

Product goal is to provide simple agile functionality with an intuitive and easy to use user interface. Product has the following features;

- Application has roles with different kinds of access levels. Each action is restricted to some role. A user can not access information or do something without the necessary access level.
- Create, delete and edit projects,tasks or sprints.
- View and change task status in the project board.
- Search based on project,user,task or sprint content.

2.2. Product Overview

The product uses a client-server model where the server is a Spring Boot application and the client is a react application. Server side is a RESTful API that the client interacts with. Each event in the client side corresponds to a HTTP request to a server, afterwards server side logic processes this request and carries out necessary tasks. Apart from sign up and sign in endpoint, every endpoint requires authentication. If the user or the client which sent the request is authenticated then server side check for authorization because each function requires an authentication level. After these steps the server processes the incoming request and return response that may contain data and a HTTP Status code that reflects the current situation. If authorization fails, the server either returns an HTTP status code of unauthorized or processes the request based on the user access limitations. For authentication server side uses embedded LDAP authentication. To store data, the application utilizes MongoDB which is a NoSQL solution.

2.3. Project Configuration

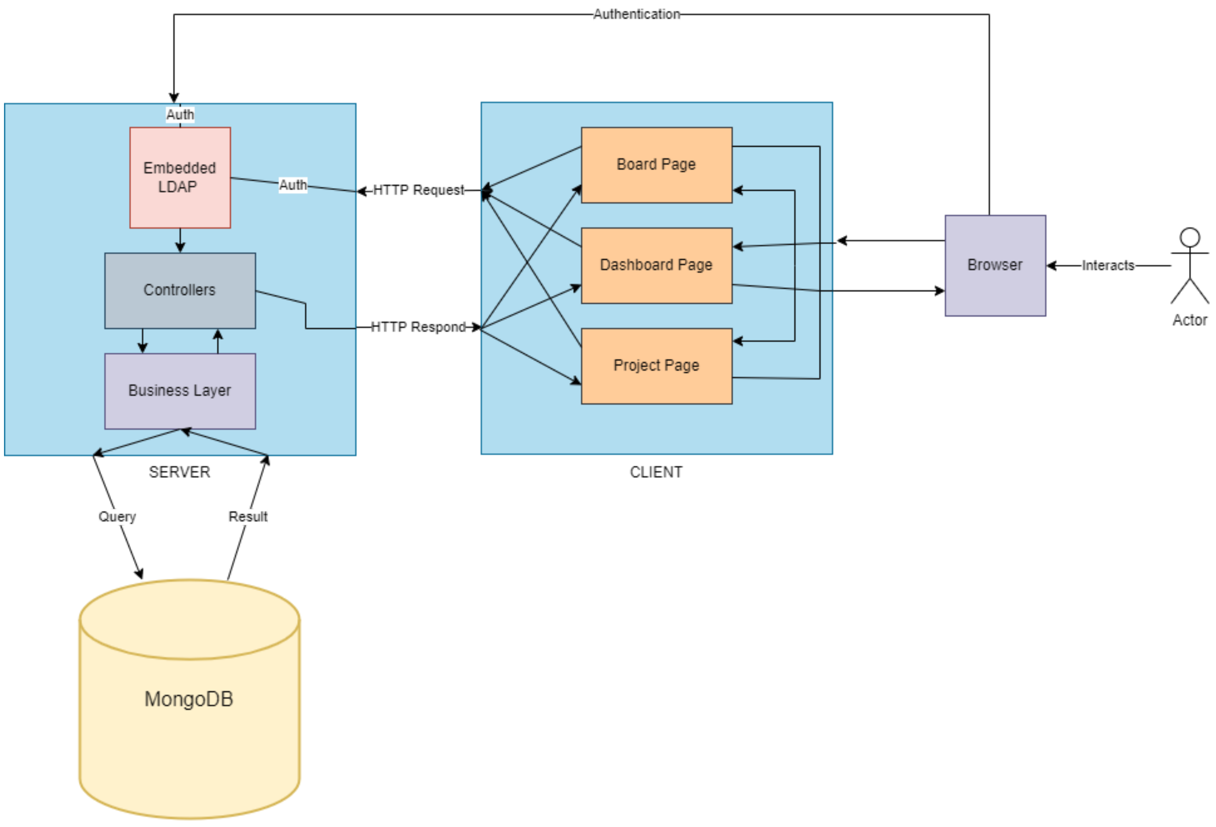
The server side project uses Spring Boot. To make spring work with React application uses frontend-maven-plugin and maven-resources-plugin. frontend-maven-plugin builds the React application and then maven plugin moves that build file to our target file where our executables are. This way if a controller in the server project returns an “index.html” it automatically returns the entry point of the React application.

3. Architecture

3.1. Main Design Features

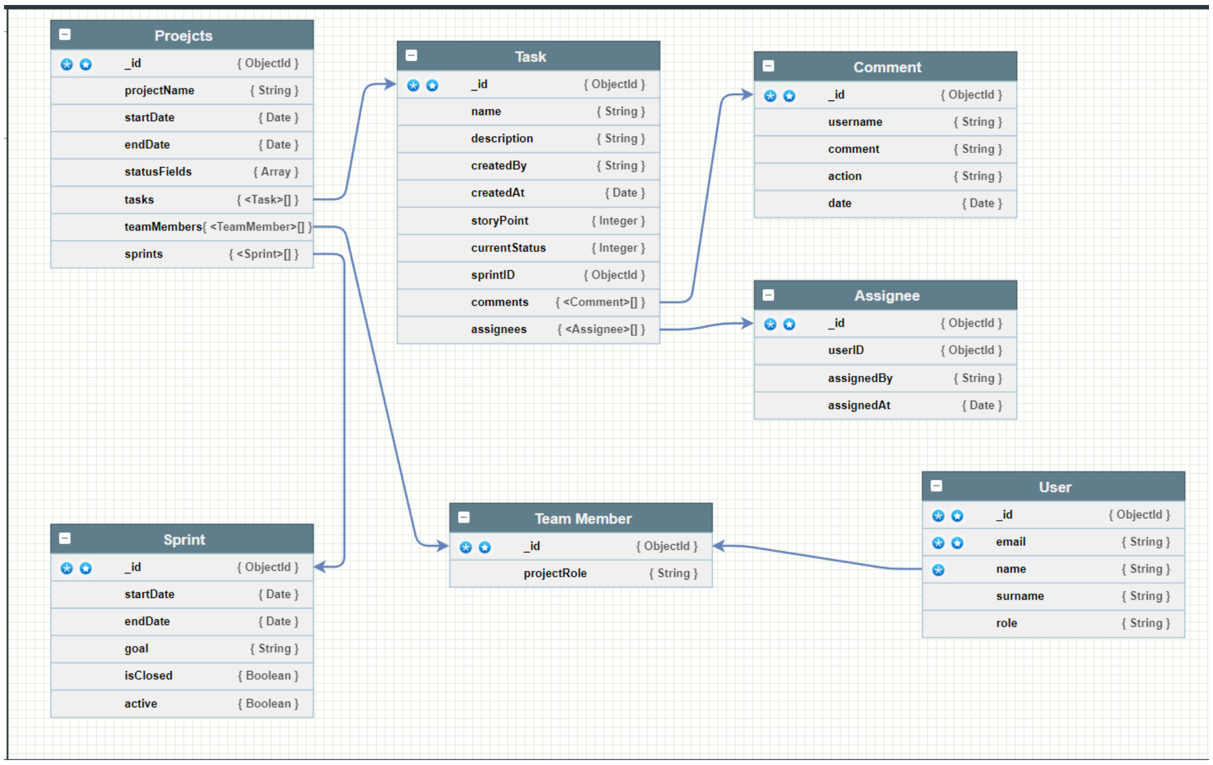
Main design decisions for this project were made in 3 parts; application architecture design,user interface and database design. Explanations will be supported with diagrams and screenshots.

3.2. Application Architecture



Application consists of a server and client project. Server project is a Restful API that client interacts through a browser by the user. To secure the application server project uses embedded LDAP. Any request to any endpoint(except for sign up and sign in requests) checks for authentication. If authentication fails, the user will be redirected to the login page to authenticate. If the request is authenticated then it will reach the controllers to the appropriate endpoint handler and will be processed. During this processing the authorization level of the user will matter. Some functions are allowed to certain users and these handlers will return 401(Unauthorized). Some functions will look at the authorization and depending on it will return a different value. The REST API will accept JSON and return JSON values except for IndexController which returns the index.html. This index.html is the entry point for the client project which is a seperate React application. On the client side React Router will return the specified page. The client side has 3 pages which are a React Component. These pages are; dashboard, project detail and board pages. Client side provides a minimal interface that the user interacts in order to communicate with the backend.

3.3. Database Design



3.3.1. Collections

As seen in the diagram database design mainly consists of 2 collections, projects and users. Users collection holds the user info such as email, or access level whilst project collection holds all the other information.Project collection has the following sub documents;

- Tasks
- Team Members
- Sprints

Task document holds all the information about a task and has the comment and assignee sub documents. Sprint sub document hold the relevant information about a sprint. And team members hold the current member of a project and their current project role. However this current project role is redundant because the application checks their user roles for their access level. But this field can be used if the need to promote or demote a user in a project arises. Holding all the relevant information about a project in a single collection reduces complexity and removes the joins that would be needed if the database was a relational one.

3.3.2. Indexes

Database current contains only 2 kinds of indexes. Indexes to ensure a field is unique and to use in search. All _id fields of a document are unique by default and additionally the username and email fields of the users collection are unique indexed.

Additionally the projectName field in the projects collection, name and the description field in the task sub-document and lastly the goal field of the sprint sub-document are text indexed. Furthermore name, surname, username and email fields in the users collection are also text indexed. Text index allows Mongo to search for values in these specific fields with the given query.

3.4. User Interfaces

The application will have 3 page views that fulfills the requirements in the project definition document.

Dashboard Page: After a successful login the user is directed to this page. Main goal of this page is to show projects and their summary so the user can look at them in a breeze. If the user has the role of system admin, then the user can see all the projects. Anything lower and the user will only see the projects that he/she is a part of. System admins and project managers will be able to create a new project on this screen. From this page each project will have links to go to the board and detail pages of that project. But the detail page is allowed only for the users with the access level or team lead or higher. Team members are only able to see go to board link. This is the page where search functionality is located.

Detail Page: This page displays every information available about a project and allows users to change these properties. This also the screen where a new sprint or a task can be added or edited. To edit the properties of the project, the minimum required access level is project manager. But team lead and above can add sprints or tasks and edit tasks. Team members are able to add comments to a task.

Board Page: In this page users can interact with the product board. All tasks are moveable between status fields for anyone with the access level higher than team member. Team members can only move the tasks that they are assigned to.

3.5. Files

The project has an LDIF file that the embedded LDAP security uses the search for username and password hashes to authenticate.

3.6. Error Handling

The application controllers return the appropriate HTTP status codes to the client agent. Also some controllers have additional error messages embedded in them specifying what exactly went wrong.

3.7. Cache

Since each function requires an access level check the server side application caches the user id and access level. This way unnecessary calls to the database would be prevented.