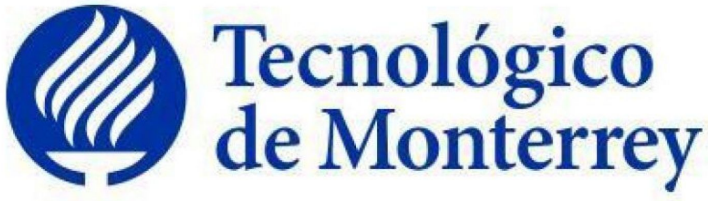


Technical Manual

Salvador Orozco Villalever - A07104218
Juan Andrés Reynoso Mazoy
Ricardo Rodiles Legaspi
Jesús Alejandro Coles Moguel - A01171039
Raúl Barranco Cortés
A01090517

18/10/2018

TECHNICAL MANUAL
TABLE OF CONTENTS

Page #

1 DOCUMENT OVERVIEW

This document intends to explain the technical aspects of the project, which include the proposed system architecture, the components, their interaction, and the way they all integrate into the system.

First there will be an explanation of the architecture design, which will be followed by a more in-depth look at the components themselves. Then we will explore the database design through a series of diagrams, and finally we will explore the information architecture.

2.0 SYSTEM ARCHITECTURE

2.1 Overview

The requirements for the software affected our decision in favor of choosing to design a web-based platform to allow for experiments to be conducted. Among the reasons for this is the improved cross-platform support, as the web app can be accessed from different operating systems.

The web site is designed as a single-page application by using Angular, since this open-source development platform allows for large flexibility and fast development, while ensuring a seamless, modern web experience and following an MVC pattern.

The frontend is thus written in TypeScript. It also comes with a built-in testing framework, Jasmine, as well as a test runner, Karma; this makes testing easy and ensures software quality of the app all along the development process.

For the backend, we use a REST API which is developed separately from the frontend, using Node.js as a server-side platform and Sequelize as an Object-Relational Mapping (ORM) tool. Sequelize allows for easy access to the database as SQL queries are abstracted behind an easy-to-use API. Communication between the client and the server is done using HTTP requests and responses as JSON objects.

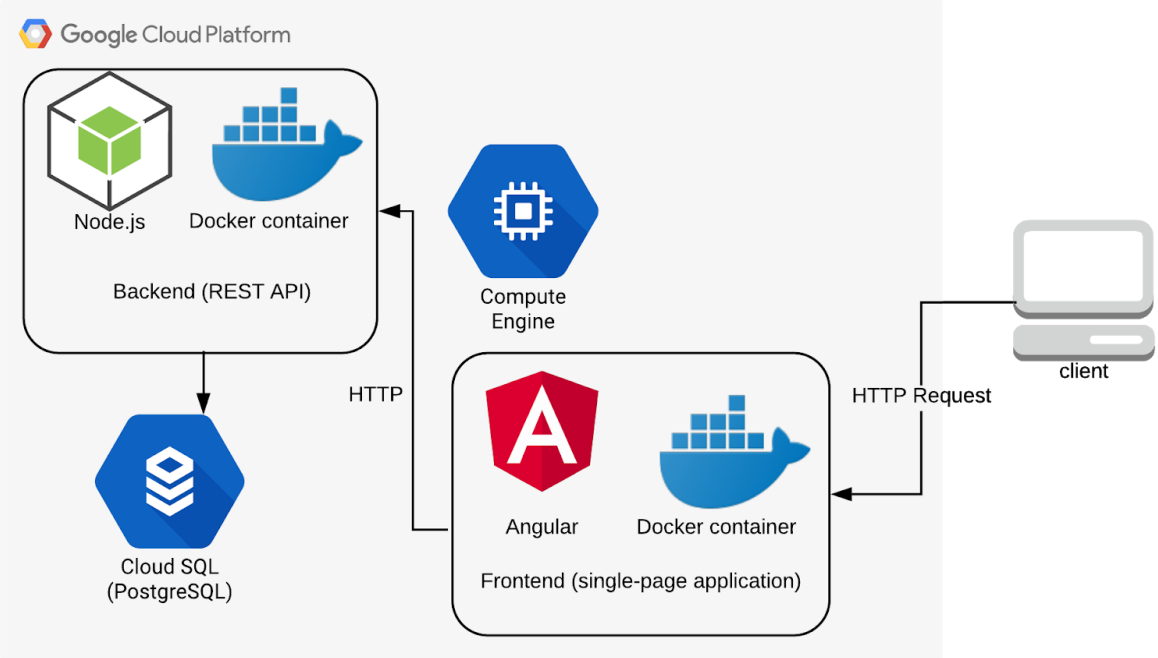
In order to conduct tests for the backend, we decided to use Mocha as a JavaScript test framework. Tests are automated and run using the Travis CI continuous integration service linked to the project's GitHub repository.

The data is stored in a PostgreSQL database. We chose this DBMS because of its popularity and robustness. Also, it has an efficient and easy to learn and apply ORM, which is Sequelize.

In order to host our application for public demos, we are using Google Cloud Platform, where the frontend and backend are running in Docker containers, while the database is running in a dedicated Cloud SQL instance.

2.2 Architecture Design

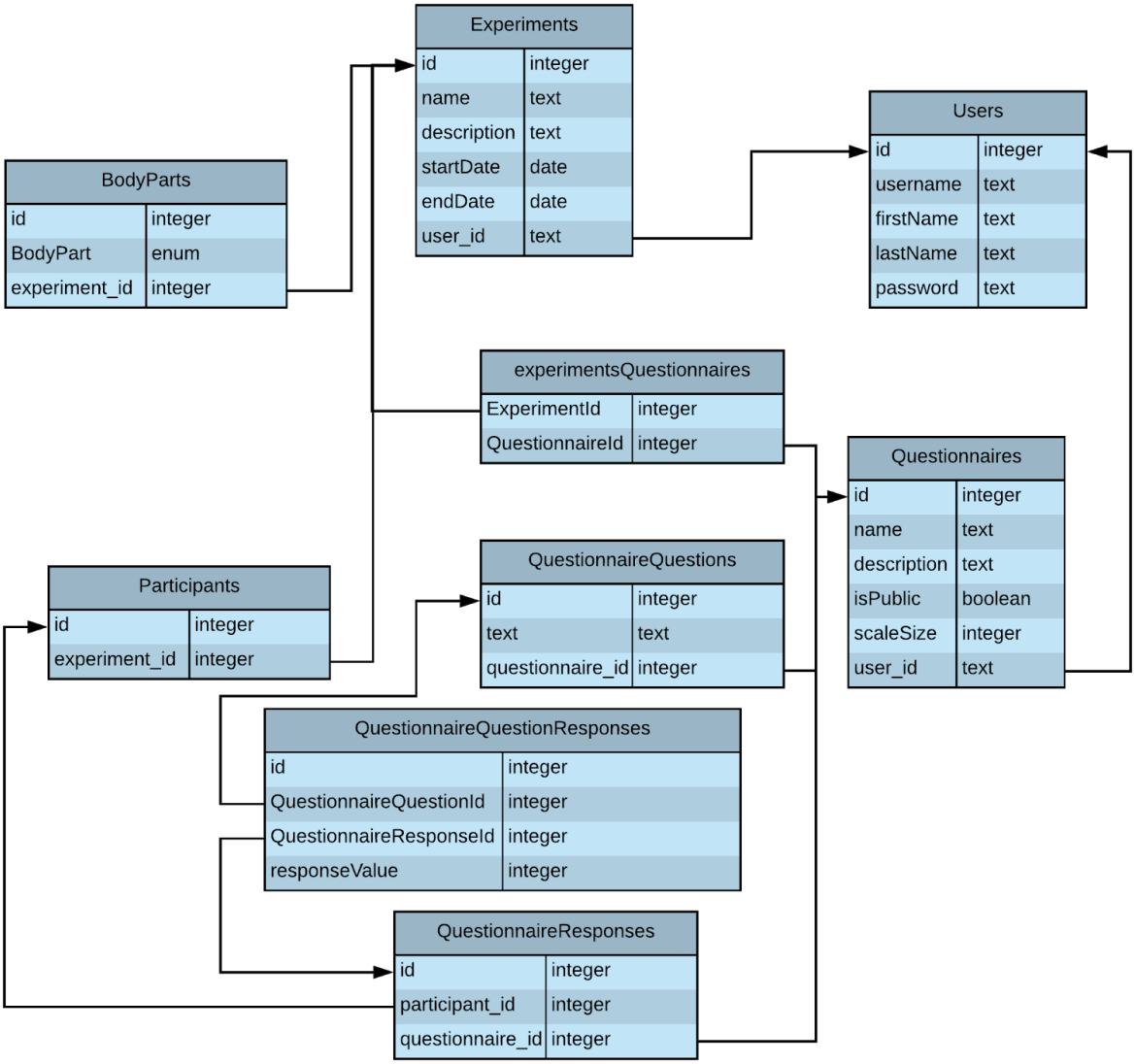
The web architecture as it stands for the demo is the following:



The current design can be adapted and modified by the company or institution that adapts this design, as it only requires a container or machine to run the backend, another one for the frontend, and a PostgreSQL server.

3.0 DATABASE DESIGN

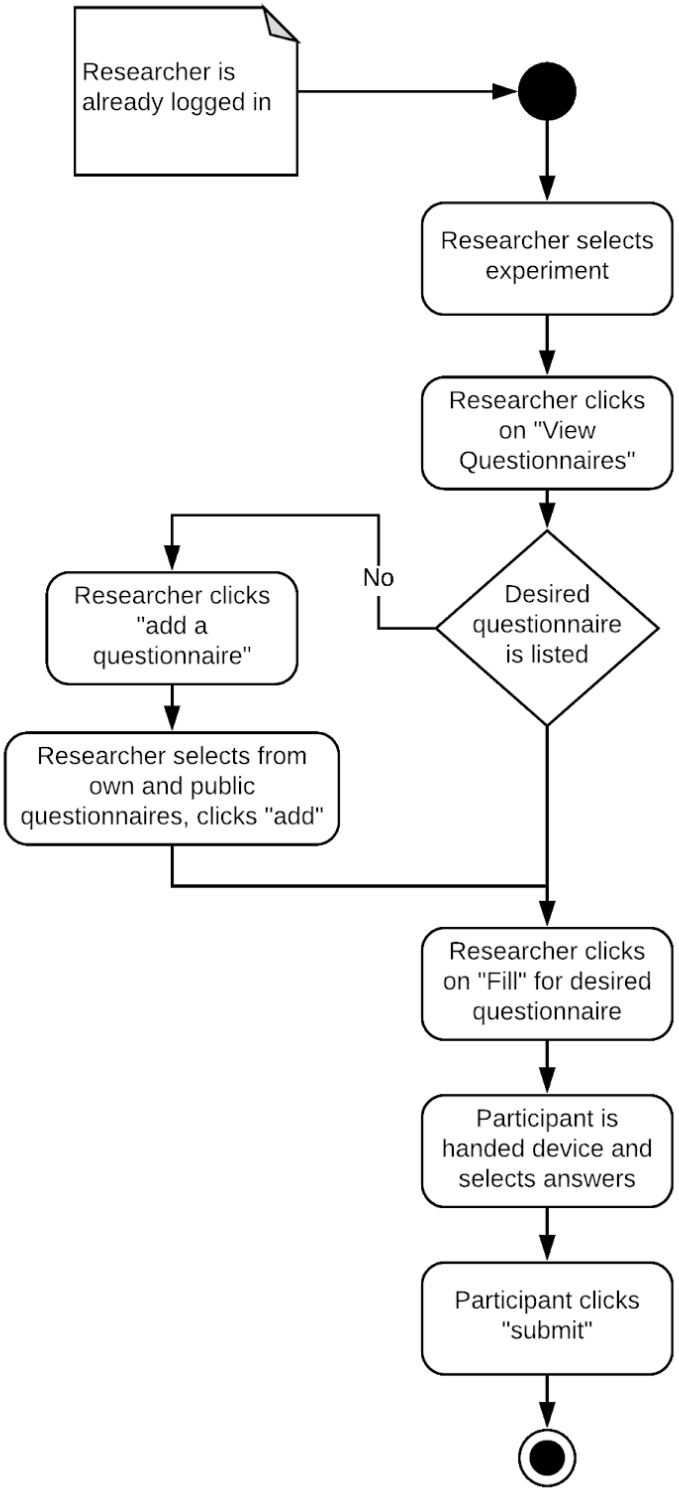
The following is a relational diagram for the database, showing all of the tables currently in use. The most important ones are Users, which models the end user (e.g. a researcher or academic) using the software, and Experiments, which are the usability experiments themselves.



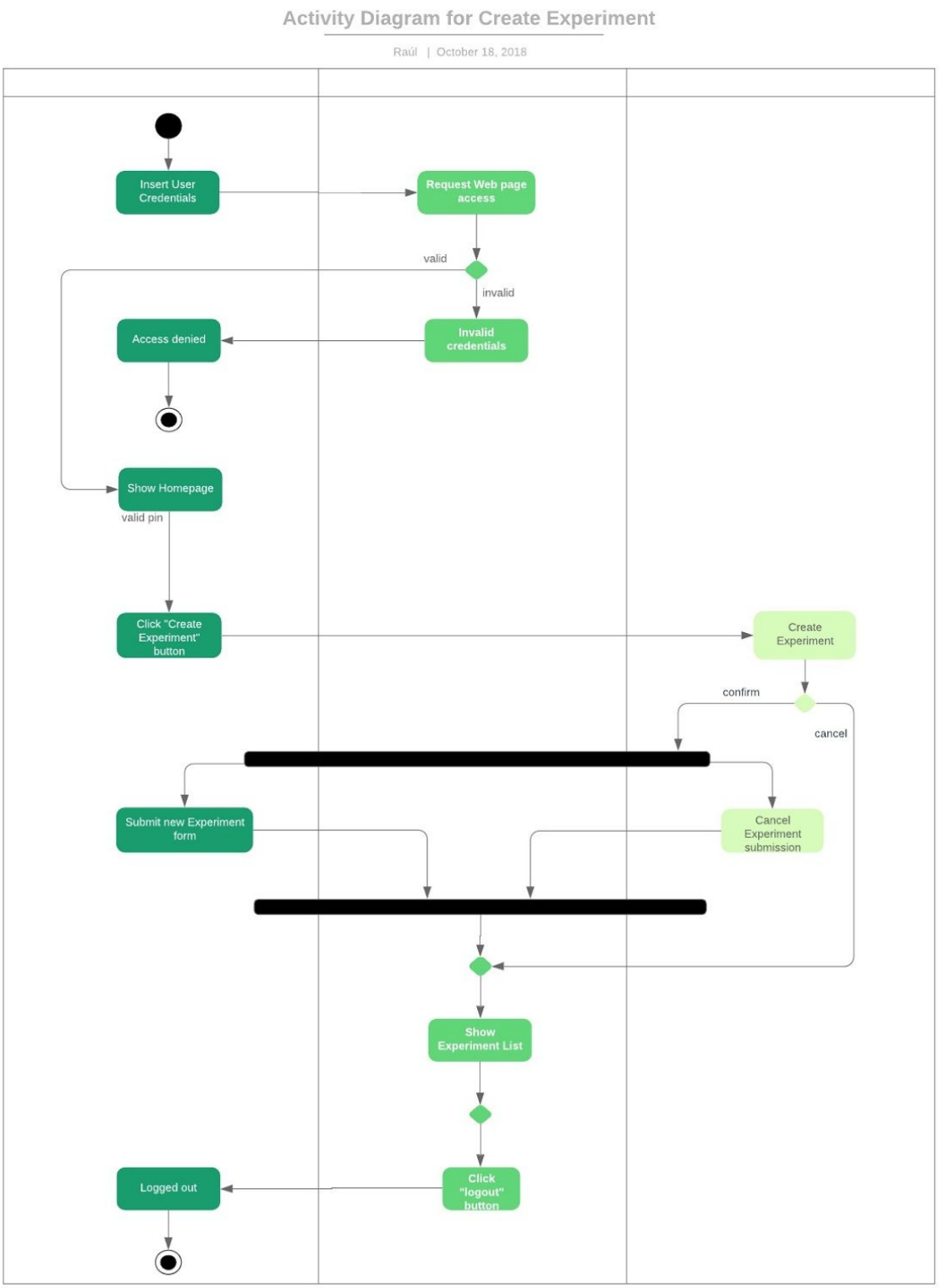
Note: in this diagram the default Sequelize timestamp attributes (createdAt, updatedAt) are omitted for brevity.

4.0 UML DIAGRAMS

The process for filling out a usability questionnaire is as follows:



The process for creating a new experiment is as follows:



The CRUD sequence of an Experiment is as follows: