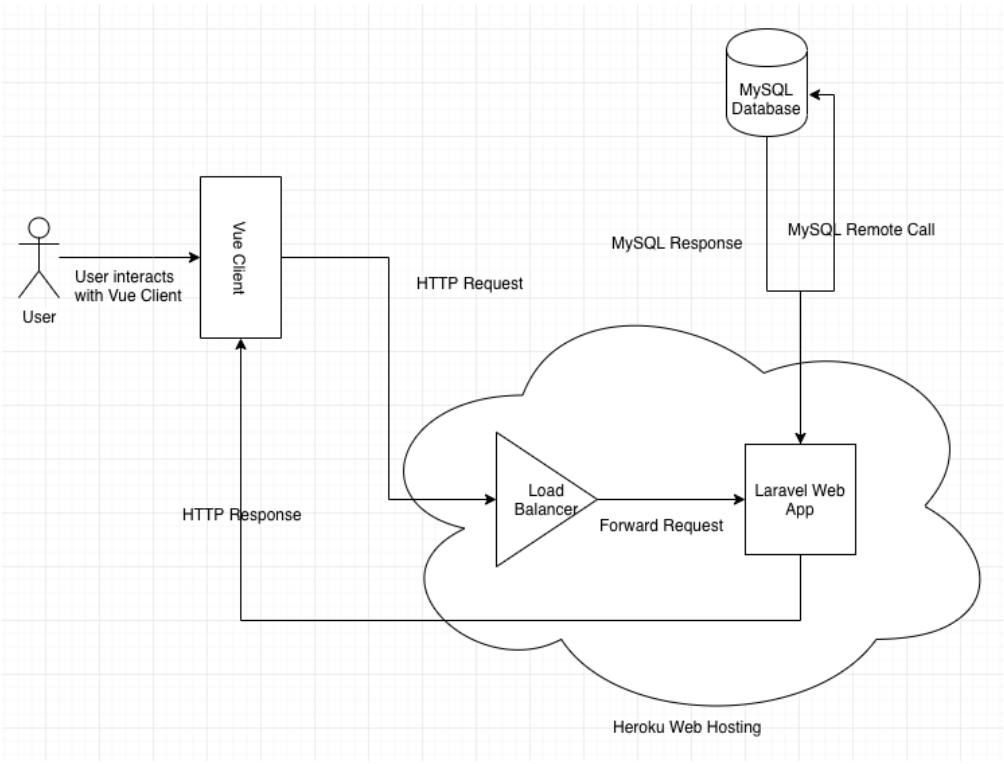
Software Documentation

This document is mainly for the working developer to offer him a high level overview of the software stack we are running in order to provide him with the technical foundation to begin working on this app.

Overview

1. Application Architecture
2. Components
3. How to get started working on project

Architectural Overview



As you can see we have kept it simple because the project was a standard web application at a relatively small scale (hundreds of users). The normal work flow would entail a user interacting with the Vue client. The Vue client will initiate HTTP requests targeting our server URL https://sleepdiaryapp.herokuapp.com/api in order to give the user the appropriate displays and feedback. The requests first go through Heroku's load balancer which will forward the requests onto our Laravel web app. We will do some processing there and depending on the request, most likely make a database call. After that Laravel web app has finished its processing it will have a final output which will be sent back to the Vue Client as JSON. The Vue Client will take this JSON and display it back in a user friendly way for the user e.g. A bar graph.

Components

The following are all the important components that exist in our web application.

Frontend

In modern web development this usually refers to what the user's browser displays. Browsers take in HTML, CSS and JS and generate a display back to the user to interact with. This display is what a user would call a website. However with the advent of Angular which inspired numerous other frontend frameworks to be developed, namely React and Vue the way content is delivered is a little more complicated. Instead of classic way where the server would generate the HTML, CSS, JS the frontend framework would do this instead. This is a fundamental shift in the way we deliver and write web applications because this moves a lot of the logic onto the client side.

This idea has taken over many large production websites now. Youtube uses Polymer, Facebook uses React, Airbnb uses React, Twitter uses React, Baidu uses Vue, Paypal uses Angular. Therefore it is important to understand why we would use a Frontend framework. Firstly it does make the development process more complicated **initially** with more setup, more libraries and frameworks to learn. But I have found that the code you write is much more ***robust*** because the framework forces you to structure your code and files in a readable and coherent manner. As a result there are less bugs, maintainability increases and the code base can scale as you add and remove team members.

Vue

This is the foundation of the frontend client which the user interacts with. The philosophy of Vue is to provide a simple, expressive and elegant way to write frontend web applications. The code located here: https://github.com/noobling/prof-comp/tree/master/website/client is written for the Vue framework

Nuxt

Nuxt is another framework which is built on top of Vue. The main purpose of Nuxt is to help write *universal* applications. The main appeal of universal applications is the ability to have server side rendering, this solves the issue of poor SEO from frontend frameworks. In our case I decided to use Nuxt because it makes writing Vue apps easier, a lot of the boiler plate code is done and it has a structure already laid out for me.

Vuetify

This is a styling library that provides web UI components designed following the material design philosophy by Google. I chose this because it makes the website look nice without a lot of coding and design required from me.

Backend

In web development backend refers to all the code that does not exist on the users machine but instead it is in the cloud. Normally this code will be the layer that is in between the user and database, reading, updating and deleting database records.

Laravel

This is a modern web framework written in PHP. The main purpose for using Laravel is to make our lives easier because it provides a lot of baseline code for us and also gives us a good foundation to write a production ready web application. For example as developers we don't want to write code to parse a HTTP request and we don't want to write code to connect to a MySQL database. Laravel handles all this critical but boring things so we can focus on business level details e.g. implementing the database schema.

MySQL Database

This is the most common implementation of a SQL database because it is free and open source. For our purposes it doesn't really matter what type of database we use, we just need one to store our data safely and securely. The main purpose of using a database is to add persistance to user data. In our case the most obvious data we need to store are sleep records.

Working on Project

The key files to note for development purposes are the `website` and the `client` folder within the `website` folder. As the name suggests the website folder is where the Laravel project is housed and the client folder is where the frontend client code exists.

Software you will need

- Local mysql database
- Composer and php 7.1
- Node and npm

How to run development server

- `cd website` Make sure you are in the correct directory
- `npm install` Install client side dependencies
- `npm run dev` for client side
- `composer install` Install server side dependencies
- Copy .env-example to .env and enter in your details.
- Create `.env` file using `.env.example` as a guide
- `php artisan jwt:secret`
- `php artisan serve` server side

How to deploy the app

This refers to launching the app for the world to see.

- Make sure dev server is not running
- `npm run build:prod`
- Move the generated files in dist to public folder in Laravel app
    - Delete `.nojekyll` file since it hinders the copying
    - Click replace all
- Go back to root directory
- `./deploy.sh`
- Go to heroku dashboard and deploy prod branch