Eventsync Server Documentation

Popescu Nicolae-Aurelian
Bogdan Ionescu

May 19, 2019

Contents

1 Introduction

1.1 Overview

Eventsync is an application that helps circles of friends from all over the world stay connected by merging together their calendars and helping them find the perfect time to hang out.

This document provides the Eventsync server documentation, including its scope, requirements, capabilities, arhitecture and many more.

1.2 Requirements

The Eventsync server provides all the necessary backend services for the Eventsync Android App, available for all Android phones by accessing the Google Play Store.

1.3 Capabilities

The server provides the following capabilities, which can be requested by the Eventsync app:

- registration of a new user
- login of an existent user
- user logout
- adding a new event

It also maintains several databases, which contain users information, events details and session persistence cookies, all of which are vastly explained in the next sections.

1.4 Software stack

The server is written in Node.js, making use of its excellent asynchronous support, superior speed and REST APIs. MongoDB is used for the NoSQL database instances and Docker for deployment operations and testing purposes.

2 Database configuration

2.1 Connection settings

Connection settings to the MongoDB instance, including the IP address, port and URL used are defined in the `configuration.js` source code file. It also defines the `SESSION_SECRET` constant, which is a custom cookie string used for user sessions.

2.2 Database instances

The first connection to the database and its settings are defined in the first lines of the `server.js` file. Two collections are initially created, named `users` and `events`.

The first one contains all the information known for every user, such as his username, password, name, and is indexed by username, which allows faster queries and ensures that no two users hold the same username, for login purposes.

The second one holds details for all user added events, including the start date, duration, participants and description. This collection ensures two indexes, after the event name and its end date.

3 Server arhitecture

3.1 Session establishment

A session to the server is established after the connection to the database in the `server.js` file, when a request is made to the server's IP. This session is maintained by the aforementioned `SESSION_SECRET` cookie, used for identifying the connection.

3.2 Routes

After the session is established, several routes are available to the user. These are defined in the `routes` folder in the source code, where is file represents a different route. These act as independent microservices, implying the fact that if one of them fails due to unknown reasons, the others will still be fully functional.

Each route requires different parameters to be set, such as certain headers or cookies, which can be found in the source code. The information contained in each request, such as the username and password of a user in case of a login attempt, is parsed and cross-checked with the records in the corresponding database instance. If the request is valid, a `200 HTTP Status Code` is sent back, along with any useful or requested information. On the opposite side, if the request contained either missing or erroneous information or parameters, `500 HTTP Status Code` is raised, often accompanied by certain fields which indicate why the request was considered invalid.

4 Nonfunctional requirements

4.1 Usability requirements

100% of users will be able to use the application without any knowledge of the backend services or arhitecture.

4.2 Performance requirements

Depending on the host platform which is used for the deployment of the server, application speed and response time may vary, but provided with a good internet connection, they will be unnoticeable.

4.3 Security requirements

All user passwords are encrypted with the help of several security protocols, thus a malicious attack on the server would be at most be able to obtain only the usernames, which presents no harm to the general public.

5 Testing

Several automated scripts were designed and deployed to send a large number of requests to the server, at different waiting times. Apart from a very insignificant delay in response time, no other features were affected.