Metropolia University of Applied Sciences

Internet of Things

Group Project

Web Interface for ABB Ventilation Controller
Technical Documentation

Authors:
Janine Paschek
Maya Hornschuh
Theresa Brankl
Simon Schädler

Submitted on: September 22, 2021

Contents

1 Introduction

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

2 Authentication and Authorization

Authentication and authorization are both very important for this interface. Only logged-in users are allowed to open the page. Also, not every logged-in user has permission to use all available features. Therefore, on each request a client sends, the server will try to authenticate the user before providing any information. In this project, basic HTTP authentication is used log in/log out users.

2.1 Front end authentication

When a user initially connects to the web interface, the server will return a 401 error code and ask for authentication. The browser will automatically show a form and ask the user to enter a username and a password. If the entered credentials are correct, the server will redirect to the landing page.

Every page of the interface provides a log-out button, which enables the user to manually log out. If a user logs out, the client will first send a basic get request to log out from the server. Then, the client sends an invalid authentication request and redirects to the logout page. Sending a separate logout request before the invalid authentication enables the server to tell apart the logout request from any other invalid authentication like logging in with incorrect credentials. This is important to keep track of the users that are currently logged in and can be used to log all login activities.

2.2 User database

The server uses one central SQLite database (*data/data.db*) to store all kinds of data. The *users* table contains information about each user in the system in the following format:

```
CREATE TABLE users(username TEXT, hash TEXT, timestamp INT, role TEXT);
```

Next to the username and the hashed password, the table contains a timestamp of the users' last login (in milliseconds since 01.01.1970 00:00:00 UTC) as well as a role. The timestamp can be used to distinguish new logins and navigation between different subpages (REFERENCE CHAPTER), but also for logging all login activities on the server (REFERENCE CHAPTER). The role indicates whether the user has admin privileges or not.

2.3 Routing

All routes that the server handles require the client to authenticate before serving any page or data. The only exception to this is the */logout* route, that returns the *views/logout.ejs* file to the client without authentication. All other routes call the function *auth_user(req, res, next, redirect)* (REFERENCE SECTION) and pass a string of the requested redirect as a parameter. The function will check the authentication parameters provided by the client. If the provided information is valid, the function will call the requested function and return information to the client.

2.4 User Authentication

If a route is called by a client, it will call the *auth_user(req, res, next, redirect)* function with information about the requested service. The function reads authorization parameters from the request body and checks if valid data was received. If so, the function generates a hash based on the provided username and password and compares it with the hashes stored in the database. If the username and password match the information in the database, the user is authenticated successfully. If a client was successfully authenticated, the function checks if the request was a new login or just an authenticated request for a page or service. To do so, the current time is being compared to the users' timestamp in the database. Three cases can be detected that way:

- If the timestamp equals zero, it either is still zero from its first initialization or has been reset during a logout procedure. Therefore, the client is performing a new login. Then, the login will get logged and the timestamp in the users' database entry will get updated.
- If the difference between the current time and the timestamp of the users' last login is greater than 30 minutes, the request will be interpreted as a new login. Then, the login request will get logged and the timestamp in the users' database entry will get updated.
- If the difference between the current time and the timestamp of the users' last login is less than 30 minutes, the request will be interpreted as a request for changing the page or fetching data. The timestamp will not get updated and the request will not get logged in the database.

After deciding, if the database needs to get updated because of a new login, the function switches depending on the passed *redirect* parameter. If a page is requested, the parameter will directly include the filename of the page that should be rendered. Otherwise, the corresponding function will get called to perform actions and return the requested data to the client.

2.5 User authorization

Users are allowed to use most of the features provided by the web interface. Still, some actions can get performed by authorized users only. For example, only admins are allowed to add a new user to the system or to see all users' login activity. As described in section (REFERENCE CHAPTER), the *users* database has an attribute that tells the role of each user. There are two roles, the *default* and the *admin* role. If the client requests a service that is available to admins only, or that returns different results depending on the users' role, the *auth_user()* function will pass the current users' role as an argument to the function, that performs the requested actions. Then, the server decides if the user is authorized or not. In general, the server returns one of the following status codes on requests that require specific privileges:

200	'OK'	The requested action was successfully executed
403	'Forbidden'	The user has no permission to execute the requested action
409	'Conflict'	The action could not be executed because of conflicting arguments

The server sends those status codes alongside the resulting data or message. Then, the client displays received data, takes the user to a different page, or displays an alert depending on the result.