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: October 21, 2021

Contents

1 Introduction

VentPro is a web interface for controlling an ABB ventilation controller. The interface displays all available information about the connected IoT device and enables the user to control the ventilation system using a website.

This technical documentation provides specific information about the implementation of both the front end and back end of the system. It does not include any descriptions of how to use the web interface itself. This information can be found in the user manual.

The system consists of an IoT device, a server, and a web interface. The IoT device controls the speed of a connected fan and measures the current air pressure regularly. The device is connected to the server which provides the web interface allowing users to set a specific pressure or fan speed. Also, the interface displays current and former sensor data received from the IoT device to the user.

1.1 Installation

This project is based on Node which needs to be installed to run the server. Please visit nodejs.org and follow the instructions to install node (LTS or latest version). The following node packages are required to run the server. The packages can be installed by running *"npm install <package>"*.

- express
- ejs
- body-parser
- sqlite3
- ws

The server also requires a running MQTT broker. To connect the server to a running broker, please open the */src/server.js* file and check the MQTT configuration section. Please enter the correct IP, port, and topics by adjusting the following parameters:

```
const mqtt_ip = "mqtt://localhost";
const mqtt_port = "1883";
const mqtt_topic_pub = "controller/settings";
const mqtt_topic_sub = "controller/status";
```

After installing the required packages and setting up the MQTT parameters, the server can be started by running *"node src/server.js"*. The server display status information in the terminal if everything started up correctly.

2 Back end implementation

The server provides users with all necessary information for using the web interface. It serves requested web pages and data in general but also establishes the connection to an IoT device and stores different kinds of data consistently. The server's functionality is implemented in a central JavaScript file, the *server.js*.

2.1 Global database

The server uses an SQLite database which is set up as a single global database to store all kinds of data. The database (*/src/data/data.db*) contains the following five tables to store data:

```
CREATE TABLE users(username TEXT, hash TEXT, timestamp INT, role TEXT);
CREATE TABLE log_users(timestamp INT, user TEXT);
CREATE TABLE pressure(timestamp INT, pressure INT);
CREATE TABLE fan_speed(timestamp INT, fan_speed INT);
CREATE TABLE target_values(id TEXT, value INT);
```

The *users* table contains the login information about all registered users while *log_users* keeps track of all login activities which is described separately in section 2.3. On the other hand, *pressure* and *fan_speed* are tables for data logging only. These tables store all sensor data received from the ventilation controller. The *target_values* table contains the current values of the target speed and target pressure requested by the user as well as the current mode of the system (see chapter 2.5 for more information).

2.2 Routing

The server uses Express to manage the routing. To prevent unauthenticated users from accessing data, every route refers to the *auth_user(req, res, next, redirect, arg_dyn = '')* function to authenticate users and to check their permissions (chapter 2.3.) The following example shows the routing for */control_panel*:

```
app.get('/control_panel', async (req, res, next) => {
   auth_user(req, res, next, 'control_panel');
});
```

In this example, the server will pass all necessary parameters for authenticating the user and returning a reponse as well as the information about which route was called. If the user is logged in, the server will render the */src/views/control_panel.ejs* file as a response and return it to the client.

2.3 Authentication and authorization

2.4 Settings functionality

2.4.1 Changing the password

2.4.2 Adding a new user

2.4.3 Login activity

2.5 Data transfer

2.5.1 MQTT

2.5.2 WebSocket

Data minimization

Live data

Warings

Sending commands

Storing identifiers

3 Front end implementation

3.1 Responsive header and navigation menu

3.2 Help page

3.3 Control panel

3.3.1 Activating and inactivating the panels

3.3.2 Switching the mode

3.3.3 Plotting the data

Fetching data of the selected interval

Receiving and displaying live data

Selecting different time intervals

3.3.4 Setting the target pressure and target fan speed

Fetching the current state of the system

Sending a new target pressure and fan speed

3.4 Settings page

3.4.1 Changing the current users' password

3.4.2 Adding a new user to the system

3.4.3 Displaying the login activity

3.5 Displaying warnings

3.6 Logging a user out