LABORÜBUNG PRÄSENTATION MOBILE SYSTEME UND APP-PROGRAMMIERUNG

Gruppe 1

Lucas Gomes, Kevin Schulz
Hochschule für Angewandte Wissenschaften Hamburg
20.01.2023

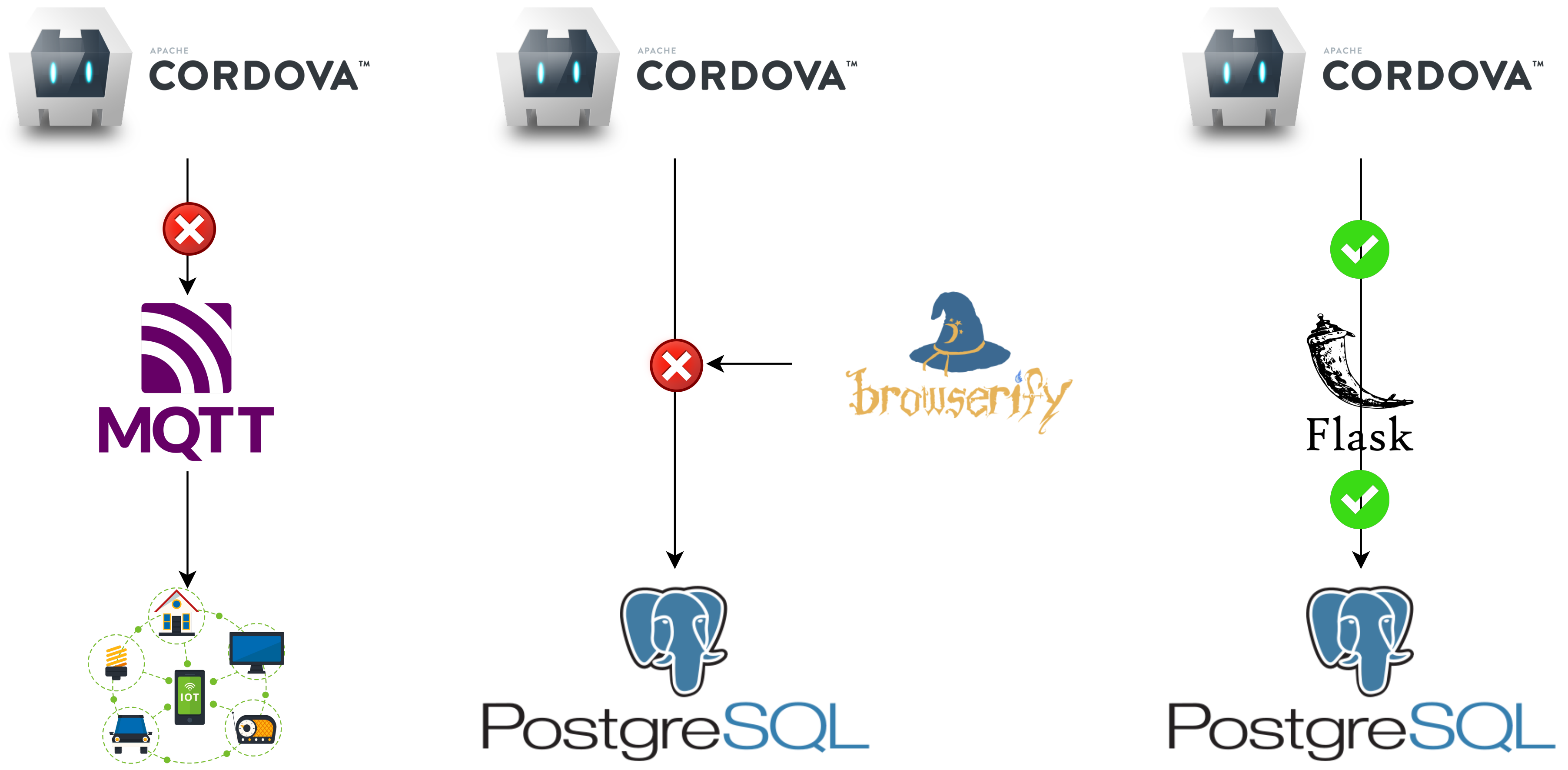
HAW
HAMBURG

INHALTSVERZEICHNIS

TECHNOLOGY OVERVIEW

HAW HAMBURG

TECHNOLOGY OVERVIEW

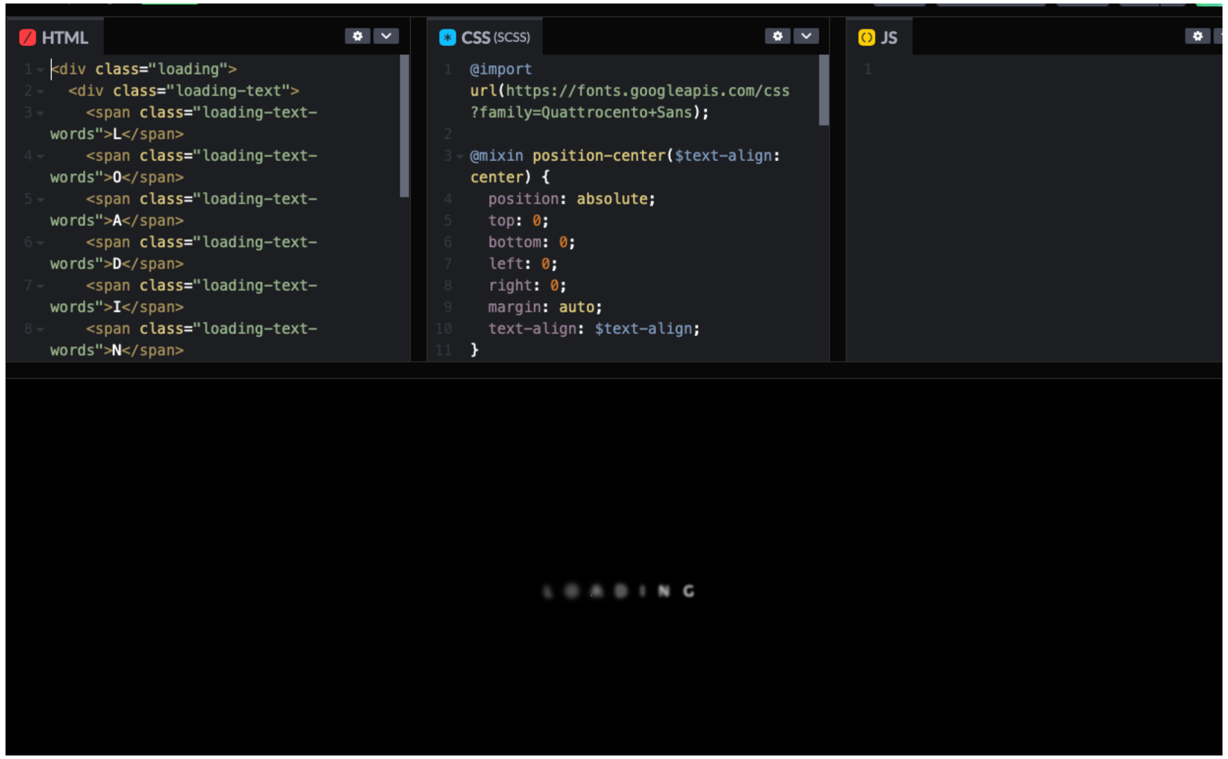


DEVELOPMENT HIGHLIGHTS

HAW HAMBURG

CODEPEN

LOADING PAGE AND LOGIN PAGE



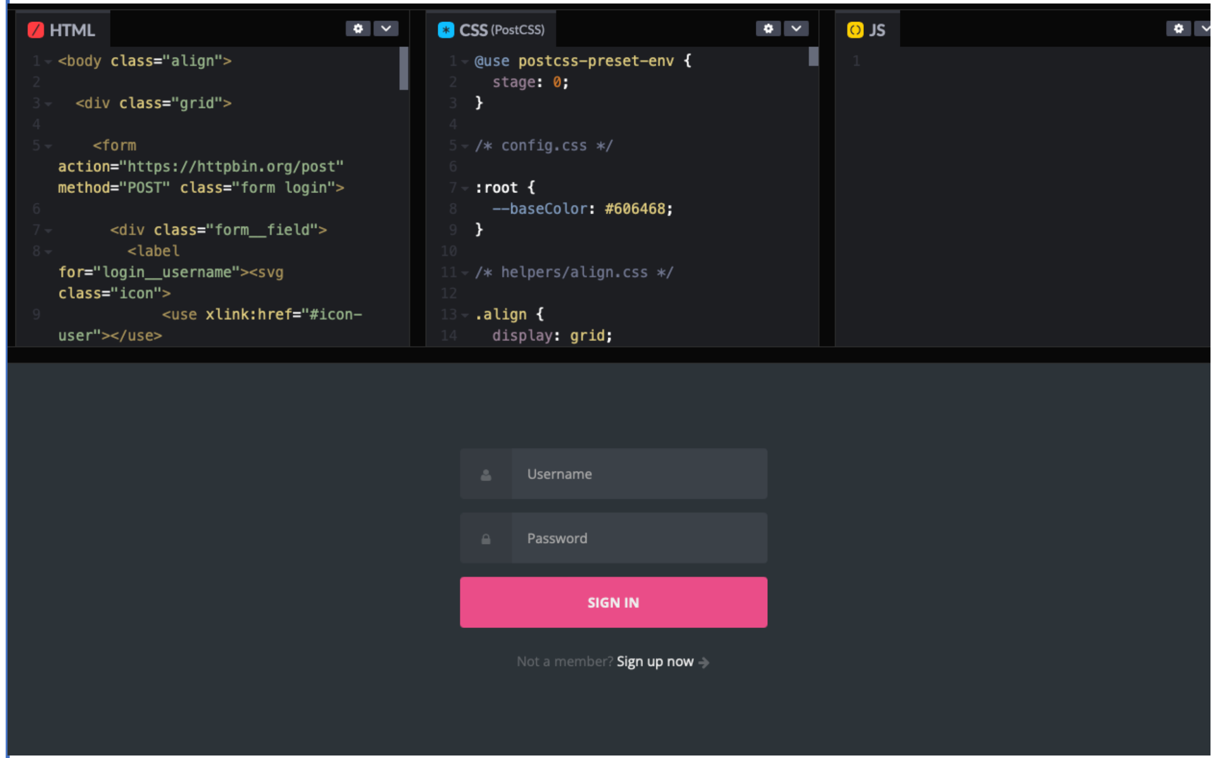
HTML

```html
<div class="loading">
  <div class="loading-text">
    <span class="loading-text-
words">L</span>
    <span class="loading-text-
words">O</span>
    <span class="loading-text-
words">A</span>
    <span class="loading-text-
words">D</span>
    <span class="loading-text-
words">I</span>
    <span class="loading-text-
words">N</span>
```

CSS (SCSS)

```scss
@import
url(https://fonts.googleapis.com/css
?family=Quattrocento+Sans);

@mixin position-center($text-align:
center) {
  position: absolute;
  top: 0;
  bottom: 0;
  left: 0;
  right: 0;
  margin: auto;
  text-align: $text-align;
}
```

JS



HTML

```html
<body class="align">

  <div class="grid">

    <form
action="https://httpbin.org/post"
method="POST" class="form login">

      <div class="form__field">
        <label
for="login__username"><svg
class="icon">
            <use xlink:href="#icon-
user"></use>
```

CSS (PostCSS)

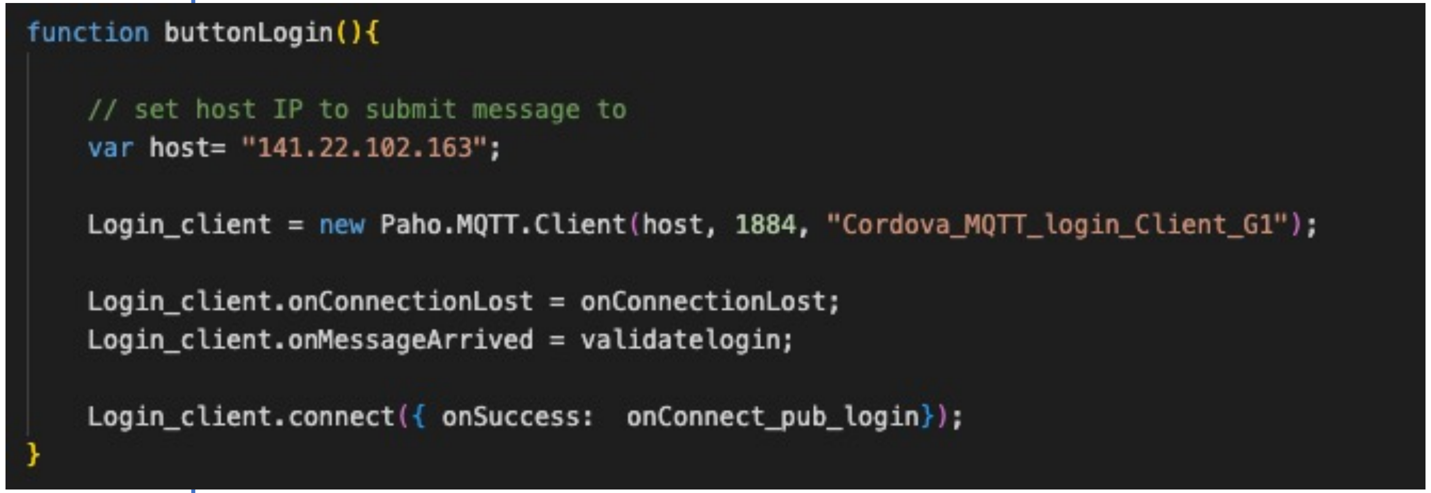
```css
@use postcss-preset-env {
  stage: 0;
}

/* config.css */

:root {
  --baseColor: #606468;
}

/* helpers/align.css */

.align {
  display: grid;
```

JS

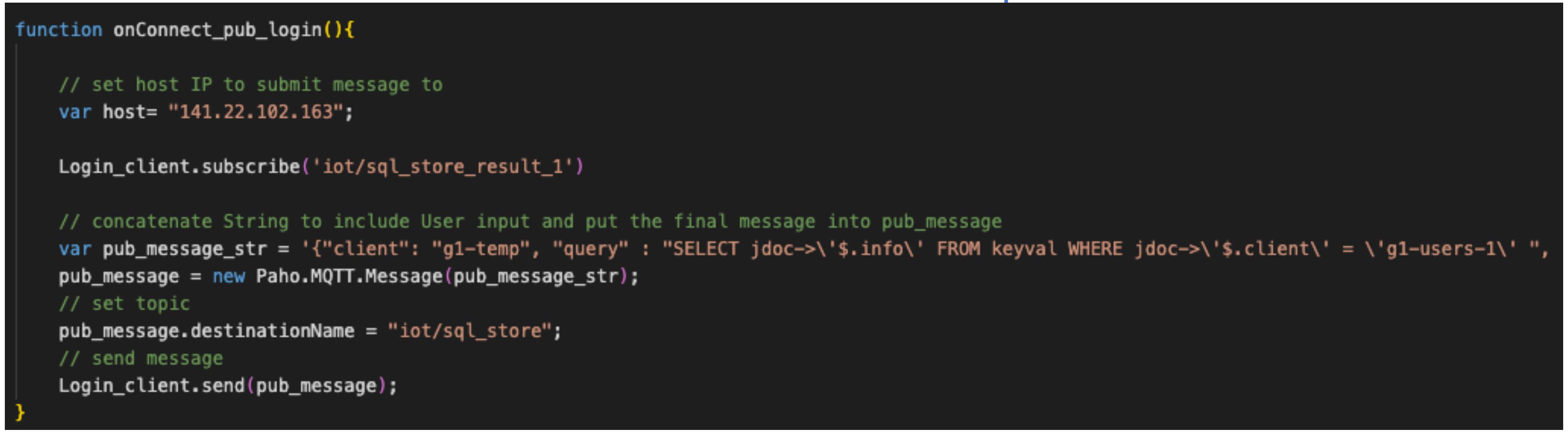
- https://codepen.io/trending
- **https://codepen.io/42EG4M1/pen/bVMzze**
- **https://codepen.io/marcobiedermann/pen/nbpKWV**

LOGIN AUTHENTICATION

- Step 1/3 - Add the users to the Data base
- mosquitto_pub -t iot/store -h 141.22.102.163 -m '{"client" :"g1-users-1", "info" :{"admin":{"password":"haw"}, "student":{"password":"haw"}}}’
- Step 2/3 - Prepare retrieval command and Integrate with App



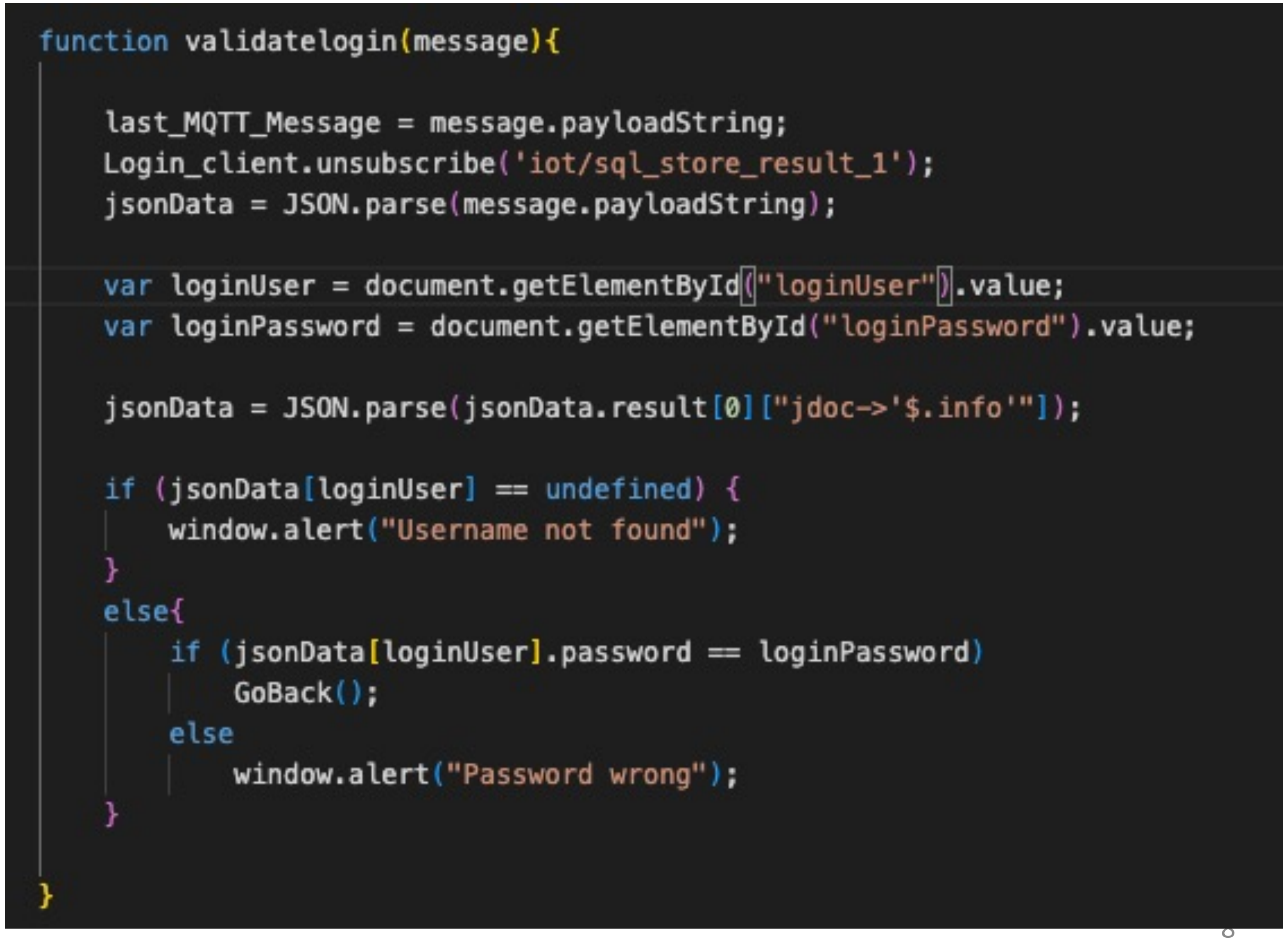
```
function buttonLogin(){

    // set host IP to submit message to
    var host= "141.22.102.163";

    Login_client = new Paho.MQTT.Client(host, 1884, "Cordova_MQTT_login_Client_G1");

    Login_client.onConnectionLost = onConnectionLost;
    Login_client.onMessageArrived = validatelogin;

    Login_client.connect({ onSuccess:  onConnect_pub_login});
}
```



```
function onConnect_pub_login(){

    // set host IP to submit message to
    var host= "141.22.102.163";

    Login_client.subscribe('iot/sql_store_result_1')

    // concatenate String to include User input and put the final message into pub_message
    var pub_message_str = '{"client": "g1-temp", "query" : "SELECT jdoc->\'$.info\' FROM keyval WHERE jdoc->\'$.client\' = \'g1-users-1\' ",
    pub_message = new Paho.MQTT.Message(pub_message_str);
    // set topic
    pub_message.destinationName = "iot/sql_store";
    // send message
    Login_client.send(pub_message);
}
```

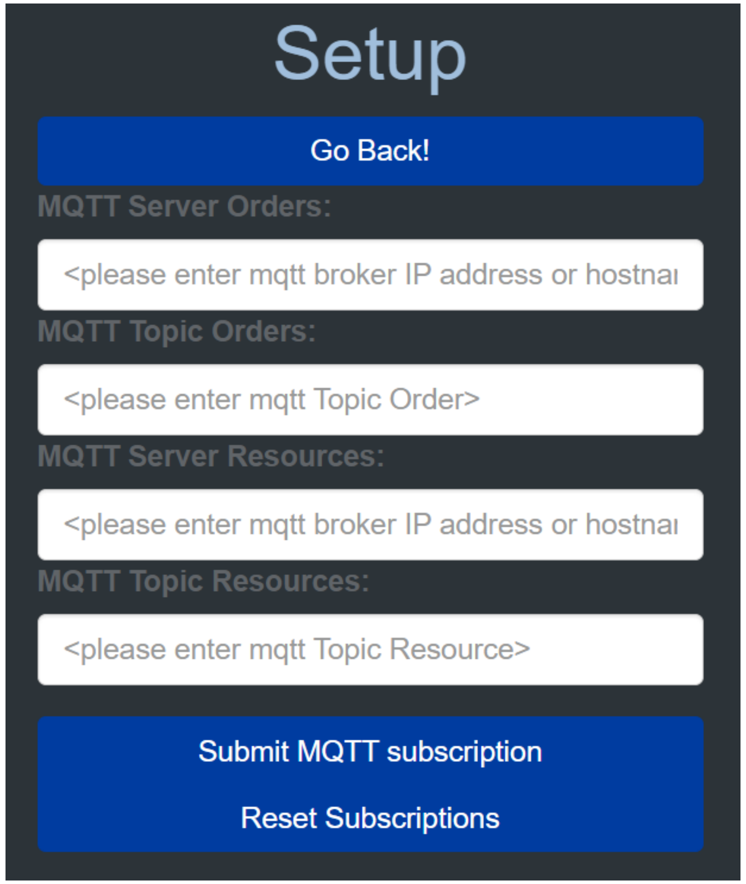
LOGIN AUTHENTICATION

- Step 3/3 - Validate information + Return feedback or proceed to login



```
function validatelogin(message){

    last_MQTT_Message = message.payloadString;
    Login_client.unsubscribe('iot/sql_store_result_1');
    jsonData = JSON.parse(message.payloadString);

    var loginUser = document.getElementById("loginUser").value;
    var loginPassword = document.getElementById("loginPassword").value;

    jsonData = JSON.parse(jsonData.result[0]["jdoc->'$.info'"]);

    if (jsonData[loginUser] == undefined) {
        window.alert("Username not found");
    }
    else{
        if (jsonData[loginUser].password == loginPassword)
            GoBack();
        else
            window.alert("Password wrong");
    }

}
```

SETUP PAGE
MQTT: SPLIT OF ORDERS AND RESOURCES


Setup

Go Back!

MQTT Server Orders:

<please enter mqtt broker IP address or hostnai

MQTT Topic Orders:

<please enter mqtt Topic Order>

MQTT Server Resources:

<please enter mqtt broker IP address or hostnai

MQTT Topic Resources:

<please enter mqtt Topic Resource>

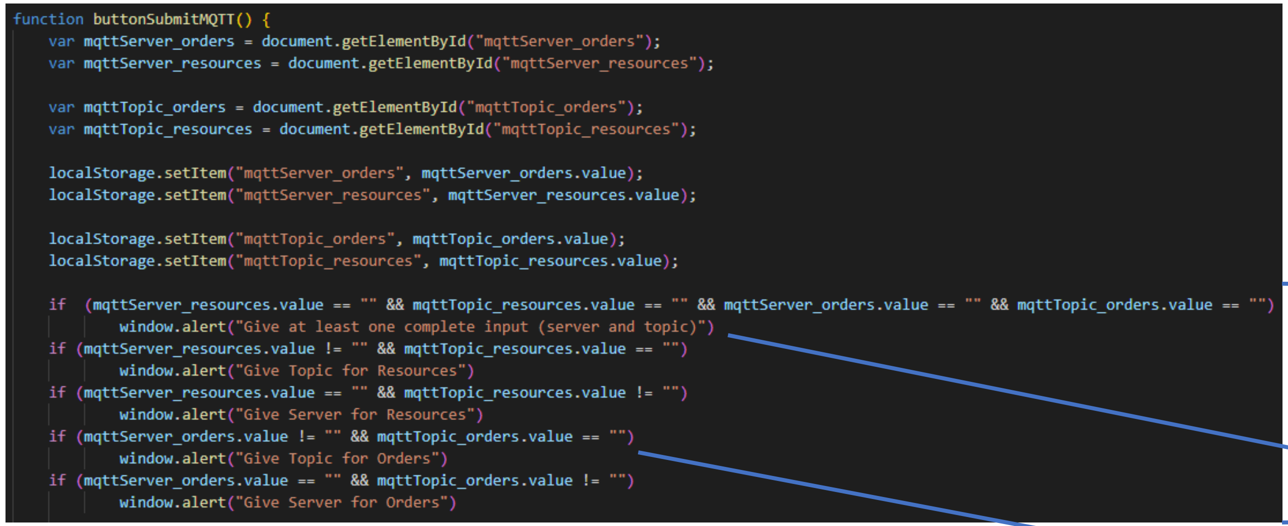
Submit MQTT subscription

Reset Subscriptions

- To meet the requirements of our new features, the server and topic for the MQTT requests from the Orders and Resources pages can now be subscribed to individually
- Full support of all variables in other functions

SETUP PAGE

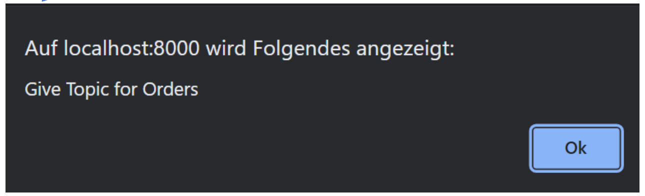
MQTT: SPLIT OF ORDERS AND RESOURCES



```
function buttonSubmitMQTT() {
    var mqttServer_orders = document.getElementById("mqttServer_orders");
    var mqttServer_resources = document.getElementById("mqttServer_resources");

    var mqttTopic_orders = document.getElementById("mqttTopic_orders");
    var mqttTopic_resources = document.getElementById("mqttTopic_resources");

    localStorage.setItem("mqttServer_orders", mqttServer_orders.value);
    localStorage.setItem("mqttServer_resources", mqttServer_resources.value);

    localStorage.setItem("mqttTopic_orders", mqttTopic_orders.value);
    localStorage.setItem("mqttTopic_resources", mqttTopic_resources.value);

    if  (mqttServer_resources.value == "" && mqttTopic_resources.value == "" && mqttServer_orders.value == "" && mqttTopic_orders.value == "")
            window.alert("Give at least one complete input (server and topic)")
    if (mqttServer_resources.value != "" && mqttTopic_resources.value == "")
            window.alert("Give Topic for Resources")
    if (mqttServer_resources.value == "" && mqttTopic_resources.value != "")
            window.alert("Give Server for Resources")
    if (mqttServer_orders.value != "" && mqttTopic_orders.value == "")
            window.alert("Give Topic for Orders")
    if (mqttServer_orders.value == "" && mqttTopic_orders.value != "")
            window.alert("Give Server for Orders")
```

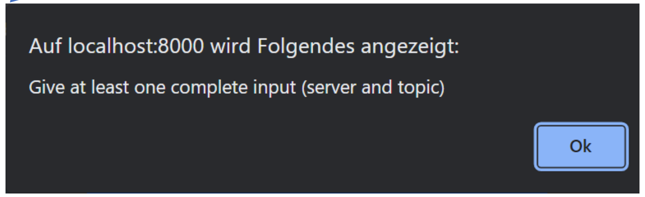
- When the Submit button is pressed, inputs are stored in the local storage as before
- For this, new variables had to be created and specified

- There is also a check whether individual or all fields are empty



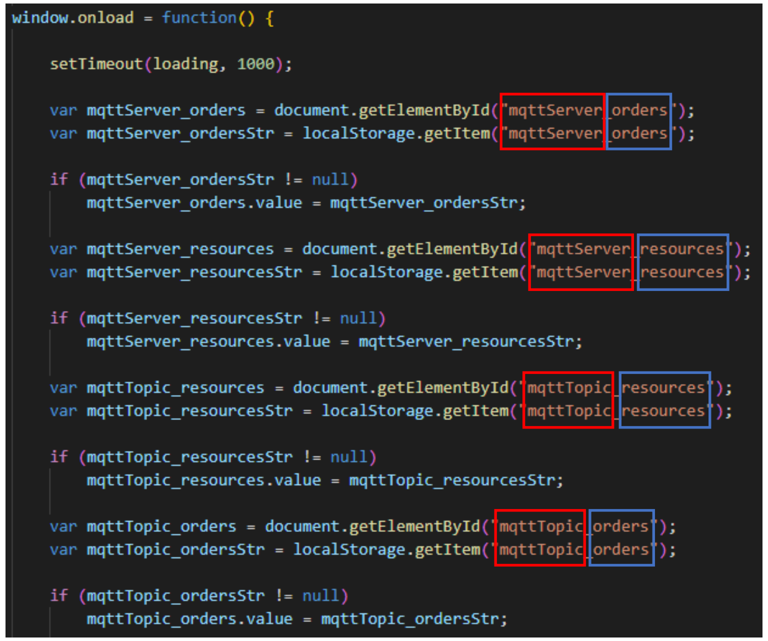
Auf localhost:8000 wird Folgendes angezeigt:

Give Topic for Orders



Auf localhost:8000 wird Folgendes angezeigt:

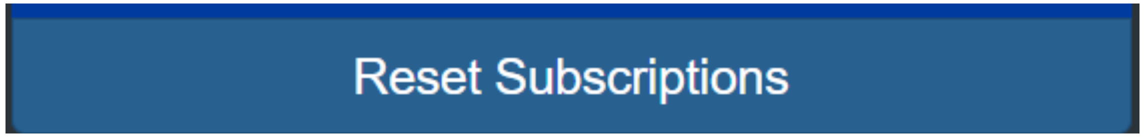
Give at least one complete input (server and topic)



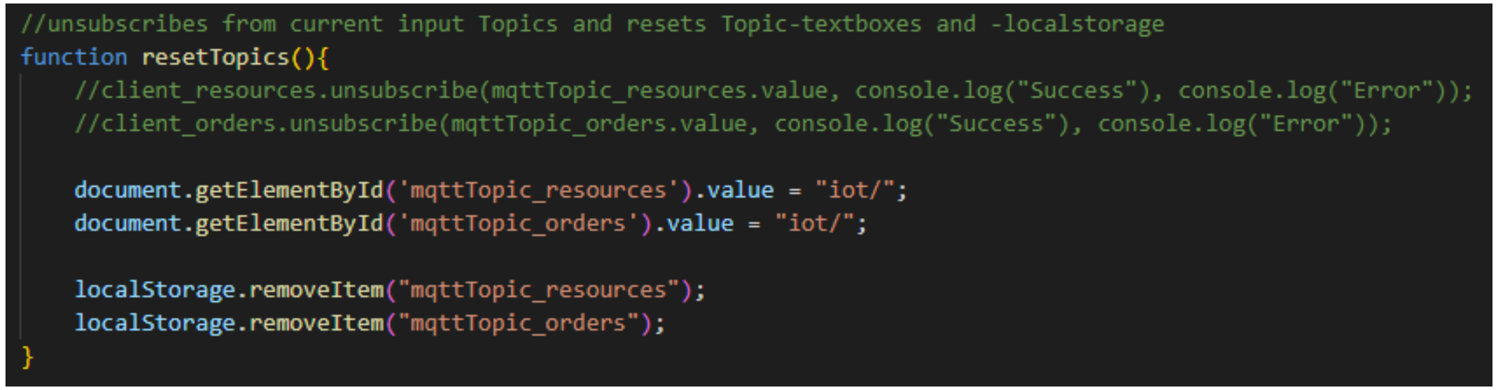
```
window.onload = function() {

    setTimeout(loading, 1000);

    var mqttServer_orders = document.getElementById("mqttServer_orders");
    var mqttServer_ordersStr = localStorage.getItem("mqttServer_orders");

    if (mqttServer_ordersStr != null)
        mqttServer_orders.value = mqttServer_ordersStr;

    var mqttServer_resources = document.getElementById("mqttServer_resources");
    var mqttServer_resourcesStr = localStorage.getItem("mqttServer_resources");

    if (mqttServer_resourcesStr != null)
        mqttServer_resources.value = mqttServer_resourcesStr;

    var mqttTopic_resources = document.getElementById("mqttTopic_resources");
    var mqttTopic_resourcesStr = localStorage.getItem("mqttTopic_resources");

    if (mqttTopic_resourcesStr != null)
        mqttTopic_resources.value = mqttTopic_resourcesStr;

    var mqttTopic_orders = document.getElementById("mqttTopic_orders");
    var mqttTopic_ordersStr = localStorage.getItem("mqttTopic_orders");

    if (mqttTopic_ordersStr != null)
        mqttTopic_orders.value = mqttTopic_ordersStr;
```

- When the app is started, all variables are of course still retrieved from the localStorage

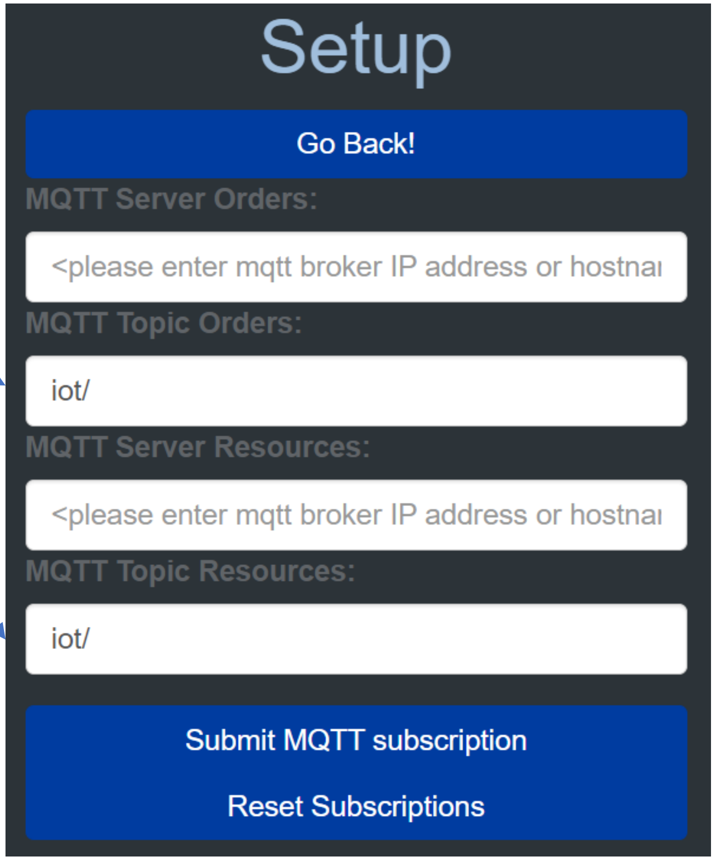
SETUP PAGE
RESET SUBSCRIPTIONS



- The „Reset Subscriptions“-Button unsubscribes all currently selected Topics and removes them from the localStorage
- At the same time, all Topic-Boxes are prepared for the new input



```
//unsubscribes from current input Topics and resets Topic-textboxes and -localstorage
function resetTopics(){
    //client_resources.unsubscribe(mqttTopic_resources.value, console.log("Success"), console.log("Error"));
    //client_orders.unsubscribe(mqttTopic_orders.value, console.log("Success"), console.log("Error"));

    document.getElementById('mqttTopic_resources').value = "iot/";
    document.getElementById('mqttTopic_orders').value = "iot/";

    localStorage.removeItem("mqttTopic_resources");
    localStorage.removeItem("mqttTopic_orders");
}
```



Setup

Go Back!

MQTT Server Orders:

<please enter mqtt broker IP address or hostnai

MQTT Topic Orders:

iot/

MQTT Server Resources:

<please enter mqtt broker IP address or hostnai

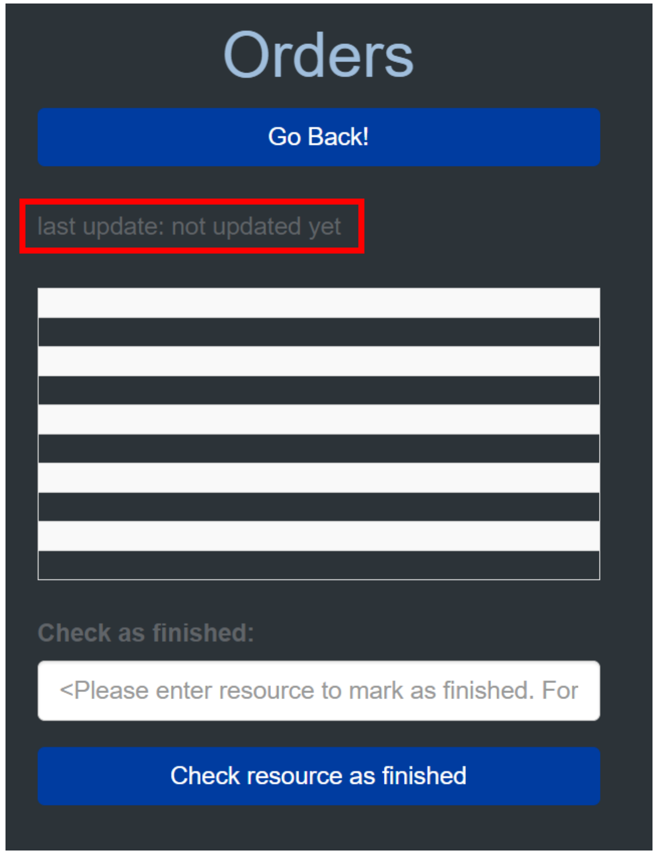
MQTT Topic Resources:

iot/

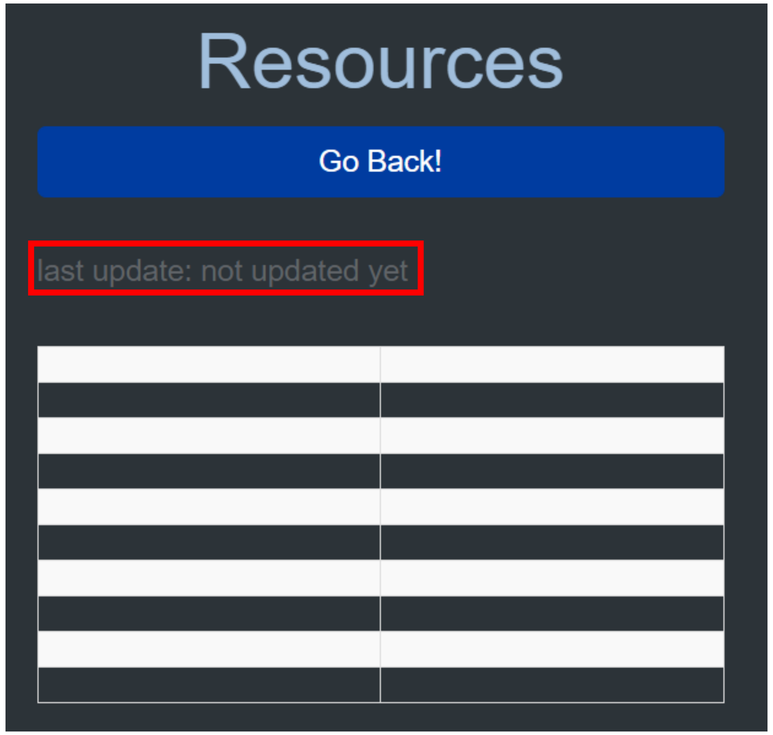
Submit MQTT subscription

Reset Subscriptions

LATEST UPDATE
FRONT END

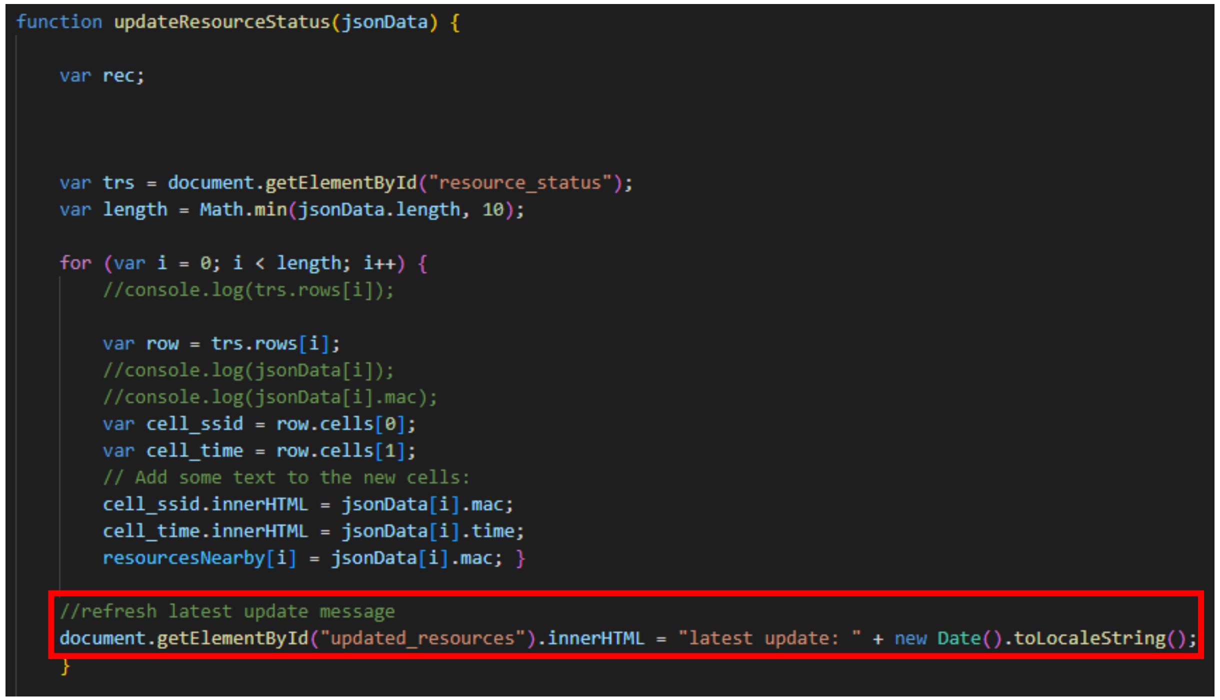

Orders
Go Back!
last update: not updated yet
Check as finished:
<Please enter resource to mark as finished. For
Check resource as finished

- Always shows the time of the latest successfull MQTT-Request
- User always knows if the the currently displayed data is outdated

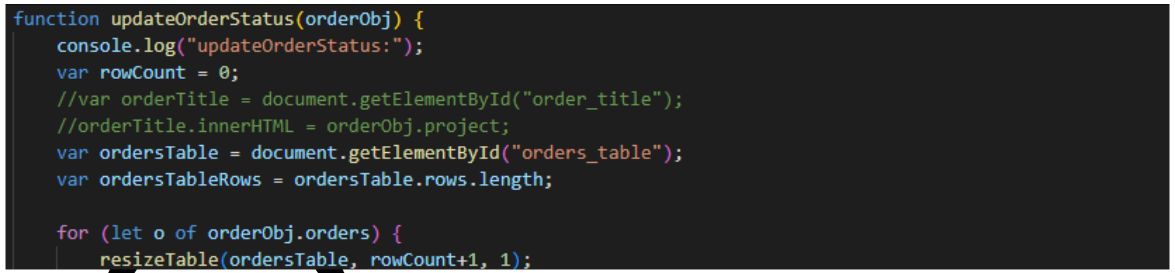


LATEST UPDATE

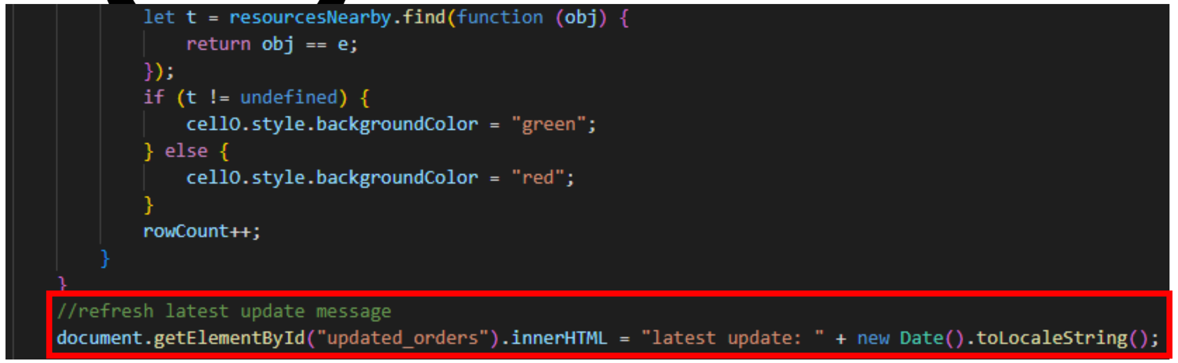
BACK END



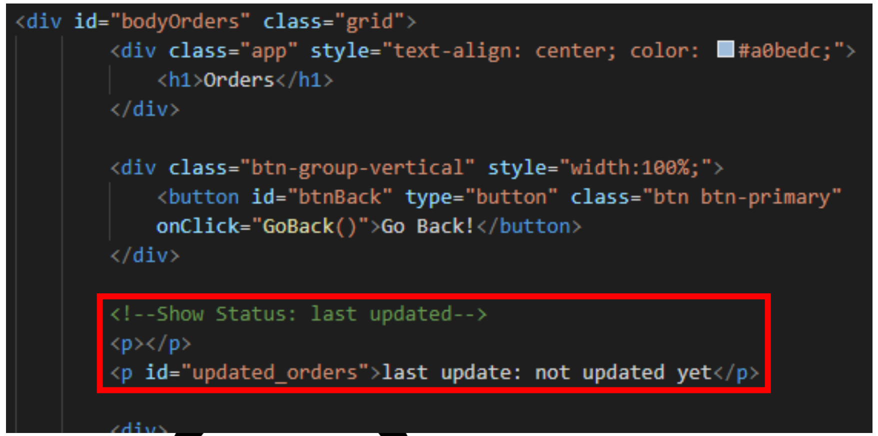
```javascript
function updateResourceStatus(jsonData) {

    var rec;


    var trs = document.getElementById("resource_status");
    var length = Math.min(jsonData.length, 10);

    for (var i = 0; i < length; i++) {
        //console.log(trs.rows[i]);

        var row = trs.rows[i];
        //console.log(jsonData[i]);
        //console.log(jsonData[i].mac);
        var cell_ssid = row.cells[0];
        var cell_time = row.cells[1];
        // Add some text to the new cells:
        cell_ssid.innerHTML = jsonData[i].mac;
        cell_time.innerHTML = jsonData[i].time;
        resourcesNearby[i] = jsonData[i].mac; }

    //refresh latest update message
    document.getElementById("updated_resources").innerHTML = "latest update: " + new Date().toLocaleString();
    }
```



```javascript
function updateOrderStatus(orderObj) {
    console.log("updateOrderStatus:");
    var rowCount = 0;
    //var orderTitle = document.getElementById("order_title");
    //orderTitle.innerHTML = orderObj.project;
    var ordersTable = document.getElementById("orders_table");
    var ordersTableRows = ordersTable.rows.length;

    for (let o of orderObj.orders) {
        resizeTable(ordersTable, rowCount+1, 1);
```



```javascript
            let t = resourcesNearby.find(function (obj) {
                return obj == e;
            });
            if (t != undefined) {
                cell0.style.backgroundColor = "green";
            } else {
                cell0.style.backgroundColor = "red";
            }
            rowCount++;
        }
    }
    //refresh latest update message
    document.getElementById("updated_orders").innerHTML = "latest update: " + new Date().toLocaleString();
```



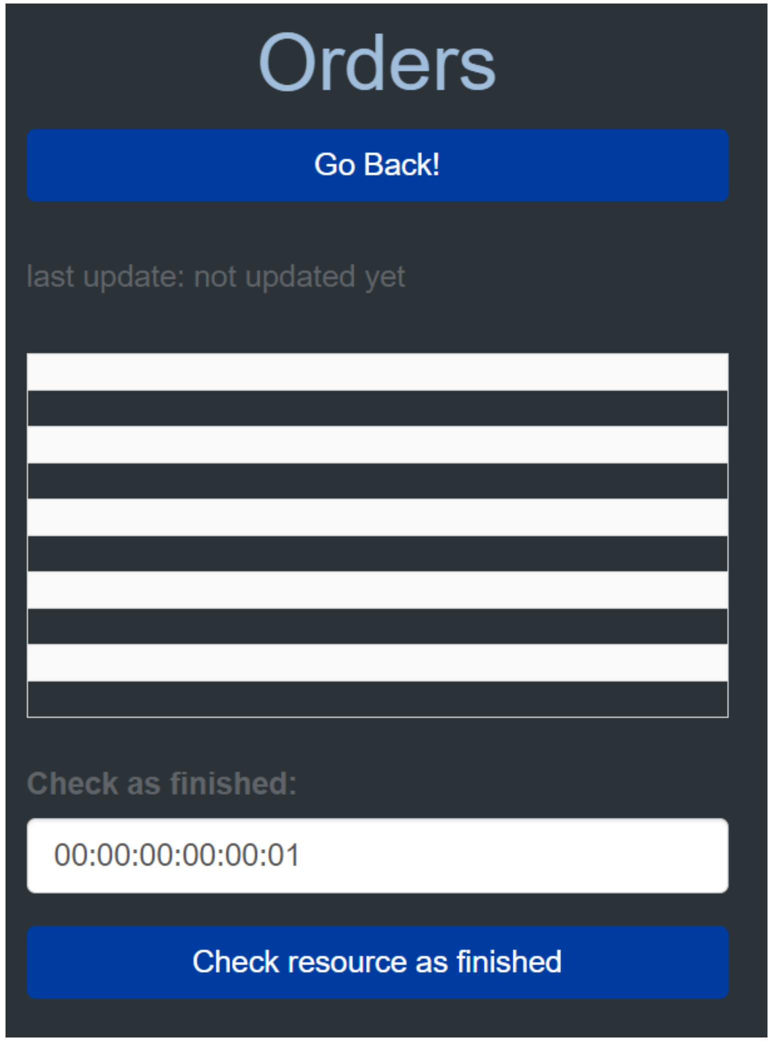
```html
<div id="bodyOrders" class="grid">
        <div class="app" style="text-align: center; color: #a0bedc;">
            <h1>Orders</h1>
        </div>

        <div class="btn-group-vertical" style="width:100%;">
            <button id="btnBack" type="button" class="btn btn-primary"
            onClick="GoBack()">Go Back!</button>
        </div>

        <!--Show Status: last updated-->
        <p></p>
        <p id="updated_orders">last update: not updated yet</p>

        <div>
```

Implementation is very simple:

- everytime a new MQTT-Message is received, the corresponding updateStatus function is executed
- At the very end of it the Element „updated_resources“ or „updated_orders“ are update with the current time

CHECK RESOURCES AS FINISHED

FRONT END

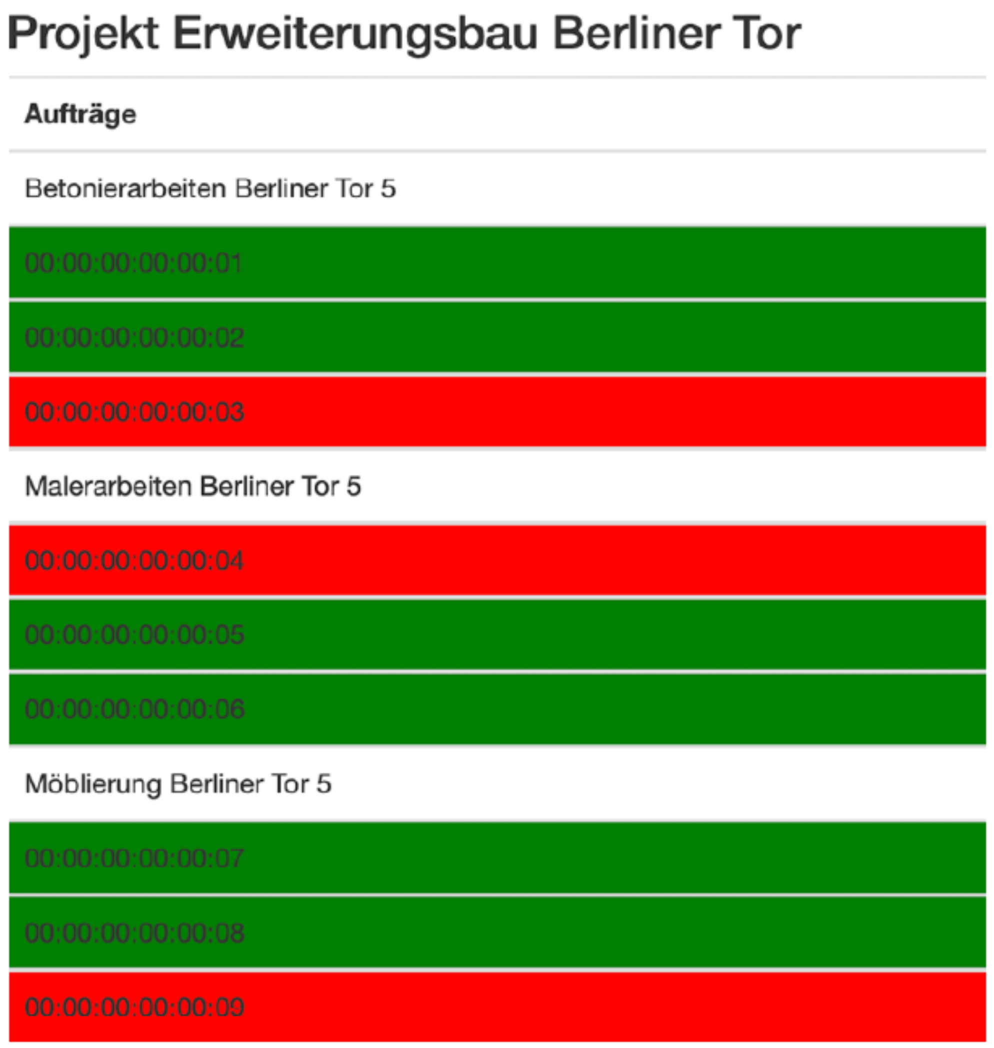


Orders

last update: not updated yet

Check as finished:

00:00:00:00:00:01



Projekt Erweiterungsbau Berliner Tor

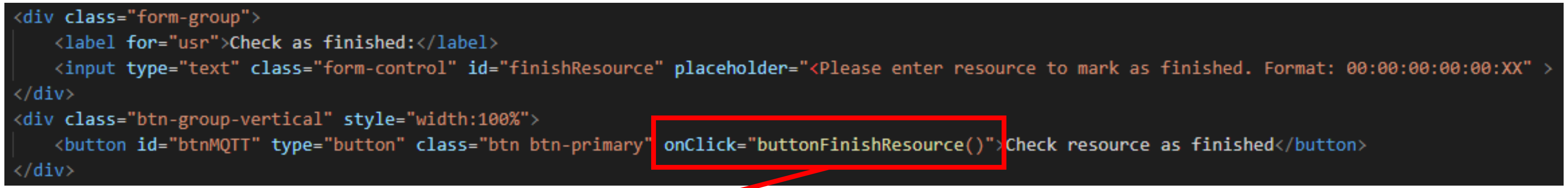
Aufträge
Betonierarbeiten Berliner Tor 5
🟩 00:00:00:00:00:01
🟩 00:00:00:00:00:02
🟥 00:00:00:00:00:03
Malerarbeiten Berliner Tor 5
🟥 00:00:00:00:00:04
🟩 00:00:00:00:00:05
🟩 00:00:00:00:00:06
Möblierung Berliner Tor 5
🟩 00:00:00:00:00:07
🟩 00:00:00:00:00:08
🟥 00:00:00:00:00:09

Abbildung 8 - Ergebnis Aufgabe 14

(Abbildung aus Vorlesungsfolien)

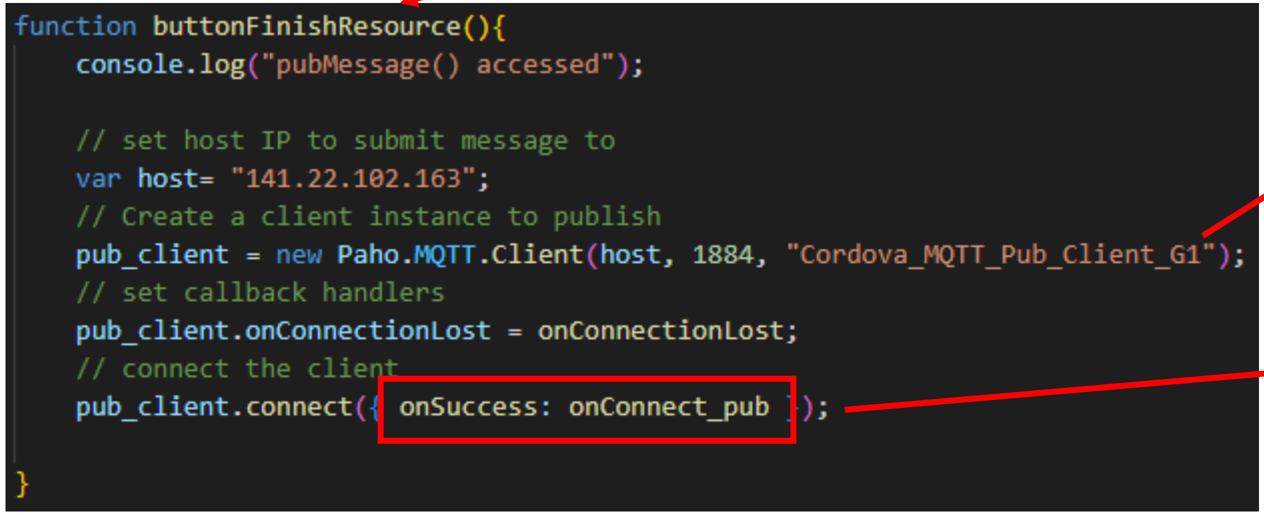
CHECK RESOURCES AS FINISHED
BACK END

HTML-Implementation:



```
<div class="form-group">
    <label for="usr">Check as finished:</label>
    <input type="text" class="form-control" id="finishResource" placeholder="<Please enter resource to mark as finished. Format: 00:00:00:00:00:XX" >
</div>
<div class="btn-group-vertical" style="width:100%">
    <button id="btnMQTT" type="button" class="btn btn-primary" onClick="buttonFinishResource()">Check resource as finished</button>
</div>
```

Function:



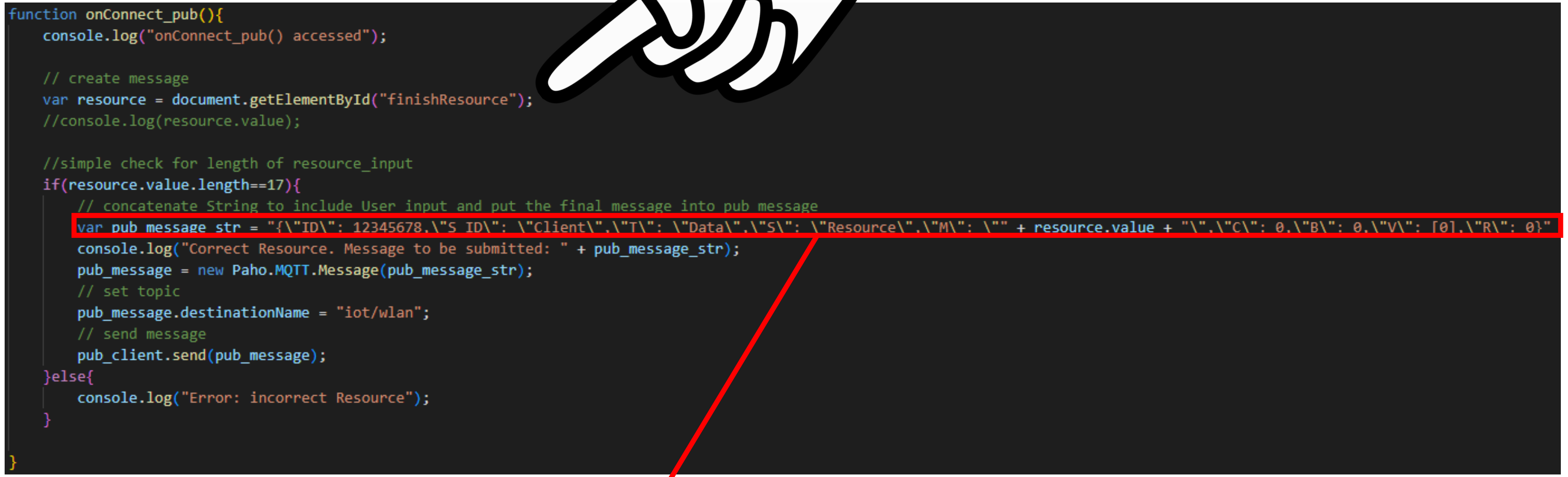
```
function buttonFinishResource(){
    console.log("pubMessage() accessed");

    // set host IP to submit message to
    var host= "141.22.102.163";
    // Create a client instance to publish
    pub_client = new Paho.MQTT.Client(host, 1884, "Cordova_MQTT_Pub_Client_G1");
    // set callback handlers
    pub_client.onConnectionLost = onConnectionLost;
    // connect the client
    pub_client.connect({ onSuccess: onConnect_pub });

}
```

- Creates a new Client to publish messages
- When successfully connected, the function onConnect_pub is called

CHECK RESOURCES AS FINISHED

BACK END

onConnect_pub:



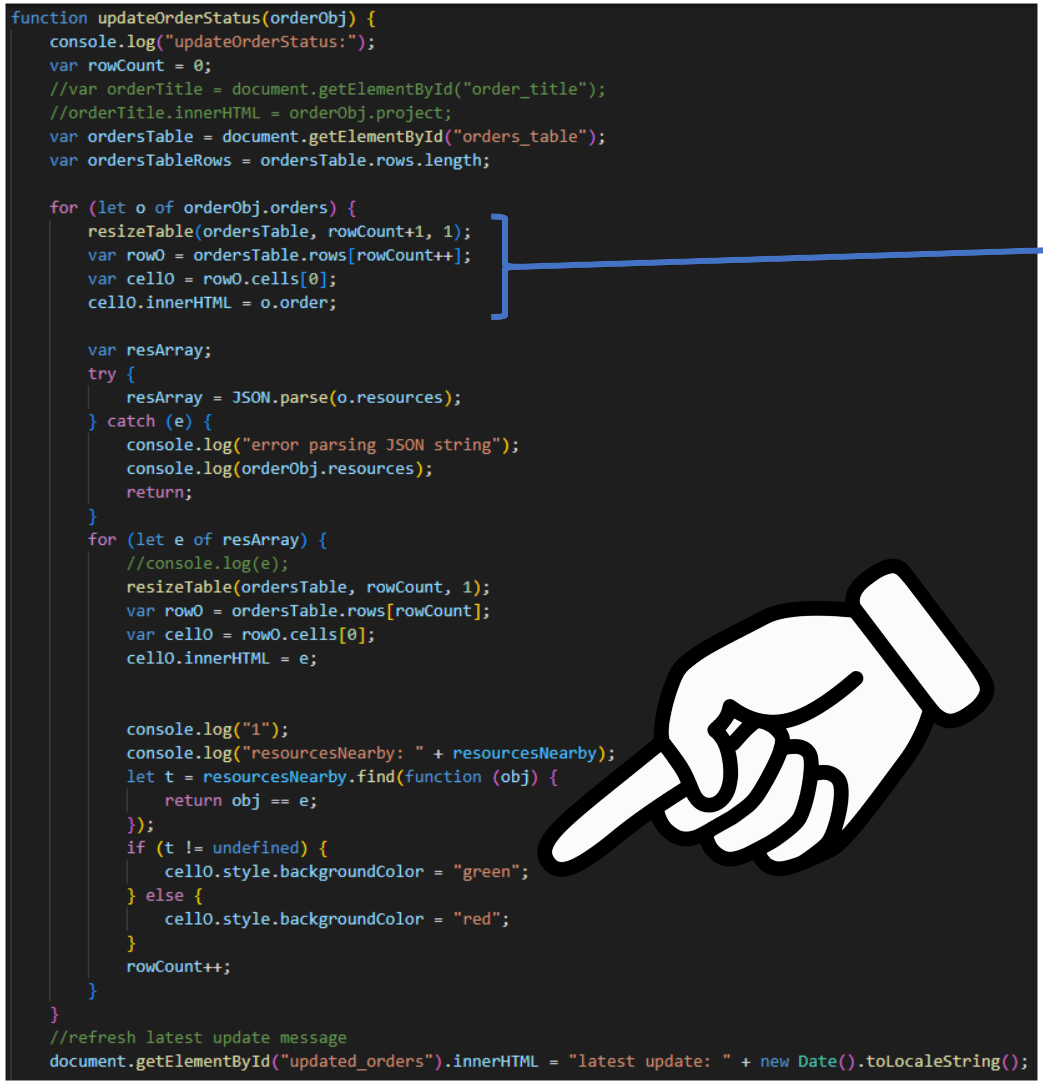
```
function onConnect_pub(){
    console.log("onConnect_pub() accessed");

    // create message
    var resource = document.getElementById("finishResource");
    //console.log(resource.value);

    //simple check for length of resource_input
    if(resource.value.length==17){
        // concatenate String to include User input and put the final message into pub message
        var pub_message_str = "{\"ID\": 12345678,\"S_ID\": \"Client\",\"T\": \"Data\",\"S\": \"Resource\",\"M\": \"" + resource.value + "\",\"C\": 0,\"B\": 0,\"V\": [0],\"R\": 0}"
        console.log("Correct Resource. Message to be submitted: " + pub_message_str);
        pub_message = new Paho.MQTT.Message(pub_message_str);
        // set topic
        pub_message.destinationName = "iot/wlan";
        // send message
        pub_client.send(pub_message);
    }else{
        console.log("Error: incorrect Resource");
    }

}
```

Structure of the JSON object to be sent:

```
{
    "ID": 12345678,
    "S_ID": "Client",
    "T": "Data",
    "S": "Resource",
    "M": "00:00:00:00:00:EE",
    "C": 0,
    "B": 0,
    "V": [0],
    "R": 0
}
```

CHECK RESOURCES AS FINISHED
BACK END

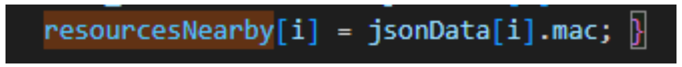
updateOrderStatus



```
function updateOrderStatus(orderObj) {
    console.log("updateOrderStatus:");
    var rowCount = 0;
    //var orderTitle = document.getElementById("order_title");
    //orderTitle.innerHTML = orderObj.project;
    var ordersTable = document.getElementById("orders_table");
    var ordersTableRows = ordersTable.rows.length;

    for (let o of orderObj.orders) {
        resizeTable(ordersTable, rowCount+1, 1);
        var row0 = ordersTable.rows[rowCount++];
        var cell0 = row0.cells[0];
        cell0.innerHTML = o.order;

        var resArray;
        try {
            resArray = JSON.parse(o.resources);
        } catch (e) {
            console.log("error parsing JSON string");
            console.log(orderObj.resources);
            return;
        }
        for (let e of resArray) {
            //console.log(e);
            resizeTable(ordersTable, rowCount, 1);
            var row0 = ordersTable.rows[rowCount];
            var cell0 = row0.cells[0];
            cell0.innerHTML = e;


            console.log("1");
            console.log("resourcesNearby: " + resourcesNearby);
            let t = resourcesNearby.find(function (obj) {
                return obj == e;
            });
            if (t != undefined) {
                cell0.style.backgroundColor = "green";
            } else {
                cell0.style.backgroundColor = "red";
            }
            rowCount++;
        }
    }
    //refresh latest update message
    document.getElementById("updated_orders").innerHTML = "latest update: " + new Date().toLocaleString();
```

function iterates over all available resources

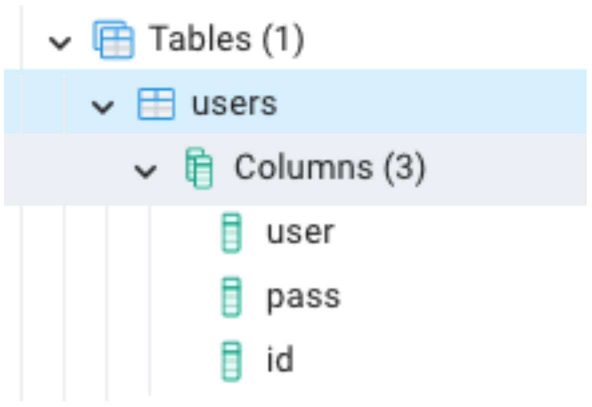
- Dynamic adjustment of the table size

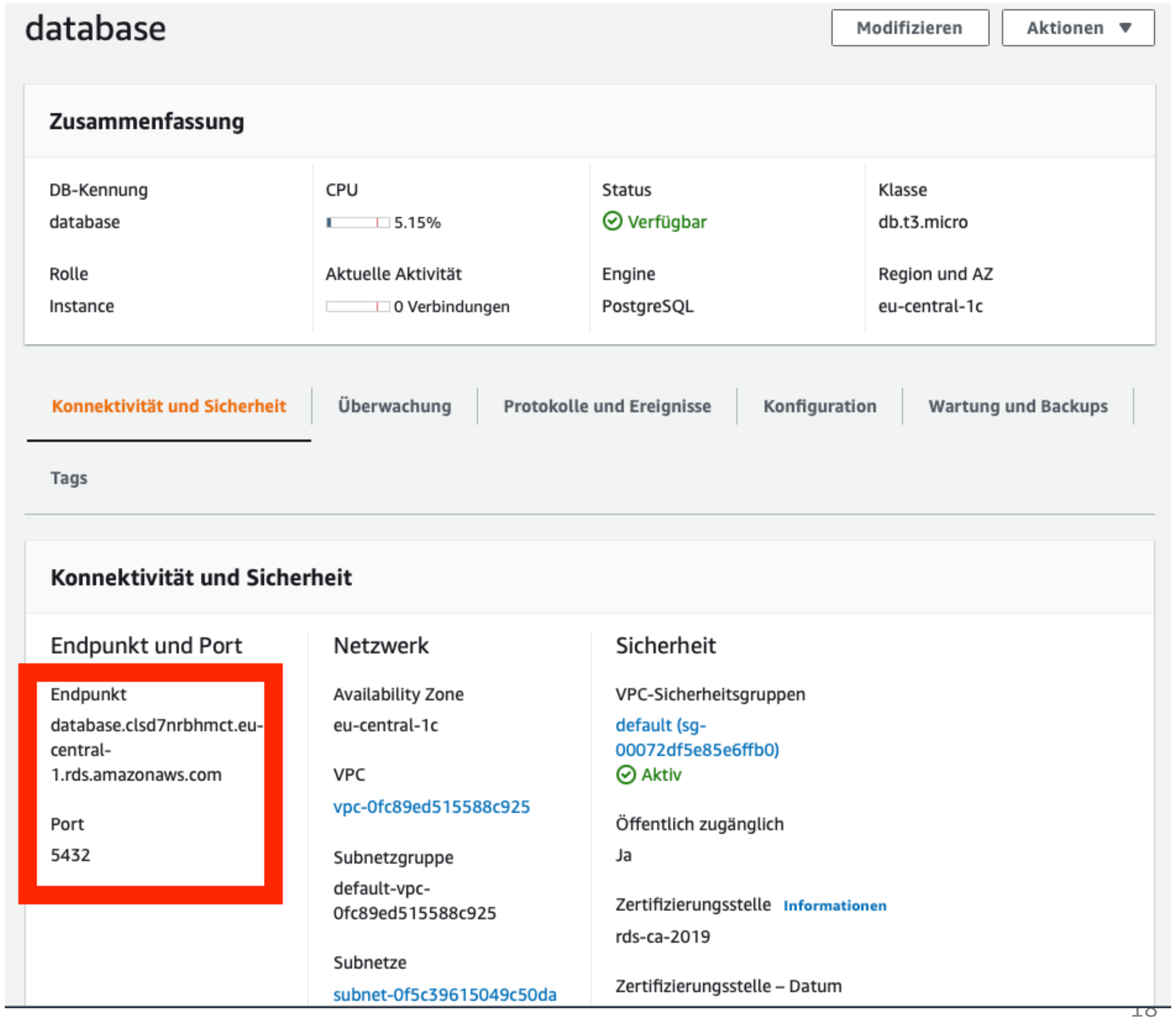


```
resourcesNearby[i] = jsonData[i].mac; }
```

- resourcesNearby is updated in the updateResourceStatus function

POSTGRESQL


Tables (1)
users
Columns (3)
user
pass
id


database

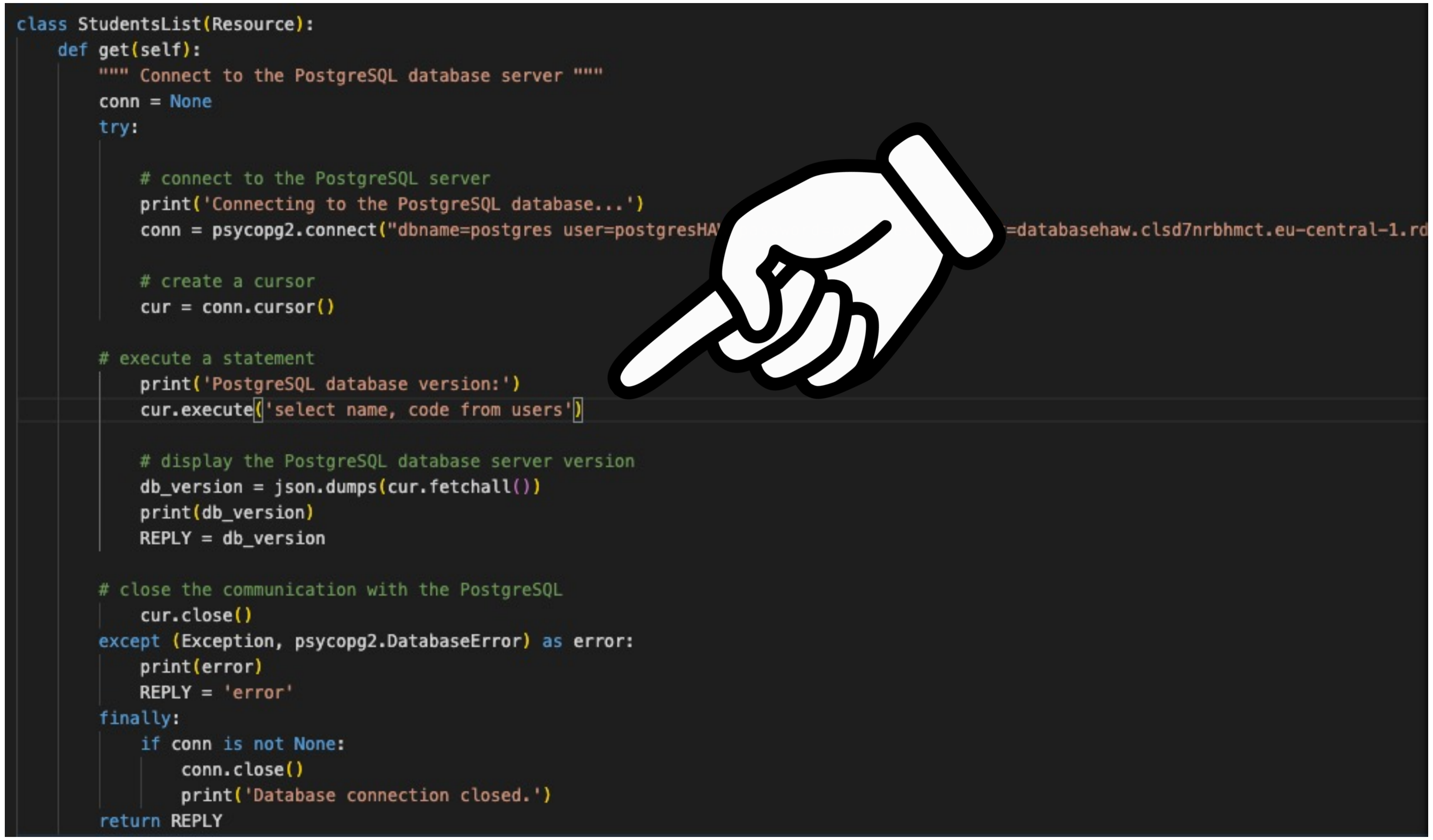
Zusammenfassung

DB-Kennung	CPU	Status	Klasse
database	5.15%	Verfügbar	db.t3.micro
Rolle	Aktuelle Aktivität	Engine	Region und AZ
Instance	0 Verbindungen	PostgreSQL	eu-central-1c

Konnektivität und Sicherheit

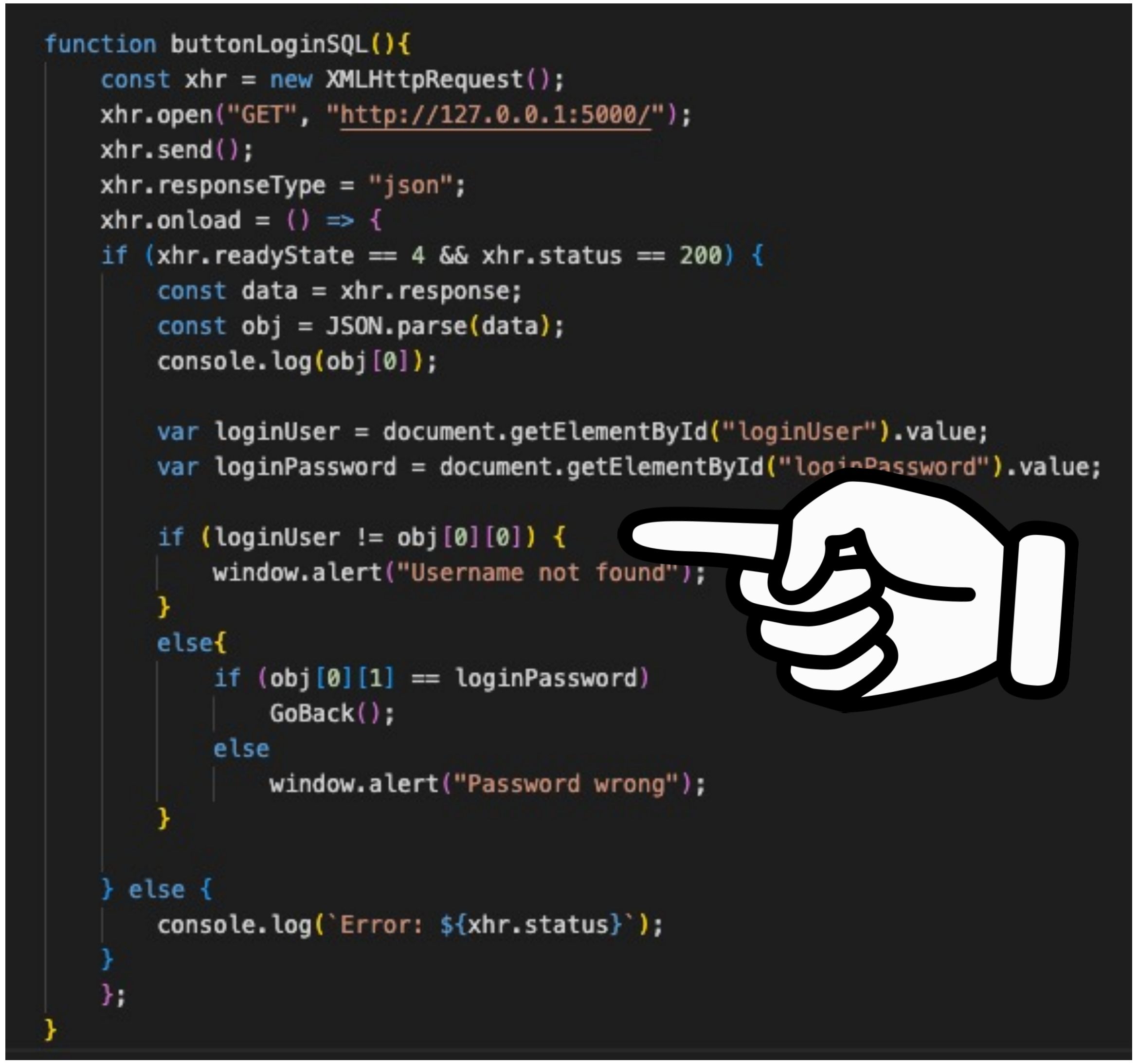
Endpunkt und Port	Netzwerk	Sicherheit
Endpunkt	Availability Zone	VPC-Sicherheitsgruppen
database.clsd7nrbhmct.eu-central-1.rds.amazonaws.com	eu-central-1c	default (sg-00072df5e85e6ffb0) Aktiv
Port	VPC	Öffentlich zugänglich
5432	vpc-0fc89ed515588c925	Ja
	Subnetzgruppe	Zertifizierungsstelle Informationen
	default-vpc-0fc89ed515588c925	rds-ca-2019
	Subnetze	Zertifizierungsstelle – Datum
	subnet-0f5c39615049c50da	

FLASK
BACK END



```python
class StudentsList(Resource):
    def get(self):
        """ Connect to the PostgreSQL database server """
        conn = None
        try:

            # connect to the PostgreSQL server
            print('Connecting to the PostgreSQL database...')
            conn = psycopg2.connect("dbname=postgres user=postgresHA[illegible]=databasehaw.clsd7nrbhmct.eu-central-1.rd

            # create a cursor
            cur = conn.cursor()

        # execute a statement
            print('PostgreSQL database version:')
            cur.execute('select name, code from users')

            # display the PostgreSQL database server version
            db_version = json.dumps(cur.fetchall())
            print(db_version)
            REPLY = db_version

        # close the communication with the PostgreSQL
            cur.close()
        except (Exception, psycopg2.DatabaseError) as error:
            print(error)
            REPLY = 'error'
        finally:
            if conn is not None:
                conn.close()
                print('Database connection closed.')
        return REPLY
```

CORDOVA -> POSTGRE
BACK END



```
function buttonLoginSQL(){
    const xhr = new XMLHttpRequest();
    xhr.open("GET", "http://127.0.0.1:5000/");
    xhr.send();
    xhr.responseType = "json";
    xhr.onload = () => {
    if (xhr.readyState == 4 && xhr.status == 200) {
        const data = xhr.response;
        const obj = JSON.parse(data);
        console.log(obj[0]);

        var loginUser = document.getElementById("loginUser").value;
        var loginPassword = document.getElementById("loginPassword").value;

        if (loginUser != obj[0][0]) {
            window.alert("Username not found");
        }
        else{
            if (obj[0][1] == loginPassword)
                GoBack();
            else
                window.alert("Password wrong");
        }

    } else {
        console.log(`Error: ${xhr.status}`);
    }
    };
}
```

LIMITATIONS

LIMITATIONS

- Security
  - Authentication page only hides real content, but its all loaded in the browser.
  - MQTT connection is made directly from the browser, exposing the address and port in client side.
  - Password check is made on client side
- Scalability
  - Although making the app load at once can make it faster, if the app were to be continue developed, this could limit development freedom.
- Stability
  - complete dependence of the app on server communication
  - No automatic reconnection in case of server crash

APP TEST

HAW HAMBURG

VIELEN DANK FÜR DIE AUFMERKSAMKEIT

HAW
HAMBURG