NowSeat

Real time public transport information

Christian Stingone, Alessio Drago, Alex Di Stefano

Index

Project Proposal

Often it happens to bring valuables, fragile or bulky objects with you and before getting on a public transport we always hope to find a seat. Through our implementation we want to provide the end user with this information in real time for various types of public transport, accessible via the website

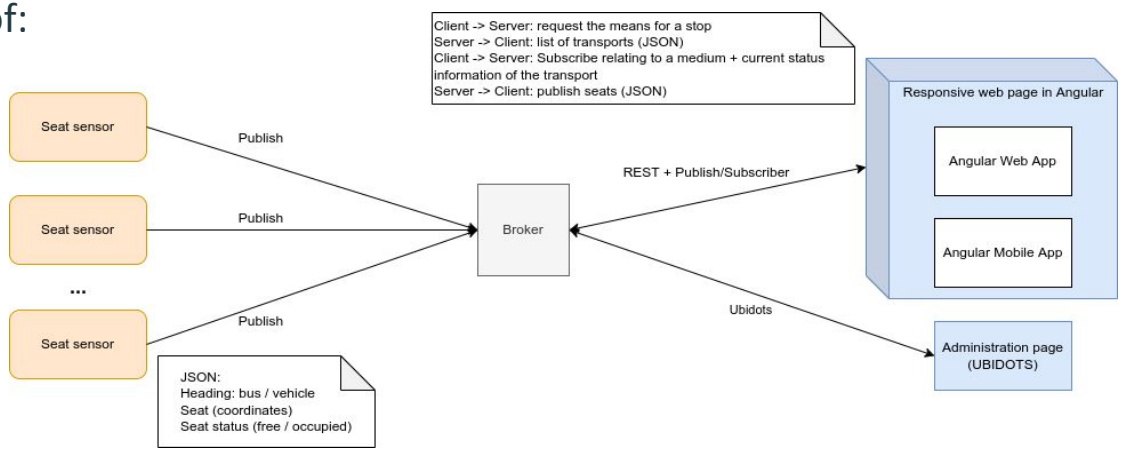
This project aims to meet the needs of those who, taking public transport, need to know how many people are actually on the vehicle so that they can choose whether to get on or not thanks to this information

SMART CITY

General Architecture

Architecture is made up of:

- Angular Website
- Node.js Broker
- Sensors
- Ubidots



Communication and Middleware

For communication we have two REST APIs and the MQTT protocol:

`GET /?stopCode={stopCode}`

It requests a list of transportation in the proximity / arrival of a stop as a parameter

`GET /?idTransport={idTransport}`

It requests the current status of the seats of a given transportation identified by a code passed as a parameter

`MQTT TOPIC = idTransport`

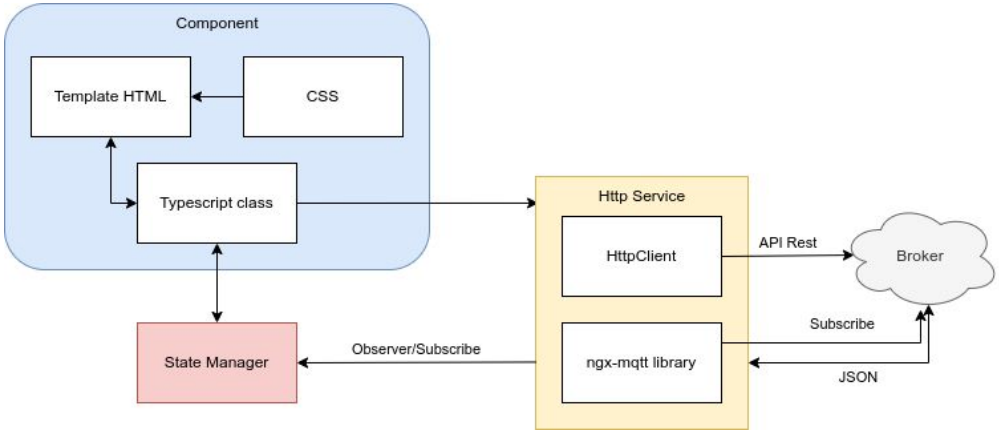
Publish/Subscribe to the Broker to exchange information provided by the sensors installed in the seats

Implementation

Web App

The Web App was developed in **Angular** as this framework autonomously manages the updating of the DOM at each change in the state of the internal variables.

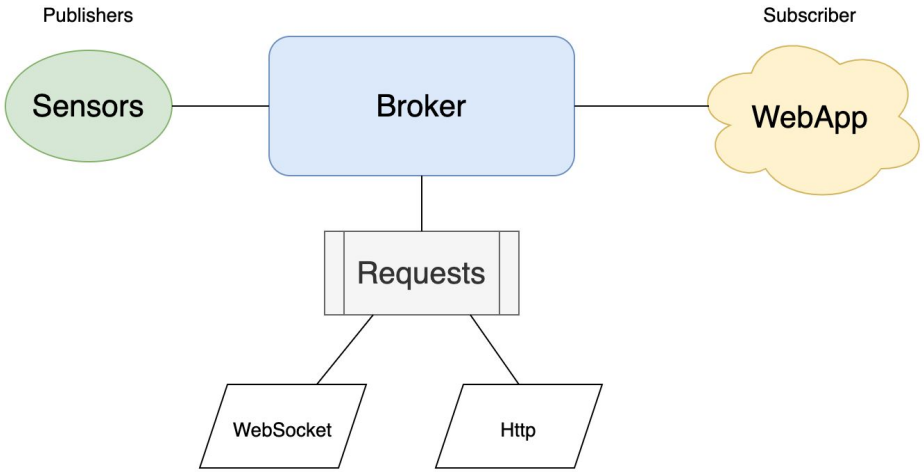
Furthermore, the use of typescripts and known and tested libraries allowed a **fast and safe development**



Automatic Change detection

Data received from MQTT/BE → Internal variables update → Graph update

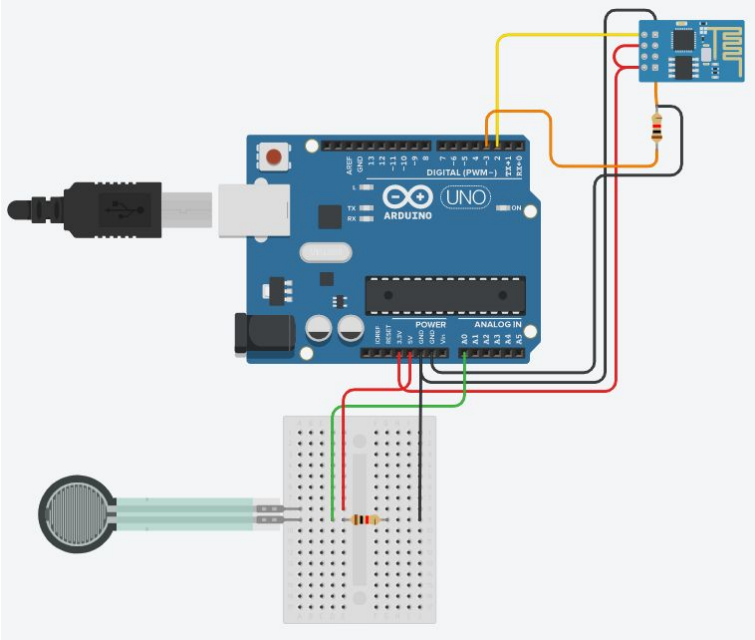
Broker



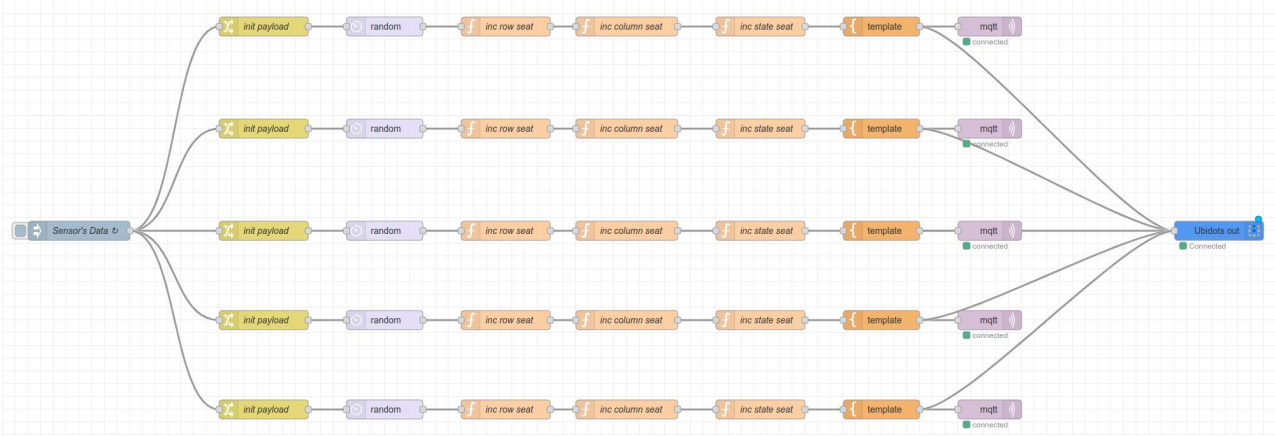
The Broker server has been developed using Node.js using the **MQTT** protocol and the **Aedes** barebone which is responsible to manage the connections between the instances of the system (publishers/subscribers) and the machine related system calls (port openers both as **websocket or http protocol**).

Sensors

The hardware we propose to use consists of an **Arduino UNO**, a **Force Pressure Sensor** and a chip **ESP8266** with integrated WiFi and full support for the TCP/IP protocol.
Thanks to this configuration, you will be able to capture the seat status change and send the information to the broker.

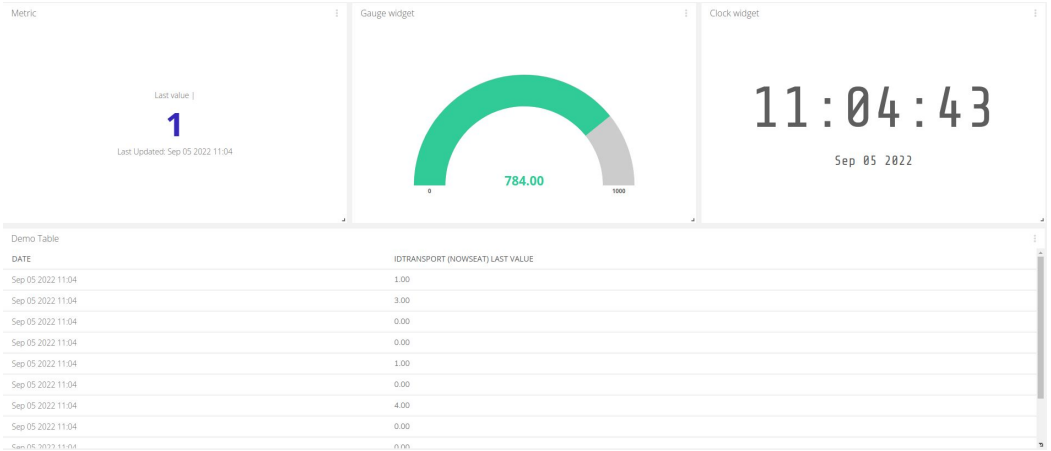


Sensors with Node-RED



Through this Node-Red implementation we **simulated** the sensors data. Initially we set the Topic / Id of the transport, after we **randomly** generate the JSON payload, then we send it via MQTT both to the **Broker** and **Ubidots** with a short delay.

Ubidots



Ubidots allows us to check the **state** of the provided sensors and the related **metrics**.
The dashboard is very intuitive and user friendly.
It is useful to **keep the control over the data** and it helps also to maintain **consistency** between the WebApp and the Sensors.

Demo

Conclusions

Through this project and this implementation we have achieved the goal we set ourselves, which is to satisfy end users.

However, we want to offer a starting point for the integration of this solution within the systems already present at AMT

It is possible to customize the WebApp to take the values related to the stop through GET parameters. In this way it would be easy to integrate an icon inside the AMT mobile app (which already allows the visualization of vehicles arriving at a stop) corresponding to a link connected to the WebApp containing the vehicle code. Obviously this would be possible after installing the sensors and the Broker as prepared by us.

Thanks

GitHub: https://github.com/ChriStingo/IOT-FinalProject

Christian Stingone, Alessio Drago, Alex Di Stefano