

AIR QUALITY MONITORING

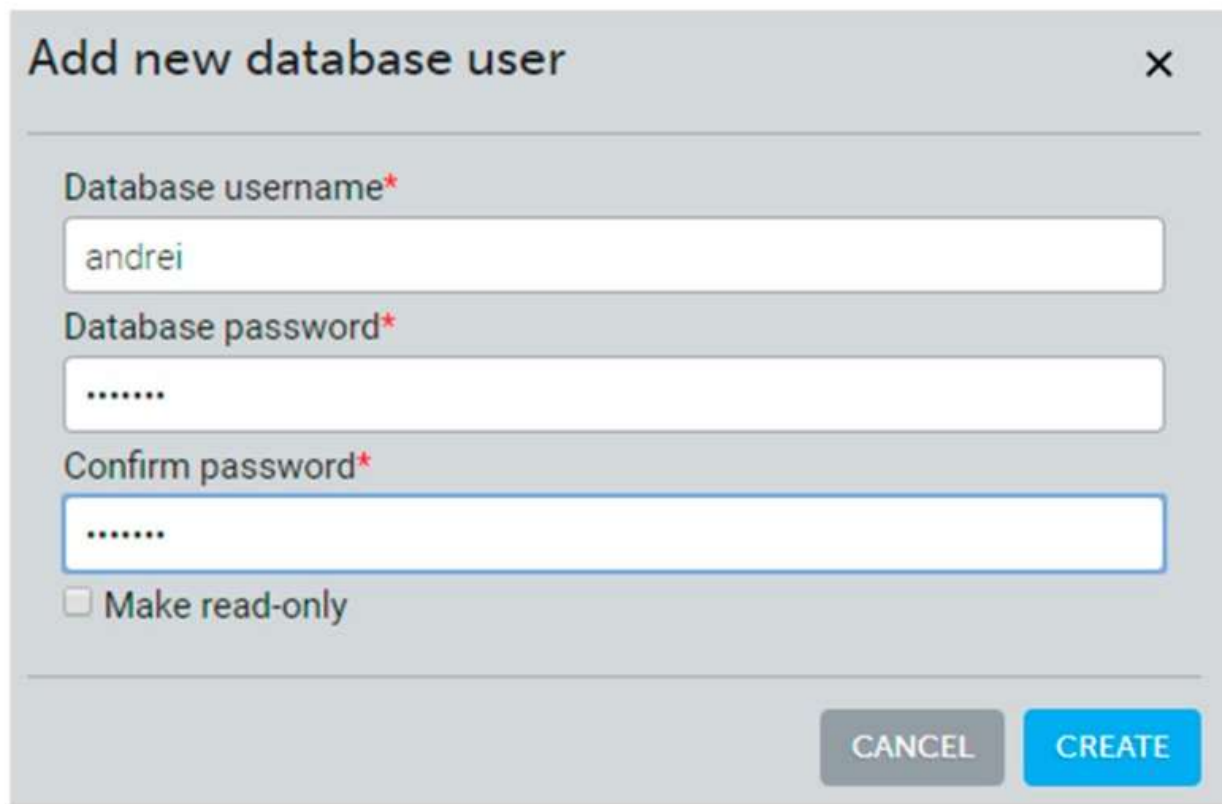
Phase 4: Development Part 2

- ❖ In this part you will continue building your project.
- ❖ Continue building the project by developing the data-sharing platform.
- ❖ Use web development technologies (e.g., HTML, CSS, JavaScript) to create a platform that displays real-time air quality data.
- ❖ Design the platform to receive and display air quality data sent by the IoT devices



Data Persistence and Server-Side NoSQL Database Configuration


The MongoDB database is hosted by the mLab platform through Amazon web services. mLab offers a sandbox with a 500 MB MongoDB instance. Adding a database user is done from the mLab platform through the add database user form storage of collected data from the IoTP4mSCp IoT gateway is performed using a non-only SQL (NoSQL) MongoDB cloud database. Using a NoSQL database under this project is ideal for storing samples recorded by stations and processing them to be distributed through MapReduce .



The screenshot shows a web form titled "Add new database user" with a close button (X) in the top right corner. The form contains three input fields: "Database username*" with the value "andrei", "Database password*" with masked characters "*****", and "Confirm password*" also with "*****". Below these fields is a checkbox labeled "Make read-only" which is currently unchecked. At the bottom right of the form are two buttons: "CANCEL" and "CREATE".


Field	Value
Database username*	andrei
Database password*	*****
Confirm password*	*****
Make read-only	<input type="checkbox"/>

The persistence of the data recorded by the stations in the MongoDB database is carried out through two collections: Stations—in which data is stored on the stations (their name, unique identifier, geographical position), and probes—a collection in which the data persistence is recorded by an IoT gateway development board station (air quality level, detected Wi-Fi networks, and their signal). Also, the IoT4mSCp solution has web interfaces, which implies the existence of users who administer the stations. These are stored in the users' collection in JSON format



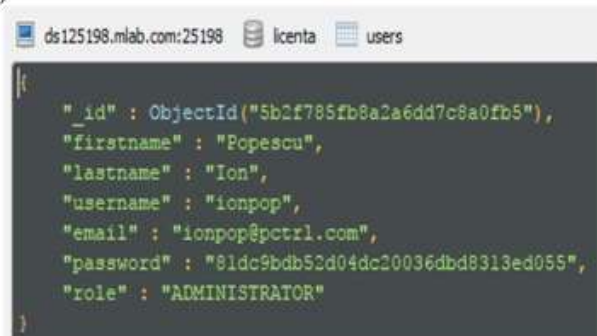
```
ds125198.mlab.com:25198  icenta  probes
{
  "_id" : ObjectId("5b2f6f81b8a2a6dd7c895a70"),
  "stationIdentifier" : "AE:22:15:25:9A:4F",
  "wifi" : [
    {
      "bssid" : "04:8D:38:F6:0E:0C",
      "signal_level" : -47
    },
    {
      "bssid" : "1C:AF:F7:25:73:FE",
      "signal_level" : -60
    },
    {
      "bssid" : "AE:22:15:16:18:ED",
      "signal_level" : -93
    }
  ],
  "temperature" : 22.3,
  "pressure" : 1015,
  "humidity" : 54,
  "co_ppm" : 0,
  "co2_ppm" : 384,
  "ch4_ppb" : 1830,
  "nh4_ppm" : 202,
  "toluene_ppm" : 230
}
```

(a)



```
ds125198.mlab.com:25198  icenta  stations
{
  "_id" : ObjectId("5b2f7673b8a2a6dd7c89e8dd"),
  "stationIdentifier" : "AE:22:15:25:9A:4F",
  "name" : "Bucharest-Pt-a-Victoriei-1",
  "position" : {
    "lat" : 44.452336,
    "lng" : 26.086321
  }
}
```

(b)



```
ds125198.mlab.com:25198  icenta  users
{
  "_id" : ObjectId("5b2f785fb8a2a6dd7c8a0fb5"),
  "firstname" : "Popescu",
  "lastname" : "Ion",
  "username" : "ionpop",
  "email" : "ionpop@pctrl.com",
  "password" : "81dc9bdb52d04dc20036dbd8313ed055",
  "role" : "ADMINISTRATOR"
}
```

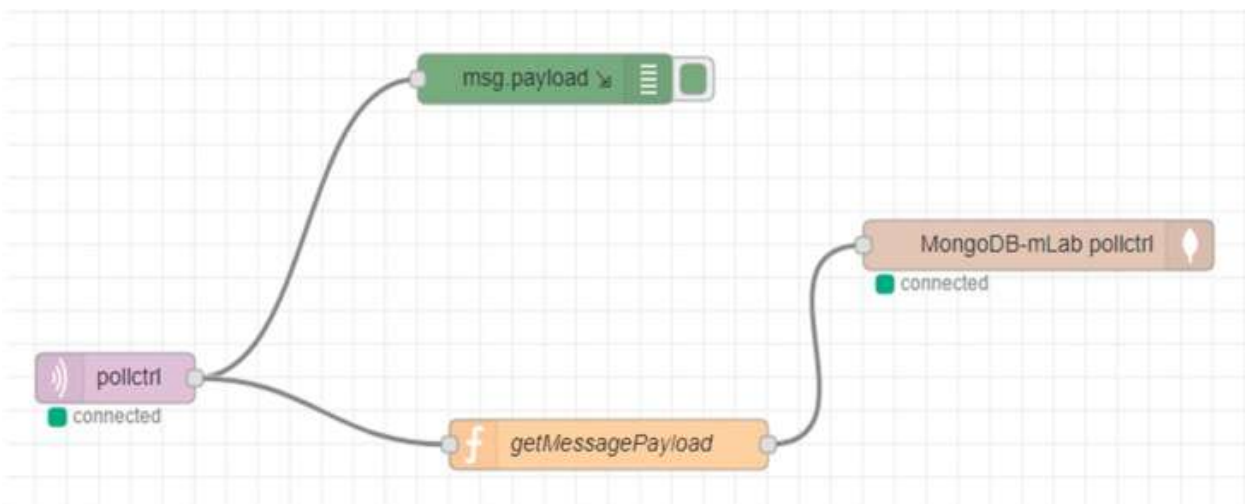
(c)

Node-RED Framework

Node-RED is an IBM-based, flow-based programming tool for connecting hardware, APIs, and online services. It is built on Node.js and uses its non-blocking and event-driven model, making it ideal for running on limited-cloud or cloud-based devices.

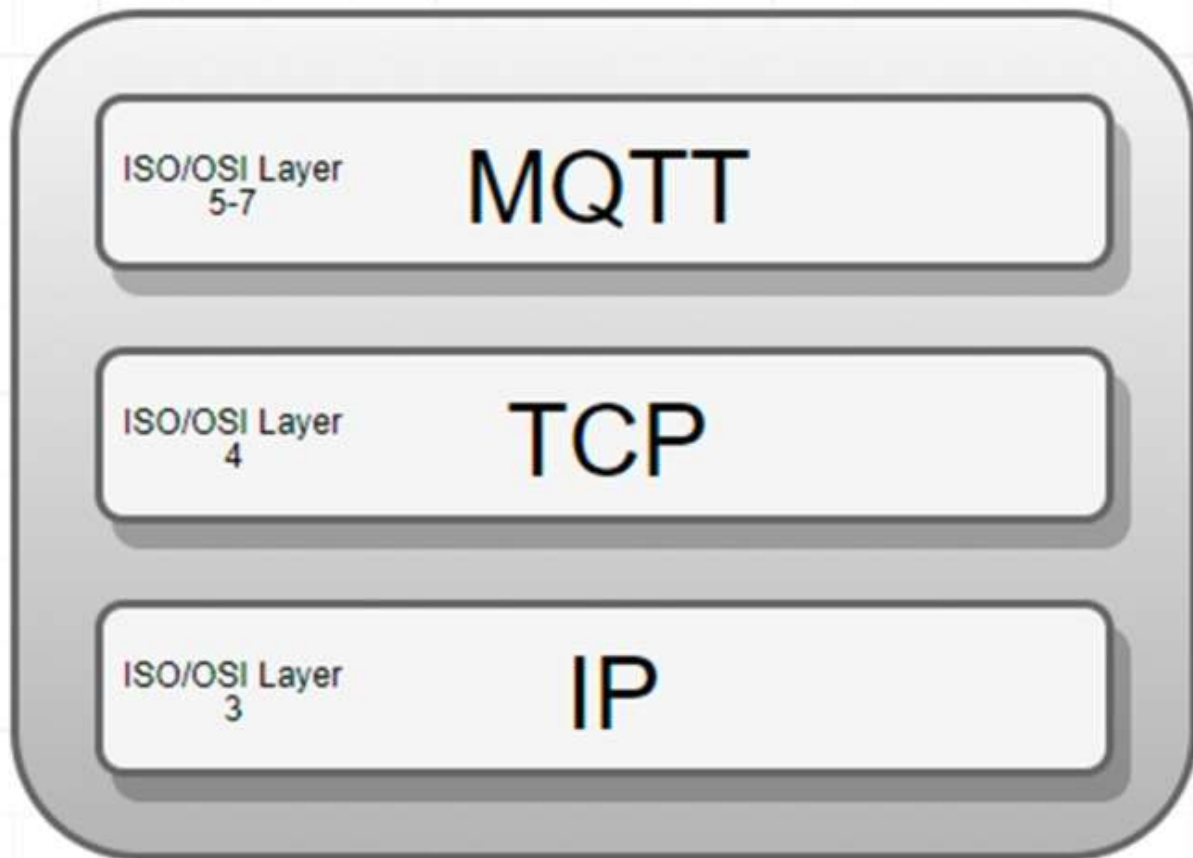
This flow contains the following nodes:

- ❖ MQTT output node (connected to mosquitto broker from AWS EC2).
- ❖ JavaScript function node (collects and validates the payload from the MQTT message).
- ❖ MongoDB input node (connected to mLab MongoDB instance).
- ❖ Debug node (logs the raw MQTT message).



IoT Communications Protocols to the IoT Cloud

As IoT middleware and communications protocols between the IoT gateways and the IoT Cloud for the solution, MQTT(s) and HTTP(s)—REST-API are used.



For implementation of the message processing, the authors have defined a MQTTClient class that contains a Mqtt Paho client—by Eclipse—and implements the MqttCallback interface. This interface defines the messageArrived() method, in which the current configuration of the station is updated.

```

45  @Override
46  public void messageArrived(String topic, MqttMessage message) throws Exception {
47      if(MQTTConstants.MQTT_CONFIG_LOCAL.equals(topic) ||
48          MQTTConstants.MQTT_CONFIG_ALL.equals(topic)){
49          final PctrFacade pctrFacade = PctrFacade.getInstance();
50          Gson gson = new Gson();
51          try{
52              StationConfig stationConfig = gson.fromJson
53                  (new String(message.getPayload(), charsetName: "UTF-8"), StationConfig.class);
54              pctrFacade.prepare(stationConfig);
55              new Thread(pctrFacade).start();
56          }catch (JsonSyntaxException jse){
57              jse.printStackTrace();
58          }
59      }
60  }

```

RESTful Services' Implementation

RESTful services are implemented through the classes: ProbeController, StationsController, and UsersController. They are annotated with `@RestController` and `@RequestMapping` specifying the URI of the resource. Adding a polluting sample is done through a POST HTTP request to `"/api/probes"`.

```

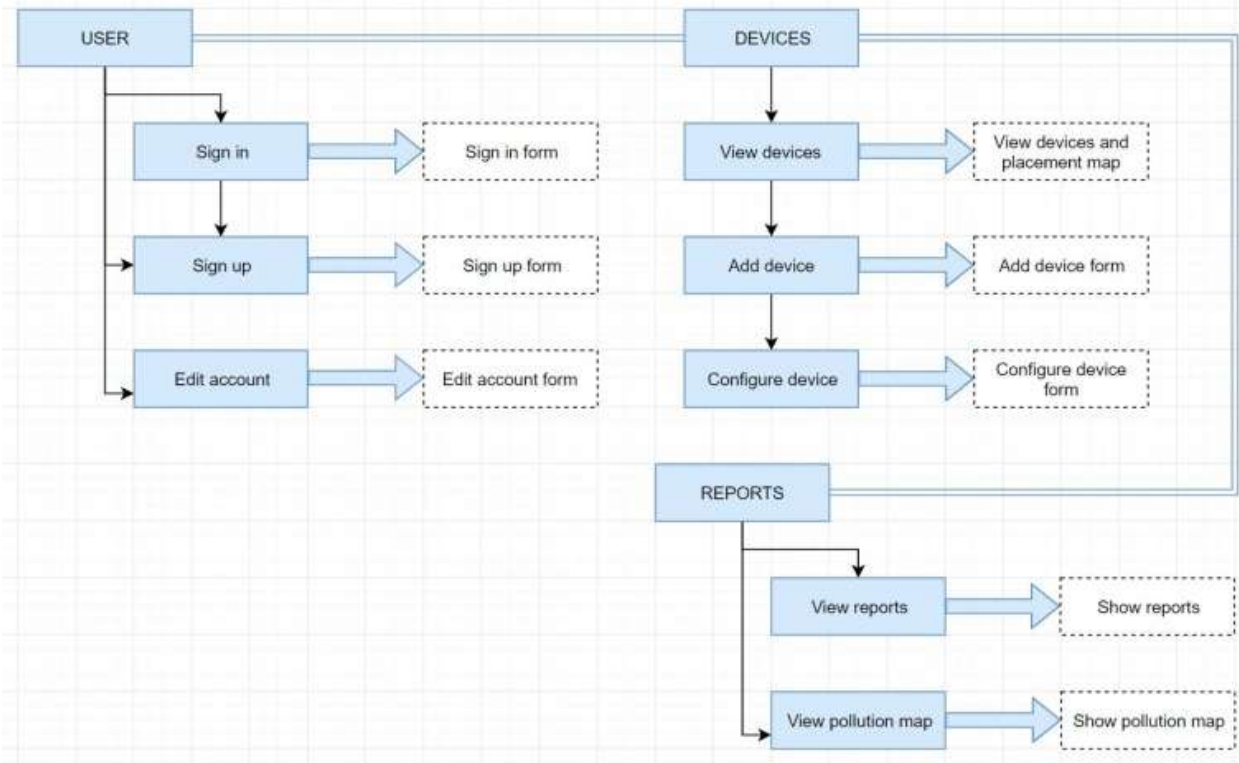
@PostMapping(value = {"", "/"})
public ResponseEntity save(@RequestBody Probe probe) {
    if (probe == null) {
        return new ResponseEntity(HttpStatus.BAD_REQUEST);
    }
    Probe savedProbe = null;
    try {
        savedProbe = probeService.save(probe);
    } catch (Exception ex) {
        LOG.error(ex.getMessage());
    }
    if (savedProbe != null) {
        return new ResponseEntity(savedProbe, HttpStatus.CREATED);
    } else {
        return new ResponseEntity(HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
}

```

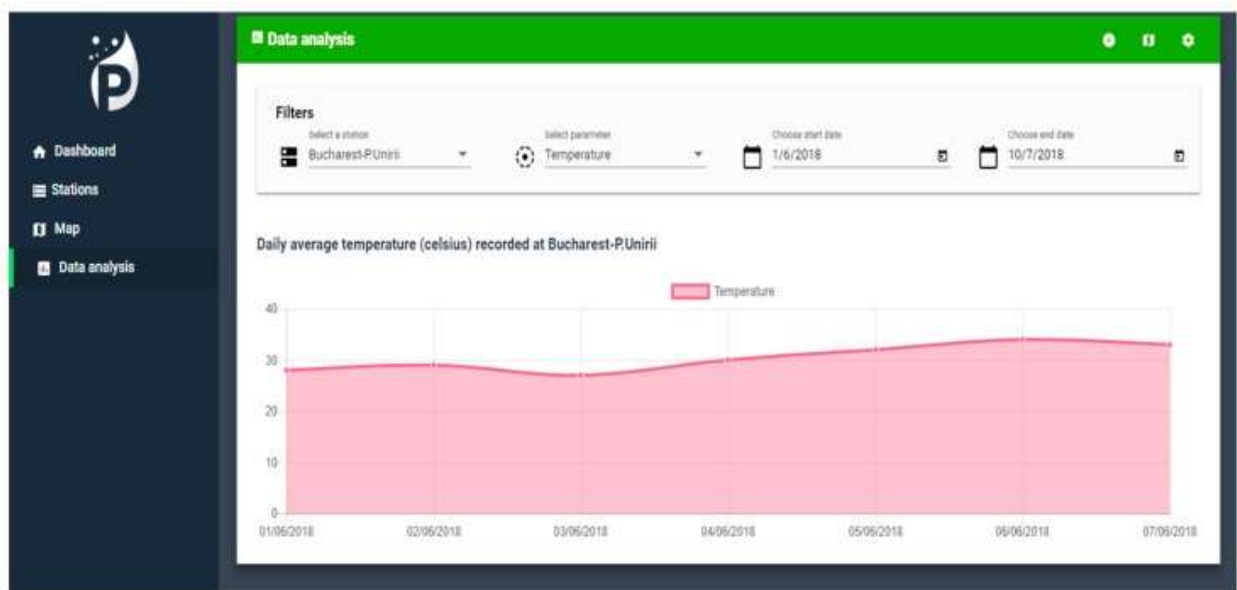
Front-End Implementation

The IoT4mSCp solution's front-end component is a single-page application (SPA) built into the Angular 5.2-based framework and the angular material library—used to use predefined components, such as dialog windows, buttons, and tables. The project was created using the command line interface (CLI), which facilitates the generation of components, services, and modules.

The user interface includes the following sections: User area—includes authentication, account creation and profile modification, and station area—fetches information about IoT Gateway stations, adding/configuring/deleting them, and the pollution sample area, where detailed reports are presented based on recorded samples



shows the real-time monitoring of the indicators reported from the IoT gateways distributed within the city.



Since the solution's frontend component is a single-page application, it is necessary to map the requests from Spring boot to Angular. For example, when receiving an HTTP GET request of the resource identified by localhost, 8080/dashboard, the Spring boot framework looks for a class controller that can handle that request. Therefore, interpretation of the applications by the P.M.A frontend component is done by redirecting them using the following View Controller class.

```
@Controller
public class ViewController {

    @RequestMapping({"/dashboard", "/login", "/stations"})
    public String index(){
        return "forward:/index.html";
    }

}
```

Conclusion

- ❖ Seasonal influence on the cold weather outside and shopping periods for Christmas, road traffic but also thermo-central consumption of gas for heating households. Also, in 2018, the transit pipes for hot water from the city were more damaged than in 2016 and this cannot be done without investments.
- ❖ Regulations and laws changes—in 2018, it was much easier to bring very old cars—taxes were lower—that produce higher pollution than in 2016 to 2017.
- ❖ The number of powerful Wi-Fi routers increased in 2018 compared with 2017 and also the decibels produced as noise pollution by Wi-Fi access points.
- ❖ Price of gas within gas distributors' "chains/stations" influenced the traffic.
- ❖ Other factors that may influence the pollution but for which a direct relation cannot be confirmed (covariance between the factor and the pollution level.)

