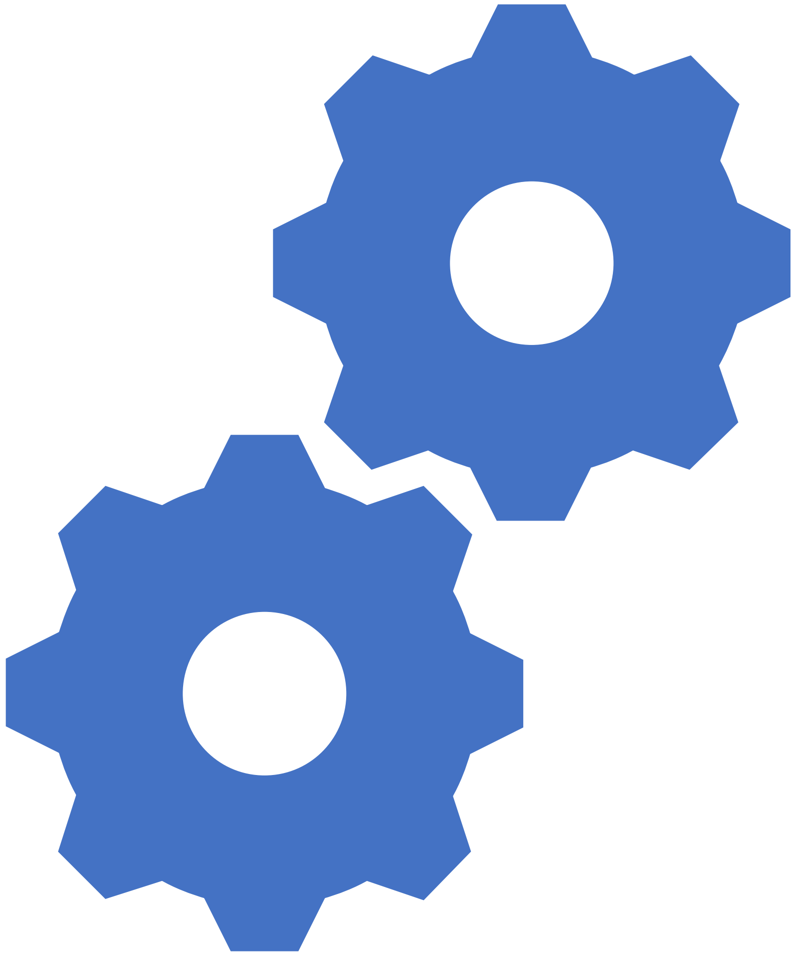
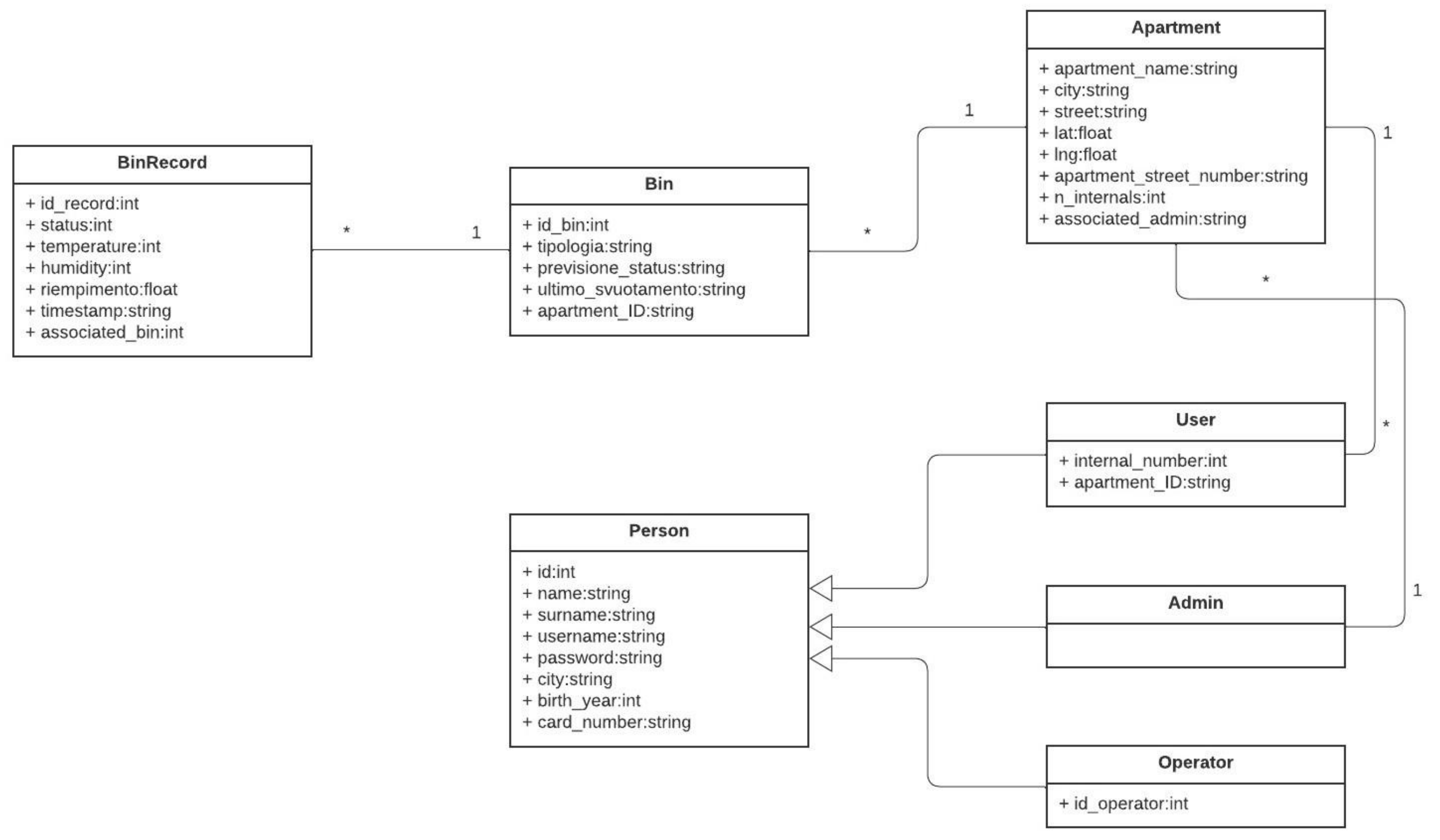
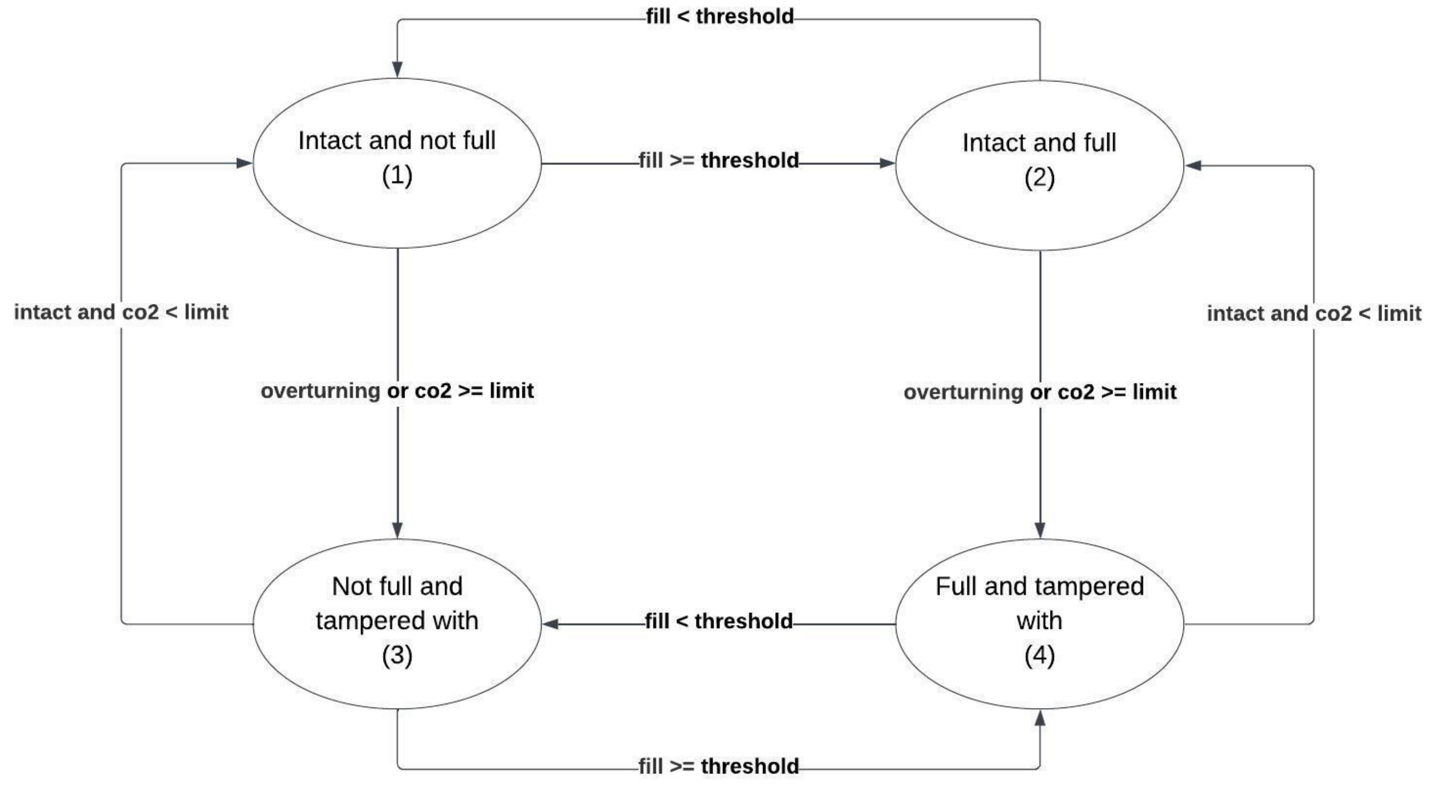
SYSTEM DESIGN

Alessia Saporita, Vincenzo Lapadula, Michele Giarletta



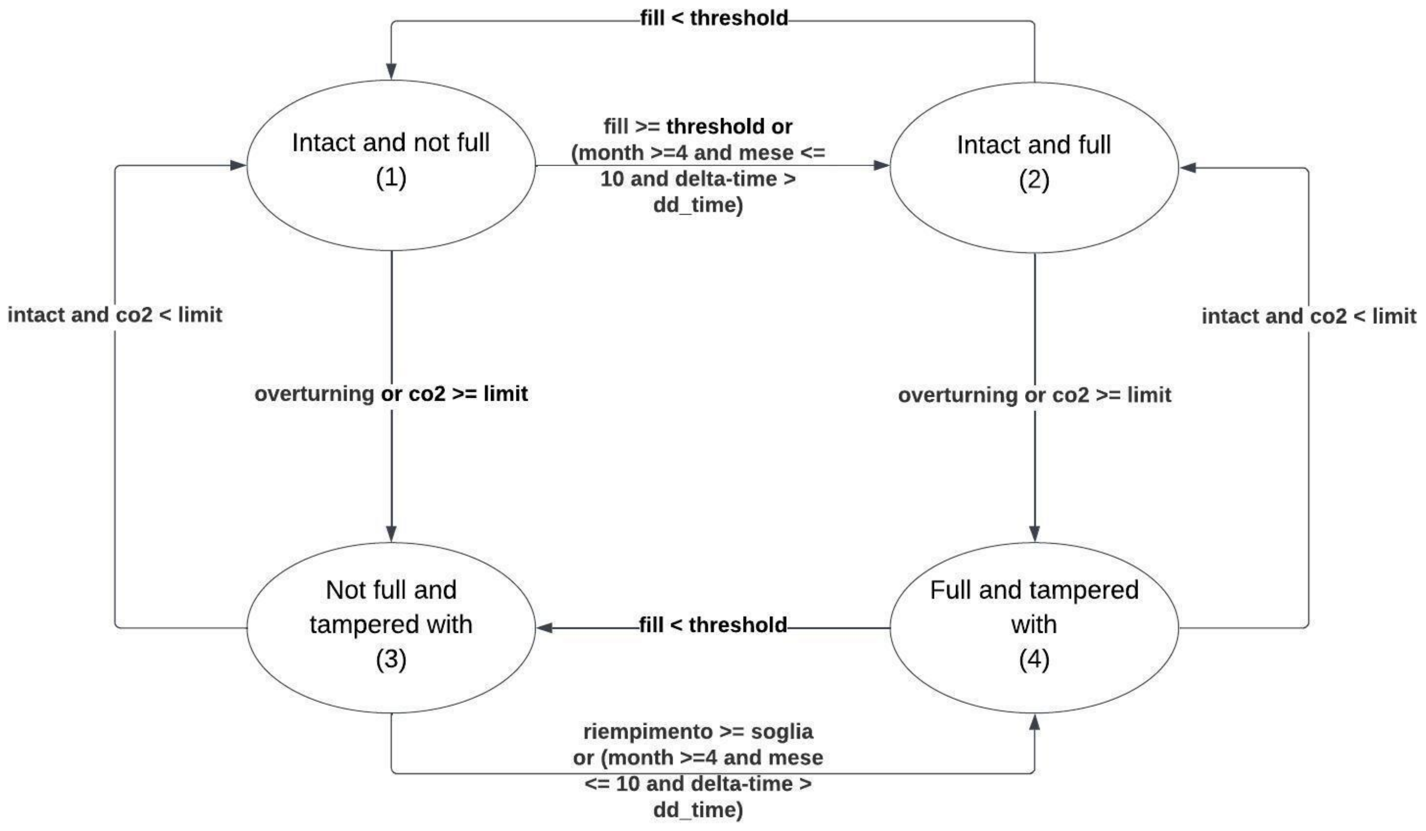
UML Class


BinRecord
+ id_record:int
+ status:int
+ temperature:int
+ humidity:int
+ riempimento:float
+ timestamp:string
+ associated_bin:int
*
1
Bin
+ id_bin:int
+ tipologia:string
+ previsione_status:string
+ ultimo_svuotamento:string
+ apartment_ID:string
*
1
Apartment
+ apartment_name:string
+ city:string
+ street:string
+ lat:float
+ lng:float
+ apartment_street_number:string
+ n_internals:int
+ associated_admin:string
1
*
*
User
+ internal_number:int
+ apartment_ID:string
Person
+ id:int
+ name:string
+ surname:string
+ username:string
+ password:string
+ city:string
+ birth_year:int
+ card_number:string
Admin
1
Operator
+ id_operator:int



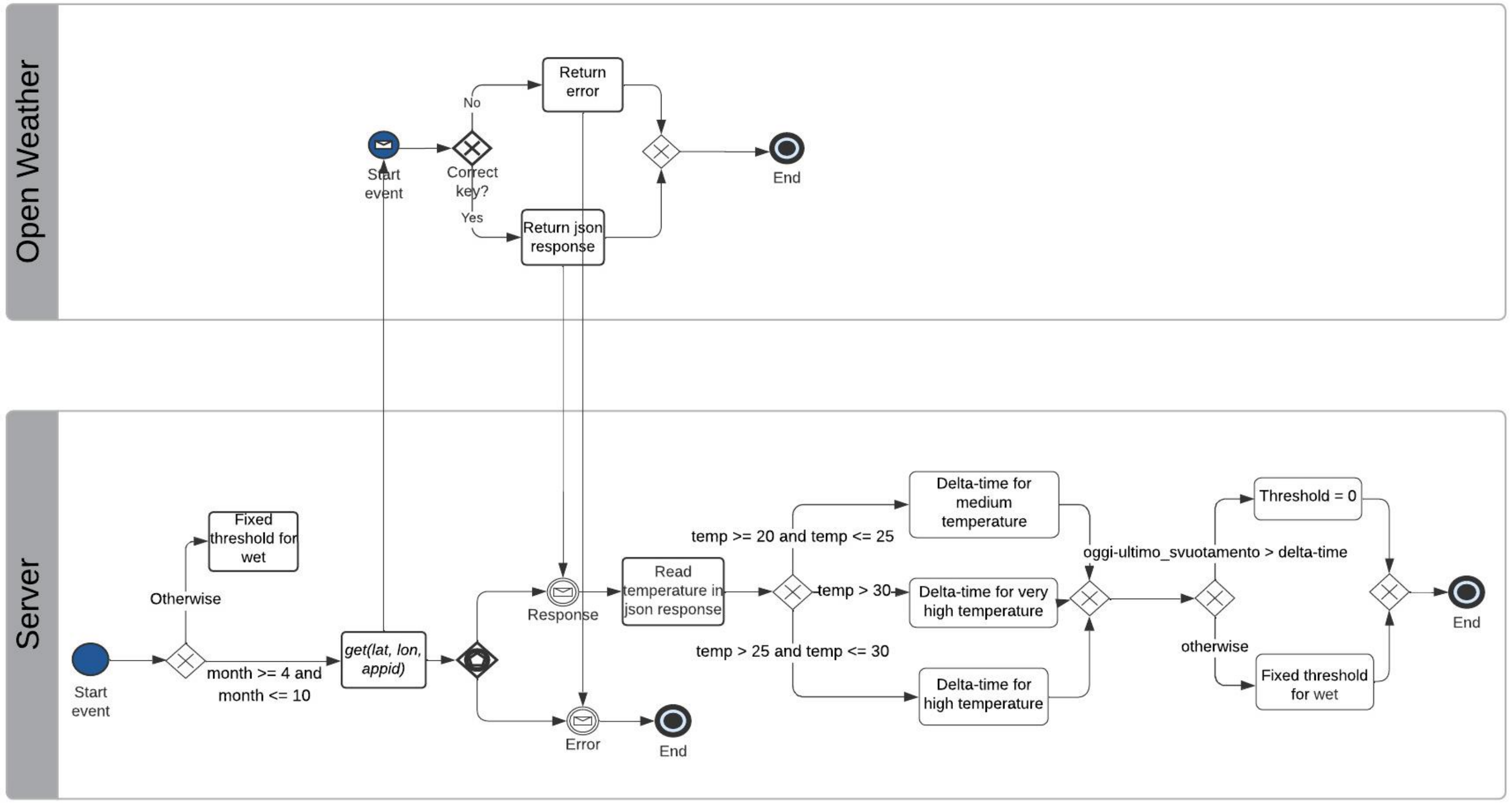
Finite state machine

Finite state machine for organic waste


fill < threshold
Intact and not full (1)
fill >= threshold or (month >=4 and mese <= 10 and delta-time > dd_time)
Intact and full (2)
intact and co2 < limit
overturning or co2 >= limit
overturning or co2 >= limit
intact and co2 < limit
Not full and tampered with (3)
fill < threshold
Full and tampered with (4)
riempimento >= soglia or (month >=4 and mese <= 10 and delta-time > dd_time)

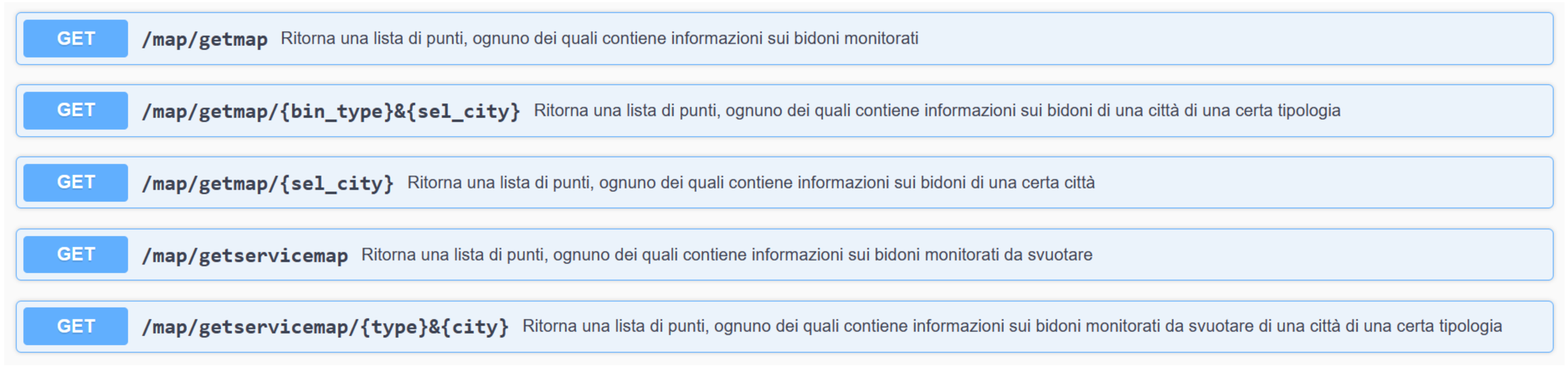
delta_time = interval of days since the last emptying

dd_time = number of days that can elapse since the last emptying. Depending on the temperature it is 5 days for average temperatures, 3 for high temperatures and 2 for very high temperatures.



Get organic waste threshold

API map



GET `/map/getmap` Ritorna una lista di punti, ognuno dei quali contiene informazioni sui bidoni monitorati

GET `/map/getmap/{bin_type}&{sel_city}` Ritorna una lista di punti, ognuno dei quali contiene informazioni sui bidoni di una città di una certa tipologia

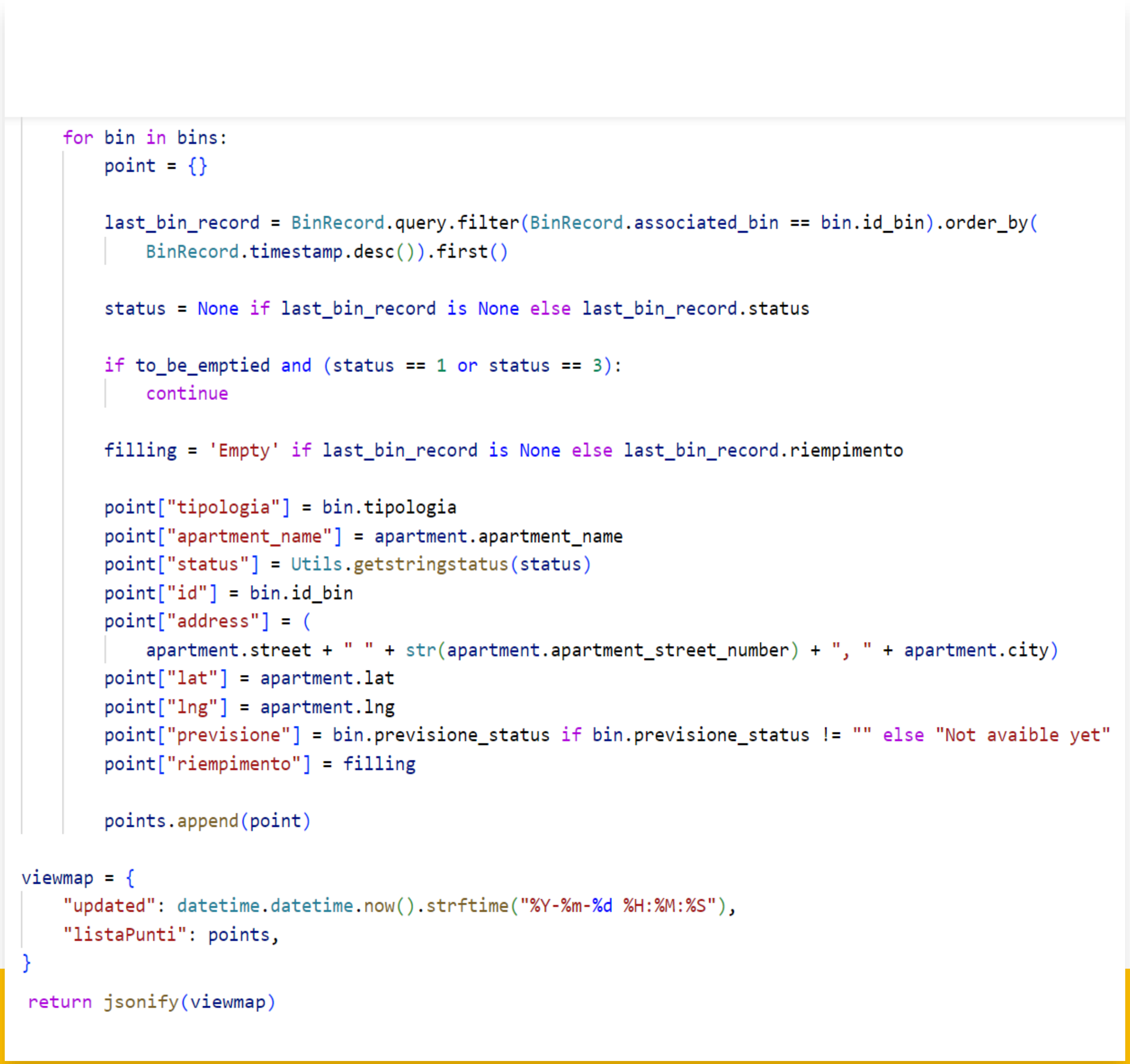
GET `/map/getmap/{sel_city}` Ritorna una lista di punti, ognuno dei quali contiene informazioni sui bidoni di una certa città

GET `/map/getservicemap` Ritorna una lista di punti, ognuno dei quali contiene informazioni sui bidoni monitorati da svuotare

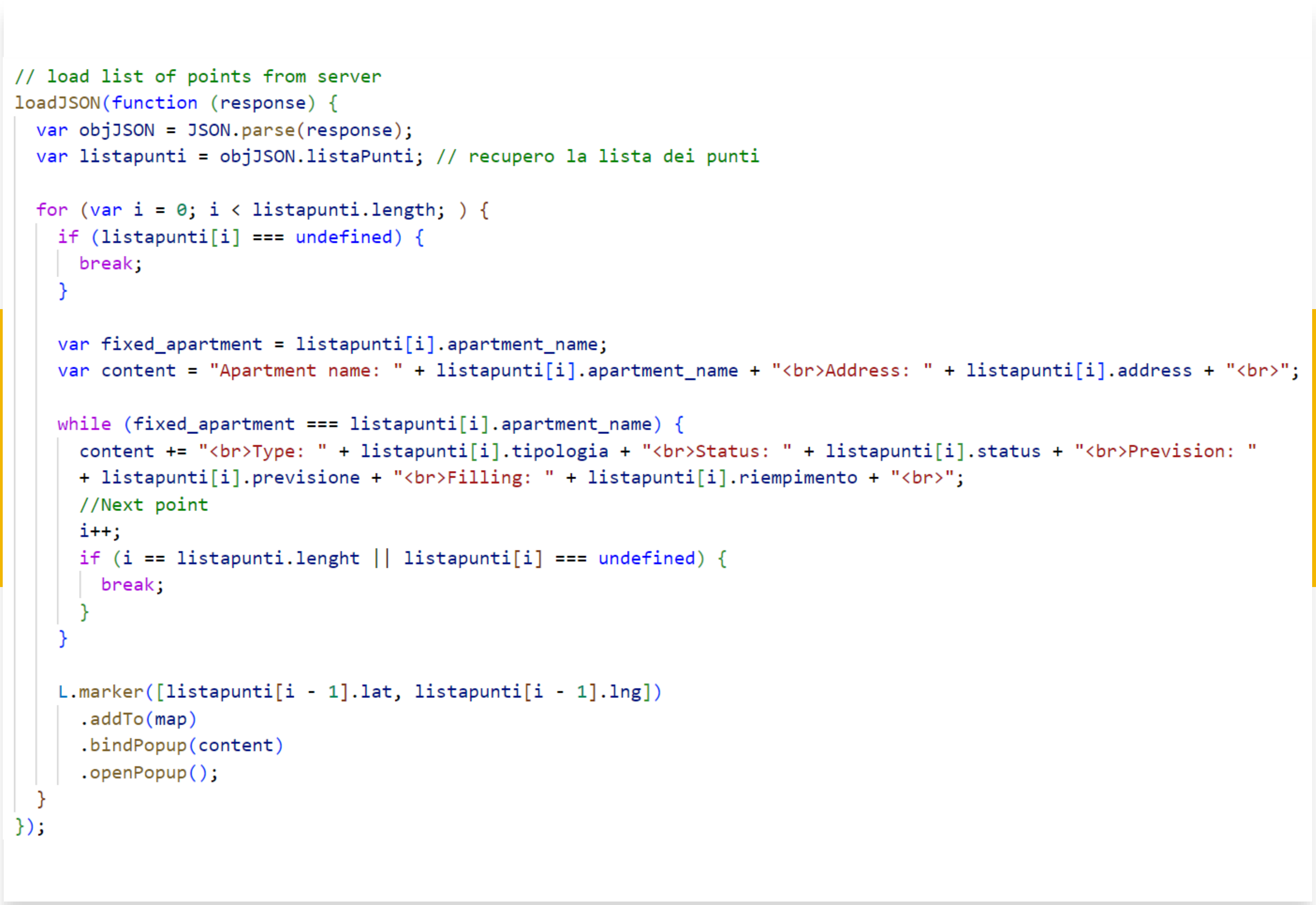
GET `/map/getservicemap/{type}&{city}` Ritorna una lista di punti, ognuno dei quali contiene informazioni sui bidoni monitorati da svuotare di una città di una certa tipologia

Map

- We create a list of points
- Each point corresponds to a bin
- Each point is described by a dictionary that contains the information about the bin (typology, status, address, etc…)



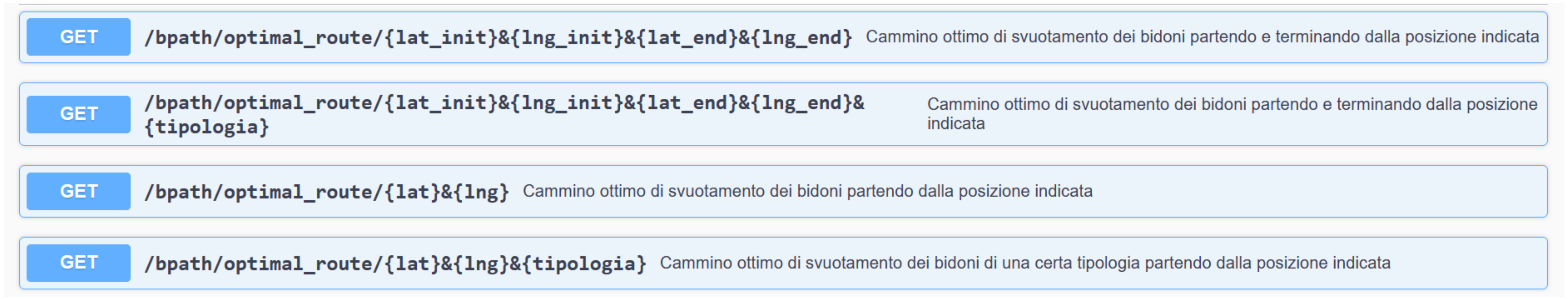
```
    for bin in bins:
        point = {}

        last_bin_record = BinRecord.query.filter(BinRecord.associated_bin == bin.id_bin).order_by(
            BinRecord.timestamp.desc()).first()

        status = None if last_bin_record is None else last_bin_record.status

        if to_be_emptied and (status == 1 or status == 3):
            continue

        filling = 'Empty' if last_bin_record is None else last_bin_record.riempimento

        point["tipologia"] = bin.tipologia
        point["apartment_name"] = apartment.apartment_name
        point["status"] = Utils.getstringstatus(status)
        point["id"] = bin.id_bin
        point["address"] = (
            apartment.street + " " + str(apartment.apartment_street_number) + ", " + apartment.city)
        point["lat"] = apartment.lat
        point["lng"] = apartment.lng
        point["previsione"] = bin.previsione_status if bin.previsione_status != "" else "Not avaible yet"
        point["riempimento"] = filling

        points.append(point)

viewmap = {
    "updated": datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
    "listaPunti": points,
}

return jsonify(viewmap)
```



```
// load list of points from server
loadJSON(function (response) {
  var objJSON = JSON.parse(response);
  var listapunti = objJSON.listaPunti; // recupero la lista dei punti

  for (var i = 0; i < listapunti.length; ) {
    if (listapunti[i] === undefined) {
      break;
    }

    var fixed_apartment = listapunti[i].apartment_name;
    var content = "Apartment name: " + listapunti[i].apartment_name + "<br>Address: " + listapunti[i].address + "<br>";

    while (fixed_apartment === listapunti[i].apartment_name) {
      content += "<br>Type: " + listapunti[i].tipologia + "<br>Status: " + listapunti[i].status + "<br>Prevision: "
      + listapunti[i].previsione + "<br>Filling: " + listapunti[i].riempimento + "<br>";
      //Next point
      i++;
      if (i == listapunti.lenght || listapunti[i] === undefined) {
        break;
      }
    }

    L.marker([listapunti[i - 1].lat, listapunti[i - 1].lng])
      .addTo(map)
      .bindPopup(content)
      .openPopup();
  }
});
```

Map

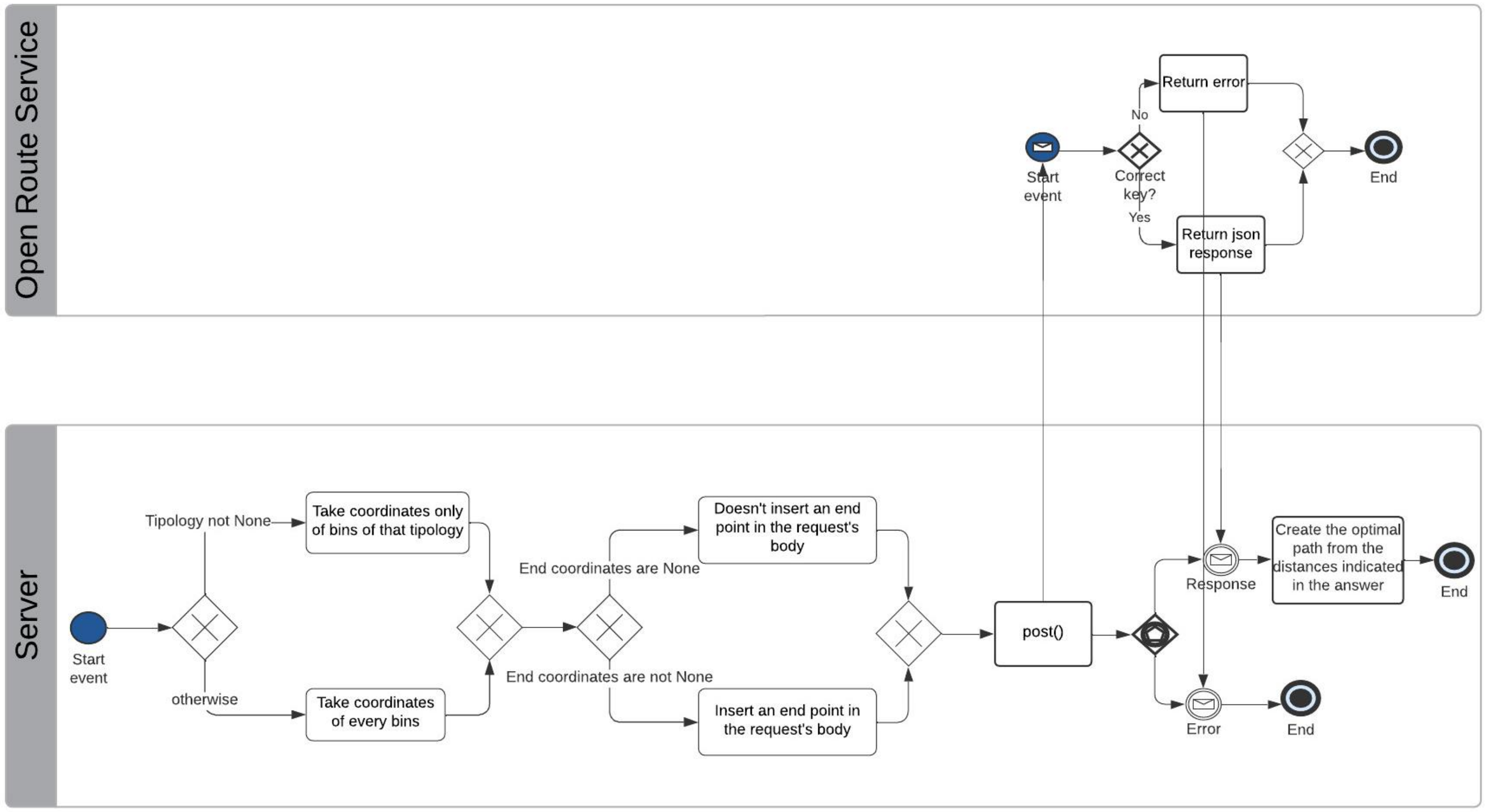
API Best Path


GET /bpath/optimal_route/{lat_init}&{lng_init}&{lat_end}&{lng_end} Cammino ottimo di svuotamento dei bidoni partendo e terminando dalla posizione indicata

GET /bpath/optimal_route/{lat_init}&{lng_init}&{lat_end}&{lng_end}&{tipologia} Cammino ottimo di svuotamento dei bidoni partendo e terminando dalla posizione indicata

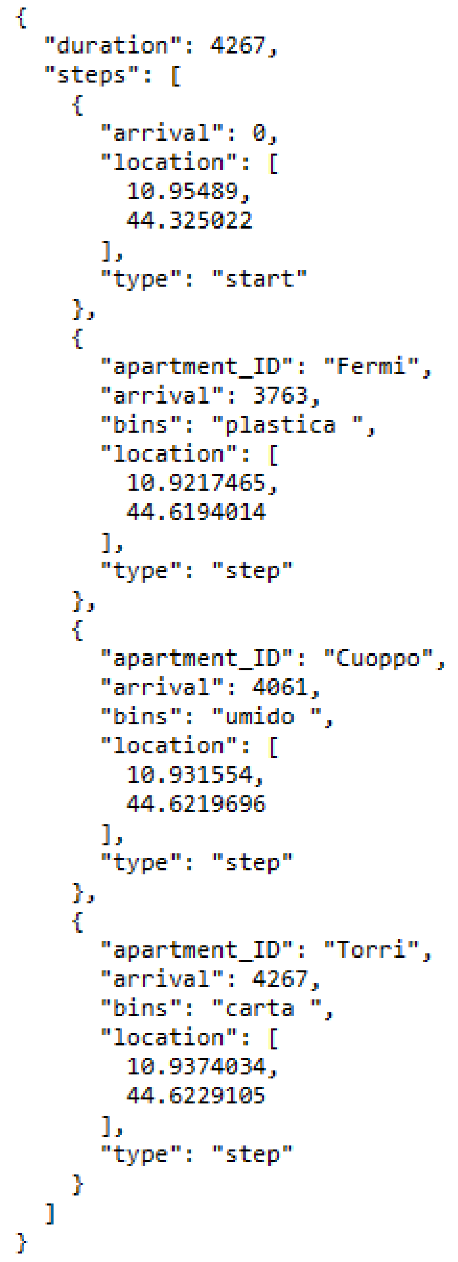
GET /bpath/optimal_route/{lat}&{lng} Cammino ottimo di svuotamento dei bidoni partendo dalla posizione indicata

GET /bpath/optimal_route/{lat}&{lng}&{tipologia} Cammino ottimo di svuotamento dei bidoni di una certa tipologia partendo dalla posizione indicata


Open Route Service
Start event
Correct key?
No
Yes
Return error
Return json response
End
Server
Start event
Tipology not None
otherwise
Take coordinates only of bins of that tipology
Take coordinates of every bins
End coordinates are None
End coordinates are not None
Doesn't insert an end point in the request's body
Insert an end point in the request's body
post()
Response
Error
Create the optimal path from the distances indicated in the answer
End
End

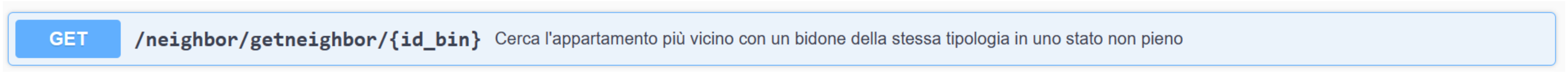
Optimal route

JSON Response

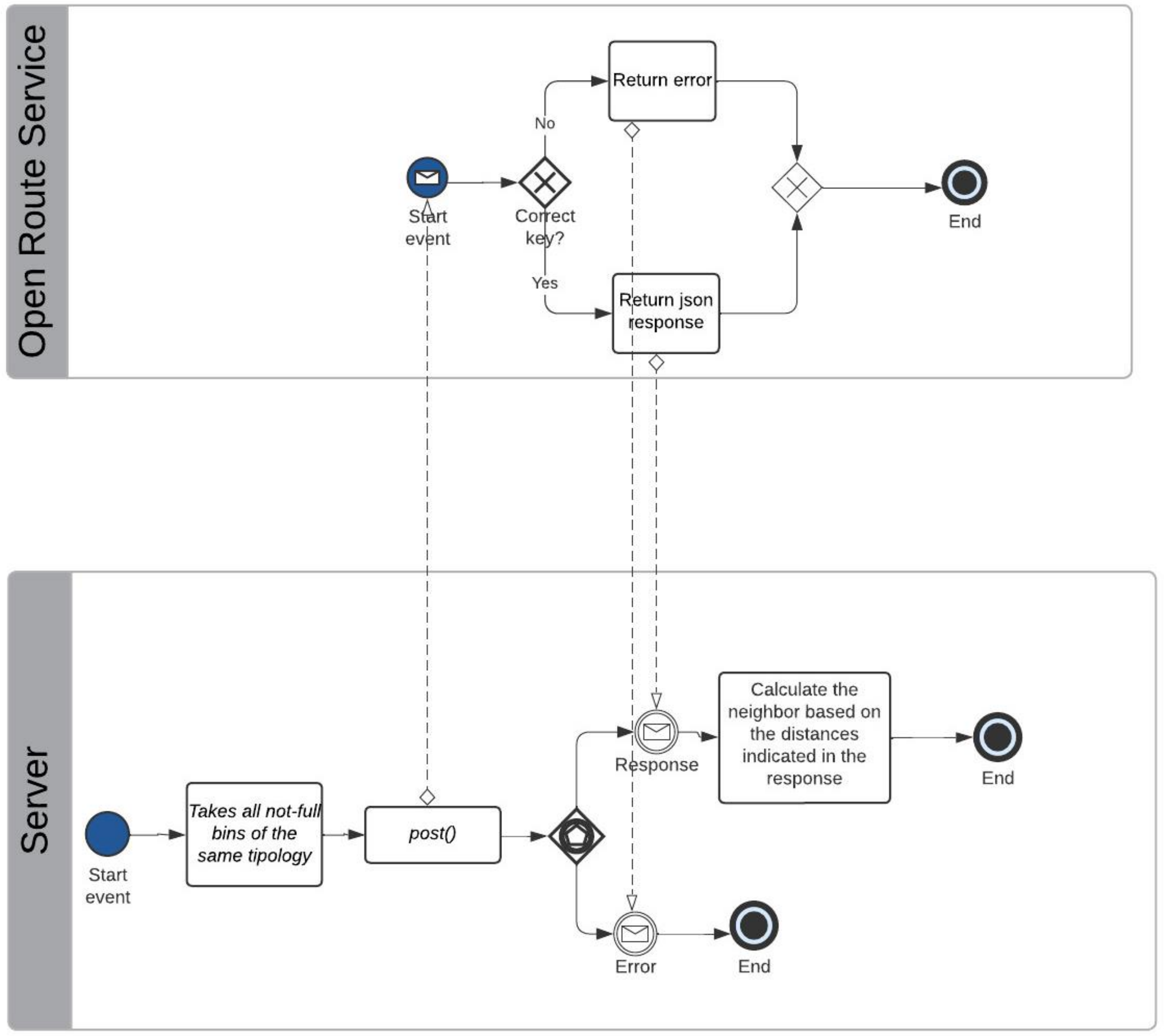


```
{
  "duration": 4267,
  "steps": [
    {
      "arrival": 0,
      "location": [
        10.95489,
        44.325022
      ],
      "type": "start"
    },
    {
      "apartment_ID": "Fermi",
      "arrival": 3763,
      "bins": "plastica ",
      "location": [
        10.9217465,
        44.6194014
      ],
      "type": "step"
    },
    {
      "apartment_ID": "Cuoppo",
      "arrival": 4061,
      "bins": "umido ",
      "location": [
        10.931554,
        44.6219696
      ],
      "type": "step"
    },
    {
      "apartment_ID": "Torri",
      "arrival": 4267,
      "bins": "carta ",
      "location": [
        10.9374034,
        44.6229105
      ],
      "type": "step"
    }
  ]
}
```

API Neighbor

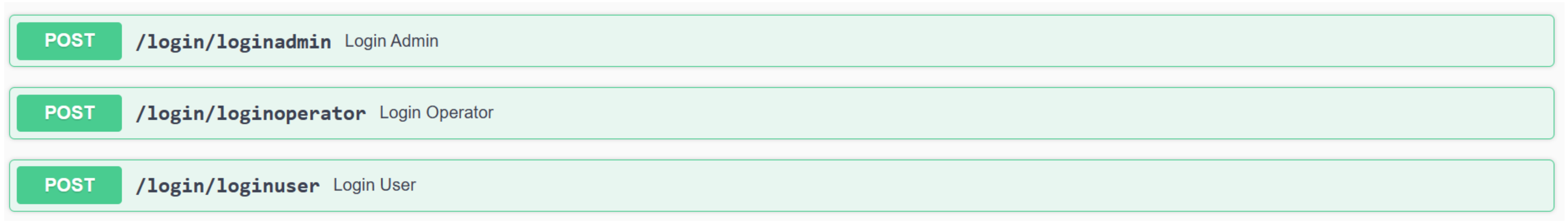


GET `/neighbor/getneighbor/{id_bin}` Cerca l'appartamento più vicino con un bidone della stessa tipologia in uno stato non pieno



Get neighbor

API login



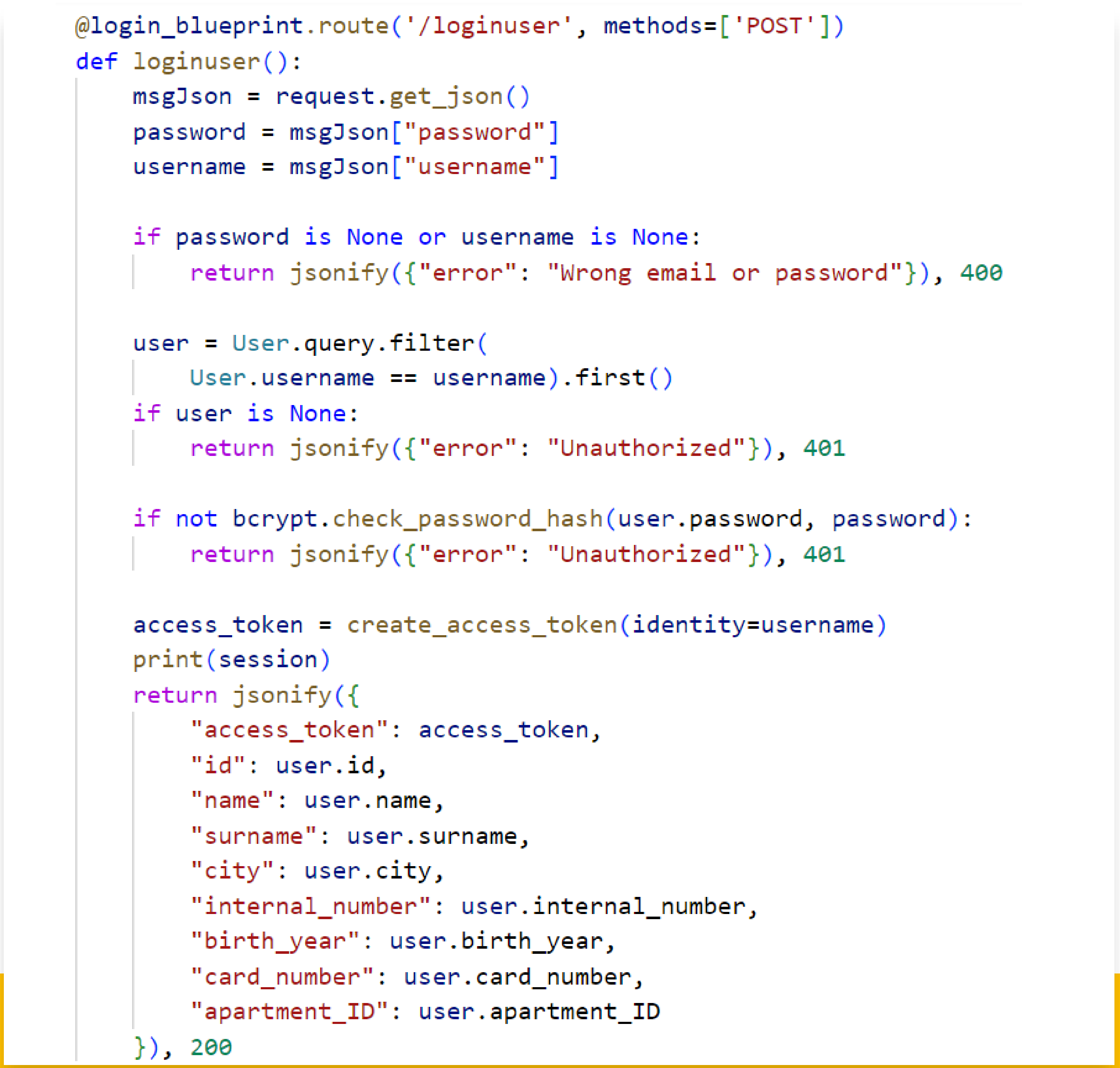
POST /login/loginadmin Login Admin

POST /login/loginoperator Login Operator

POST /login/loginuser Login User

Login

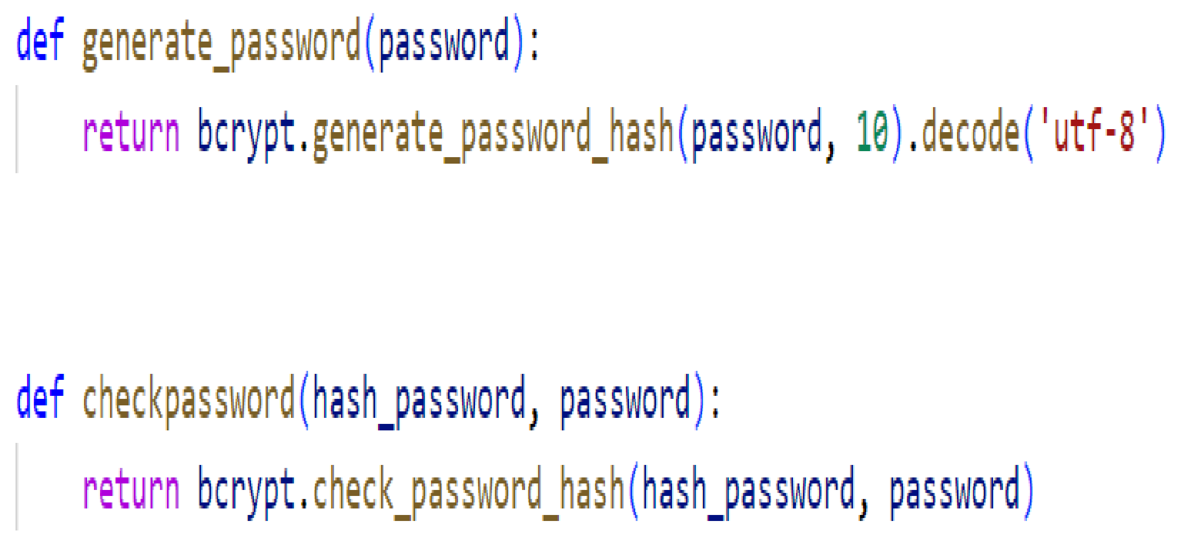
- JWT is an Open standard for creating access tokens between a server and a client.
- *"flask-jwt-extended"* is a flask extension that provides us with the *@jwt_required* decorator, which allows access to the endpoint only after verification of the Token.



```
@login_blueprint.route('/loginuser', methods=['POST'])
def loginuser():
    msgJson = request.get_json()
    password = msgJson["password"]
    username = msgJson["username"]

    if password is None or username is None:
        return jsonify({"error": "Wrong email or password"}), 400

    user = User.query.filter(
        User.username == username).first()
    if user is None:
        return jsonify({"error": "Unauthorized"}), 401

    if not bcrypt.check_password_hash(user.password, password):
        return jsonify({"error": "Unauthorized"}), 401

    access_token = create_access_token(identity=username)
    print(session)
    return jsonify({
        "access_token": access_token,
        "id": user.id,
        "name": user.name,
        "surname": user.surname,
        "city": user.city,
        "internal_number": user.internal_number,
        "birth_year": user.birth_year,
        "card_number": user.card_number,
        "apartment_ID": user.apartment_ID
    }), 200
```

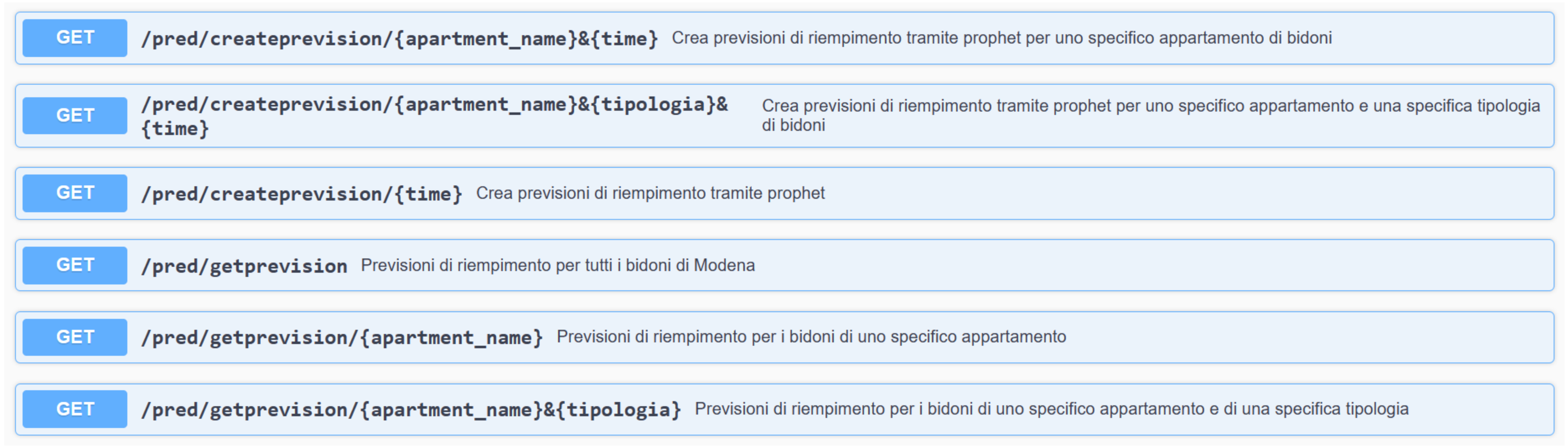
Authentication

- Flask-Bcrypt is a Flask extension that provides bcrypt hashing utilities for your application
- Bcrypt is a hashing algorithm. It takes in a plain text password as an input and returns a hash of that password.
- check_password_hash() tests a password hash against a candidate password
- generate_password_hash() generates a password hash using bcrypt



```
def generate_password(password):
    return bcrypt.generate_password_hash(password, 10).decode('utf-8')


def checkpassword(hash_password, password):
    return bcrypt.check_password_hash(hash_password, password)
```

API Prophet



GET `/pred/createprevision/{apartment_name}&{time}` Crea previsioni di riempimento tramite prophet per uno specifico appartamento di bidoni

GET `/pred/createprevision/{apartment_name}&{tipologia}&{time}` Crea previsioni di riempimento tramite prophet per uno specifico appartamento e una specifica tipologia di bidoni

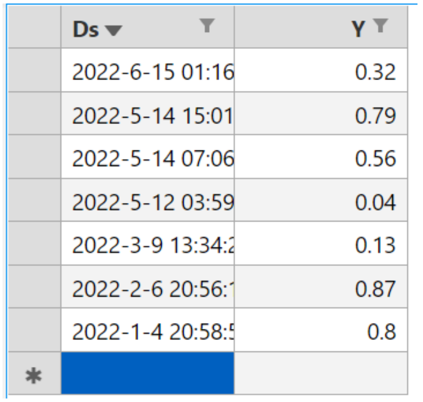
GET `/pred/createprevision/{time}` Crea previsioni di riempimento tramite prophet

GET `/pred/getprevision` Previsioni di riempimento per tutti i bidoni di Modena

GET `/pred/getprevision/{apartment_name}` Previsioni di riempimento per i bidoni di uno specifico appartamento

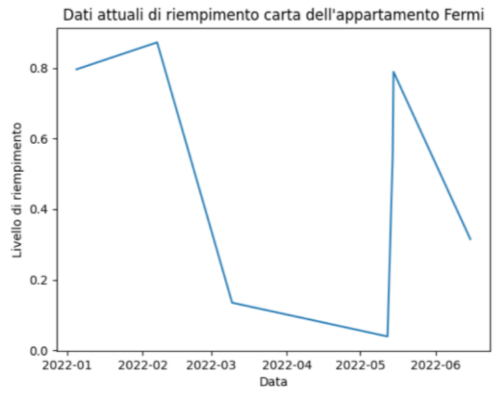
GET `/pred/getprevision/{apartment_name}&{tipologia}` Previsioni di riempimento per i bidoni di uno specifico appartamento e di una specifica tipologia

Facebook Prophet

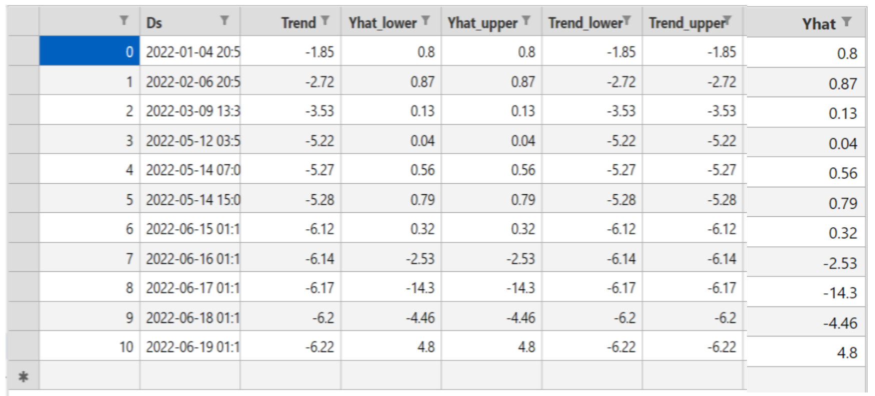


	Ds	Y
	2022-6-15 01:16	0.32
	2022-5-14 15:01	0.79
	2022-5-14 07:06	0.56
	2022-5-12 03:59	0.04
	2022-3-9 13:34:2	0.13
	2022-2-6 20:56:1	0.87
	2022-1-4 20:58:5	0.8
*		

Filling of the paper waste of the Fermi apartment



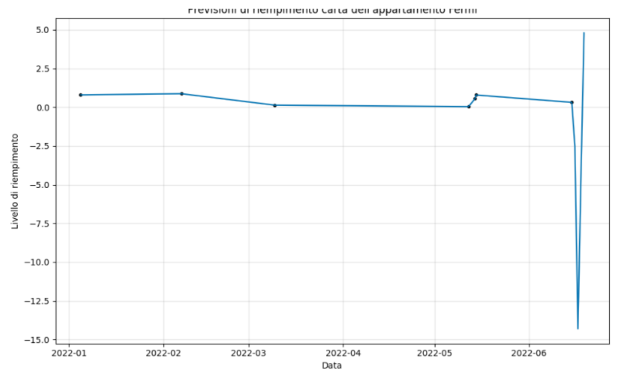
Current fill level graph of the paper waste of the Fermi apartment

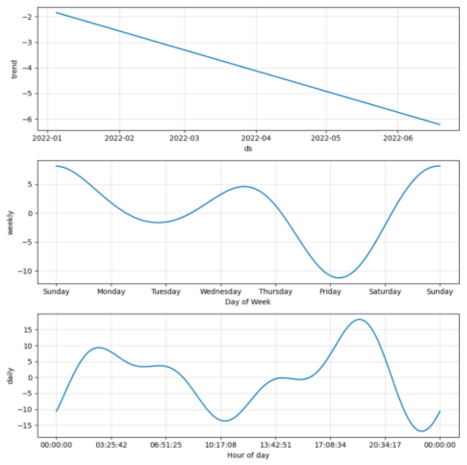


	Ds	Trend	Yhat_lower	Yhat_upper	Trend_lower	Trend_upper	Yhat
0	2022-01-04 20:5	-1.85	0.8	0.8	-1.85	-1.85	0.8
1	2022-02-06 20:5	-2.72	0.87	0.87	-2.72	-2.72	0.87
2	2022-03-09 13:3	-3.53	0.13	0.13	-3.53	-3.53	0.13
3	2022-05-12 03:5	-5.22	0.04	0.04	-5.22	-5.22	0.04
4	2022-05-14 07:0	-5.27	0.56	0.56	-5.27	-5.27	0.56
5	2022-05-14 15:0	-5.28	0.79	0.79	-5.28	-5.28	0.79
6	2022-06-15 01:1	-6.12	0.32	0.32	-6.12	-6.12	0.32
7	2022-06-16 01:1	-6.14	-2.53	-2.53	-6.14	-6.14	-2.53
8	2022-06-17 01:1	-6.17	-14.3	-14.3	-6.17	-6.17	-14.3
9	2022-06-18 01:1	-6.2	-4.46	-4.46	-6.2	-6.2	-4.46
10	2022-06-19 01:1	-6.22	4.8	4.8	-6.22	-6.22	4.8
*							

Filling predictions made by facebook prophet

Forecast plot: A graph containing a plot of historical data points indicated by black dots and the forecast curve indicated by a blue line.





Components plot: a group of plots corresponding to various time series components (trend, seasonilities)

Software Technologies

Flask

docker

React

VPS

OpenStreetMap

http://

SQLAlchemy

NGINX

SOLIDWORKS

openroute service

gunicorn

Thanks for your attention! Questions?

Electronic devices

Alessia Saporita, Vincenzo Lapadula, Michele Giarletta

Electronic devices

Sensors

- 3-axis gyroscope MPU-6050
- CO2 sensor Mq135
- Temperature and humidity sensor DHT11
- Ultrasonic sensor HC SR04

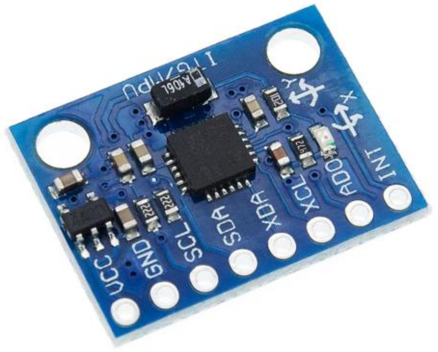
Actuators

- 16x2 I2C LCD display
- RFID RC-522

Microcontroller

- ESP32

Sensors



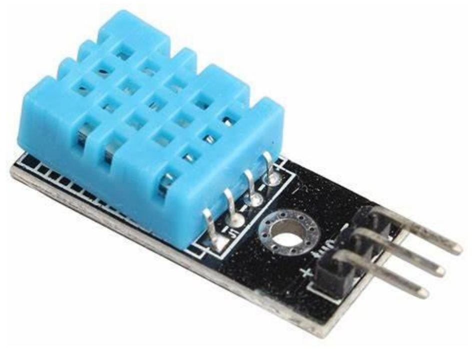
3-axis gyroscope MPU-6050:

- Used to detect the movement direction and magnitude
- I2C Interface
- Low-cost
- Easy to get up and running and capturing the raw data output of the device

CO2 sensor Mq135:

- Used to detect the presence of flammable gases
- Marketed as a generalized “air quality” sensor, capable of measuring the concentrations of several gases, one of which is CO2
- Low-cost alternative to more specialized (and more expensive) CO2-specific sensors

Sensors

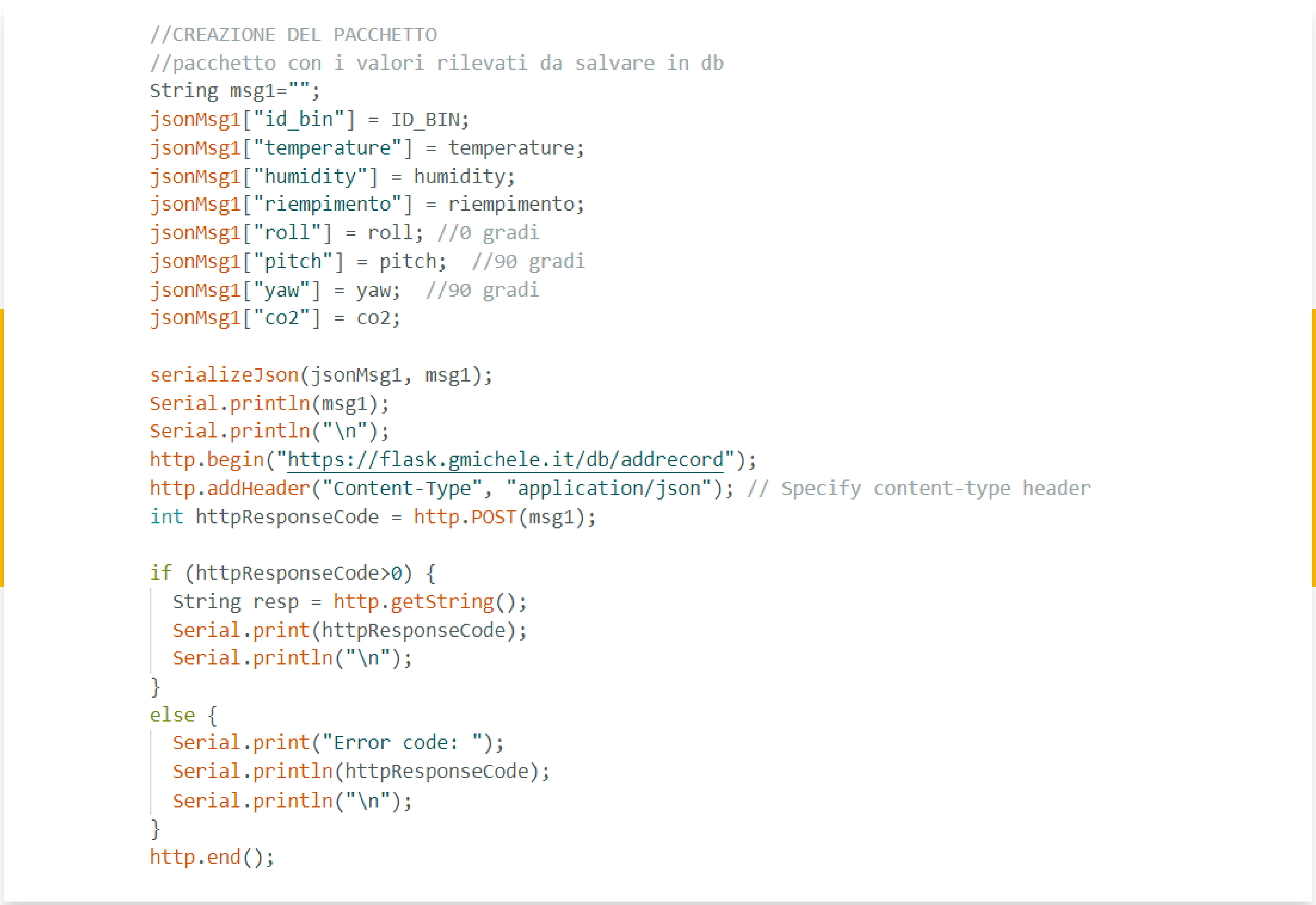


Temperature and humidity sensor DHT11:

- Output: Serial data
- Temperature Range: 0°C to 50°C
- Humidity Range: 20% to 90%
- Accuracy: ±1°C and ±1%
- Easy to interface with other microcontrollers

Ultrasonic sensor HC SR04:

- Minimum measurable distance 2cm
- Maximum distance measurable 400cm
- Resolution: 3 mm
- Simple method for measuring distances, where high precision combined with a high measuring range is required



```
//CREAZIONE DEL PACCHETTO
//pacchetto con i valori rilevati da salvare in db
String msg1="";
jsonMsg1["id_bin"] = ID_BIN;
jsonMsg1["temperature"] = temperature;
jsonMsg1["humidity"] = humidity;
jsonMsg1["riempimento"] = riempimento;
jsonMsg1["roll"] = roll; //0 gradi
jsonMsg1["pitch"] = pitch;  //90 gradi
jsonMsg1["yaw"] = yaw;  //90 gradi
jsonMsg1["co2"] = co2;

serializeJson(jsonMsg1, msg1);
Serial.println(msg1);
Serial.println("\n");
http.begin("https://flask.gmichele.it/db/addrecord");
http.addHeader("Content-Type", "application/json"); // Specify content-type header
int httpResponseCode = http.POST(msg1);

if (httpResponseCode>0) {
  String resp = http.getString();
  Serial.print(httpResponseCode);
  Serial.println("\n");
}
else {
  Serial.print("Error code: ");
  Serial.println(httpResponseCode);
  Serial.println("\n");
}
http.end();
```

Sensors: package

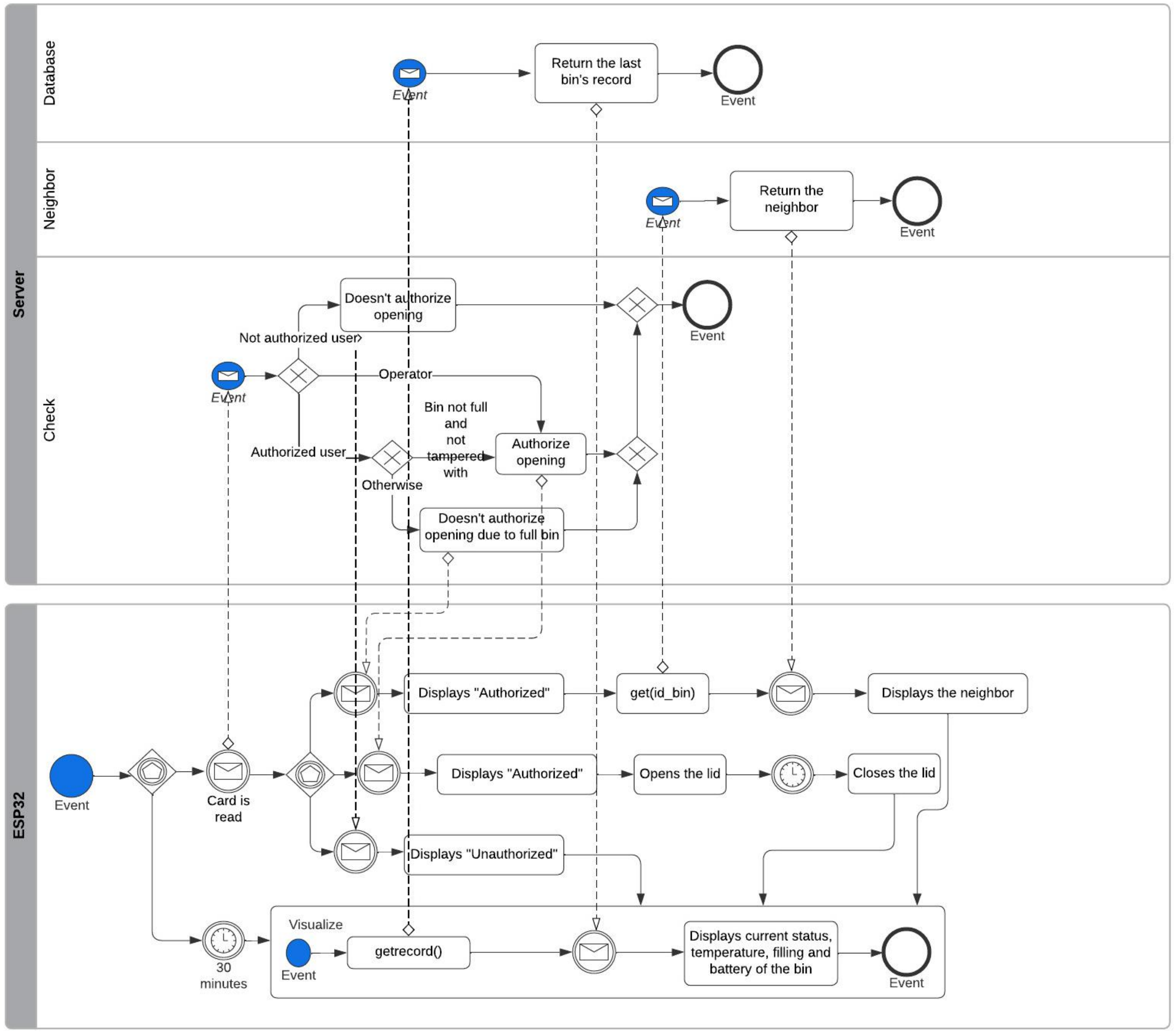
Actuators

Card Reader RFID RC-522 :

- It is normally used in application where certain person/object has to be identified with a unique ID.
- Communication : SPI, I2C protocol, UART
- The module can be easily used with Arduino because of its readily available RC522 RFID Arduino library
- RF Module consists of a RFID reader, RFID card and a key chain

16x2 I2C LCD display:

- It allows you to display up to 32 characters at a time, arranged in 2 lines of 16 characters each
- the integrated I2C communication interface makes it extremely easy to use with Arduino


Server
Database
Event
Return the last bin's record
Event
Neighbor
Event
Return the neighbor
Event
Check
Doesn't authorize opening
Event
Not authorized user
Event
Operator
Bin not full and not tampered with
Authorized user
Authorize opening
Otherwise
Doesn't authorize opening due to full bin
ESP32
Displays "Authorized"
get(id_bin)
Displays the neighbor
Event
Card is read
Displays "Authorized"
Opens the lid
Closes the lid
Displays "Unauthorized"
30 minutes
Visualize
Event
getrecord()
Displays current status, temperature, filling and battery of the bin
Event

Actuators

Thanks for your attention! Questions?