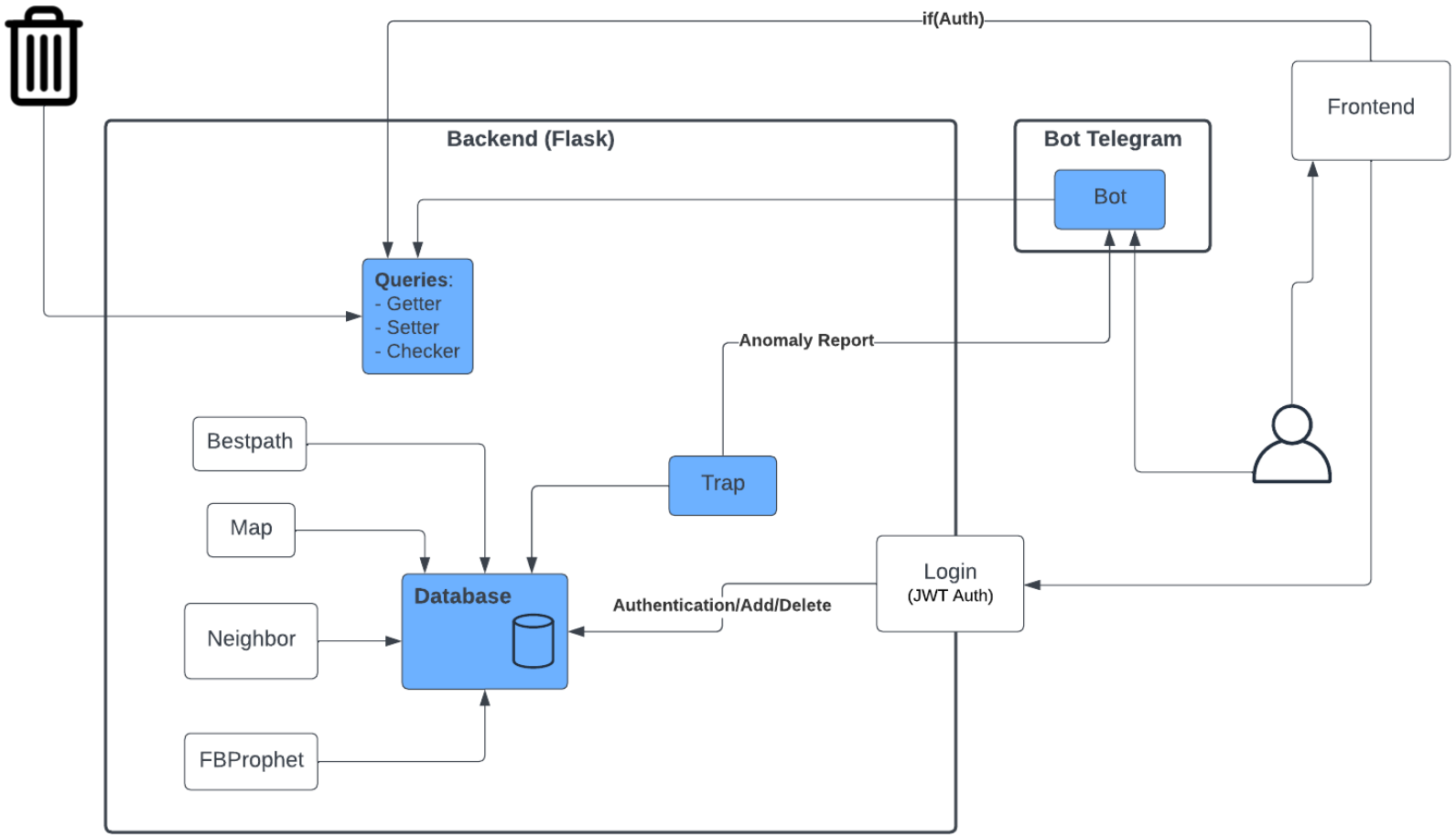
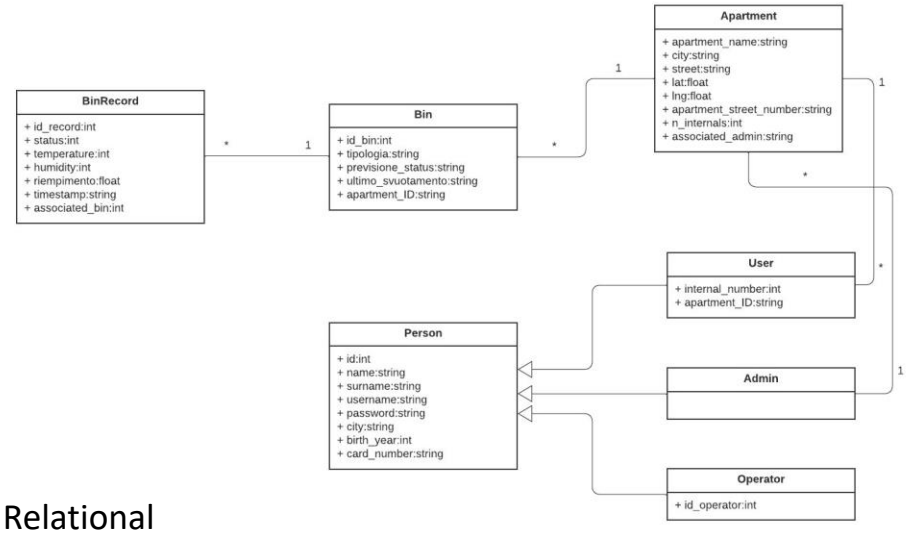
Technical part

16/03/2022

MODULES


if(Auth)
Frontend
Backend (Flask)
Bot Telegram
Bot
Queries:
- Getter
- Setter
- Checker
Anomaly Report
Bestpath
Trap
Map
Login
(JWT Auth)
Database
Authentication/Add/Delete
Neighbor
FBProphet

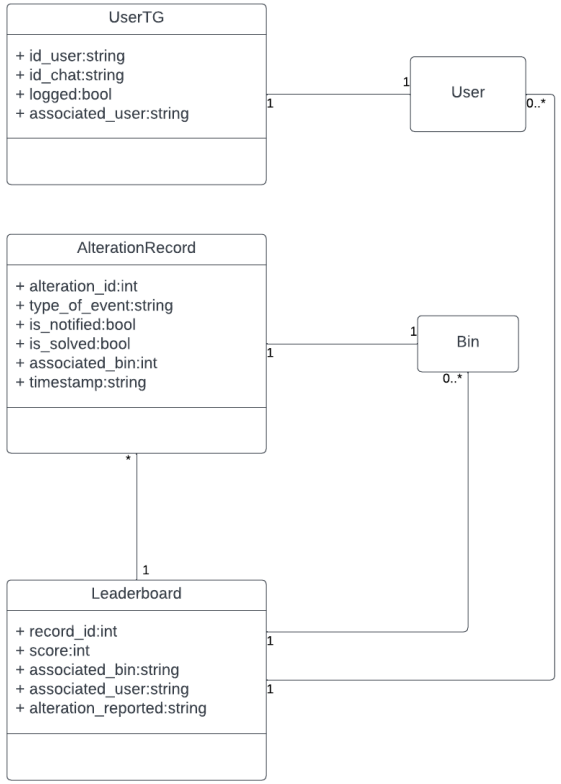
DATABASE

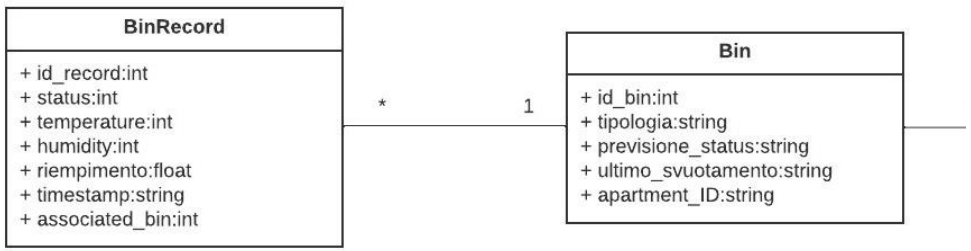


SQLAlchemy

- SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL. (https://www.sqlalchemy.org/)

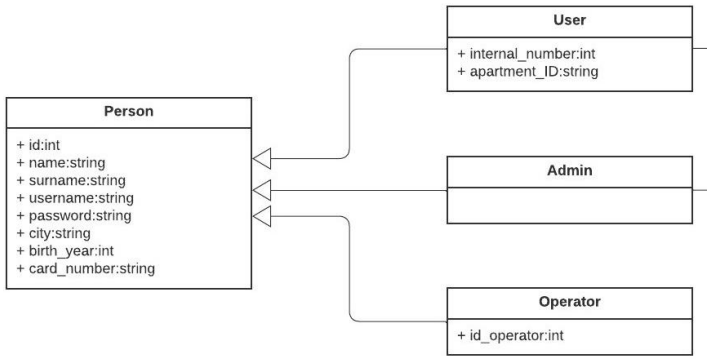
- It allows us to define regular Python objects and methods and translates them into low-level SQL database instructions.





```python
class BinRecord(db.Model):
    __tablename__ = "binRecord"
    id_record = db.Column("id_record", db.Integer, primary_key=True)

    # 1: integro e non-pieno, 2: integro e pieno, 3: manomesso e non-pieno, 4: manomesso e pieno
    status = db.Column("status", db.Integer)
    temperature = db.Column("temperature", db.Integer, nullable=False)
    humidity = db.Column("humidity", db.Integer, nullable=False)
    riempimento = db.Column("livello_di_riempimento", db.Float, nullable=False)
    timestamp = db.Column("Timestamp", db.String, nullable=False, default=str(
        datetime.utcnow().replace(microsecond=0)))

    # FK
    associated_bin = db.Column(
        "associated_bin", db.Integer, db.ForeignKey("bin.id_bin"))

    def __init__(self, jsonObj):
        self.associated_bin = jsonObj["id_bin"]
        self.status = jsonObj["status"]
        self.temperature = jsonObj["temperature"]
        self.humidity = jsonObj["humidity"]
        self.riempimento = jsonObj["riempimento"]
        self.timestamp = jsonObj["timestamp"]
```

```python
class Bin(db.Model):
    __tablename__ = "bin"
    id_bin = db.Column("id_bin", db.Integer, primary_key=True)
    tipologia = db.Column("tipologia", db.String)
    previsione_status = db.Column(
        "previsione_status", db.String, nullable=True, default=""
    )
    ultimo_svuotamento = db.Column(
        "ultimo_svuotamento", db.String(), nullable=False, default=""
    )

    # FK
    apartment_ID = db.Column(
        "apartment_ID", db.String, db.ForeignKey("apartment.apartment_name")
    )

    def __init__(self, tipologia: str, apartment_ID: str):
        self.tipologia = tipologia
        self.apartment_ID = apartment_ID
```




```python
class Person:
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column("username", db.String(20),
                         nullable=False, unique=True)

    name = db.Column("name", db.String)
    surname = db.Column("surname", db.String)
    password = db.Column("password", db.String, nullable=False)
    city = db.Column("city", db.String)
    birth_year = db.Column("birth_year", db.Integer)
    card_number = db.Column("card_number", db.String)

    def __init__(self,
        username: str,
        name: str,
        surname: str,
        password: str,
        city: str,
        birth_year: int,
        card_number: str
    ):
        self.username = username
        self.name = name
        self.surname = surname
        self.password = password
        self.city = city
        self.birth_year = birth_year
        self.card_number = card_number
```

```python
class User(Person, db.Model):
    __tablename__ = "user"
    internal_number = db.Column("internal_number", db.Integer)

    # FK
    apartment_ID = db.Column(
        "apartment_ID", db.String, db.ForeignKey("apartment.apartment_name"))

    def __init__(self, p: Person, apartment_ID: str, internal_number: int):
        super().__init__(p.username, p.name, p.surname,
                         p.password, p.city, p.birth_year, p.card_number)
        self.apartment_ID = apartment_ID
        self.internal_number = internal_number
```

```python
class Admin(Person, db.Model):
    __tablename__ = "admin"

    def __init__(self, x: Person) -> None:
        super().__init__(x.username, x.name, x.surname,
                         x.password, x.city, x.birth_year, x.card_number)
```

```python
class Operator(Person, db.Model):
    __tablename__ = "operator"
    id_operator = db.Column("idOperator", db.Integer)

    def __init__(self, x: Person, id: int) -> None:
        super().__init__(x.username, x.name, x.surname,
                         x.password, x.city, x.birth_year, x.card_number)
        self.id_operator = id
```

QUERIES

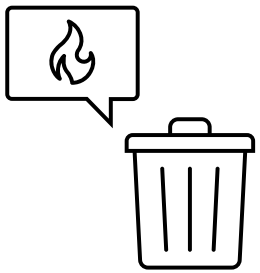
Queries:
- Getter
- Setter
- Checker

Getter (/get):
- /prevision/<string:apartment>
- /urlprevision/<string:apartment>
- /getprofileuser/<string:uid>
- /getprofileadmin/<string:uid>
- /getScore/<string:usr>
- /getSession/<string:usr>
- /leaderboard
- /getrecord/<string:id_bin>
- /dataAdmin/<string:uid>
- /getBins/<string:city>
- /getUsers/<string:city>
- /getypes/<string:apartment>
- /getApartmentUsers/<string:apartment>
- /getBinInfo/<string:id_bin>
- /getApartment/<string:name>

Setter (/set):
- /set_TelegramSession/<string:usr>&<int:idchat>

Checker (/check):
- /checkSession/<string:userid>
- /checkUsername/<string:usr>
- /checkuid/<string:uid>&<int:id_bin>
- /checkAdmin/<string:uid>&<string:password>

TRAP



Fired

Filled up

Ravaged

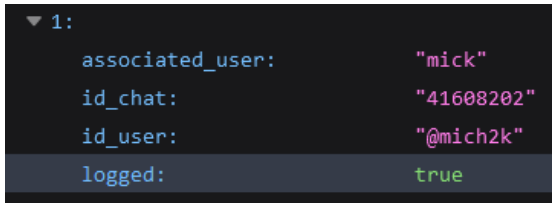
Send POST Request
`https://api.telegram.org/bot{TOKEN}/sendMessage`

```
keyboard = [[InlineKeyboardButton("Segnala", callback_data='solved')],
            [InlineKeyboardButton("Risolto", callback_data='report')]]

reply_markup = InlineKeyboardMarkup(keyboard)

data = {"chat_id": '',
        'text': text,
        'reply_markup': json.dumps(reply_markup.to_dict())}
```

BOT TELEGRAM

```
application = ApplicationBuilder().token(TOKEN).build()

start_handler = CommandHandler('start', start)
get_score_handler = CommandHandler('score', get_score)
get_leaderboard_handler = CommandHandler('leaderboard', get_leaderboard)
help_handler = CommandHandler('help', helper)

call_handler = CallbackQueryHandler(status)

application.add_handlers([start_handler,get_leaderboard_handler,
                          call_handler, get_score_handler, help_handler])

application.run_polling()
```

UserTG



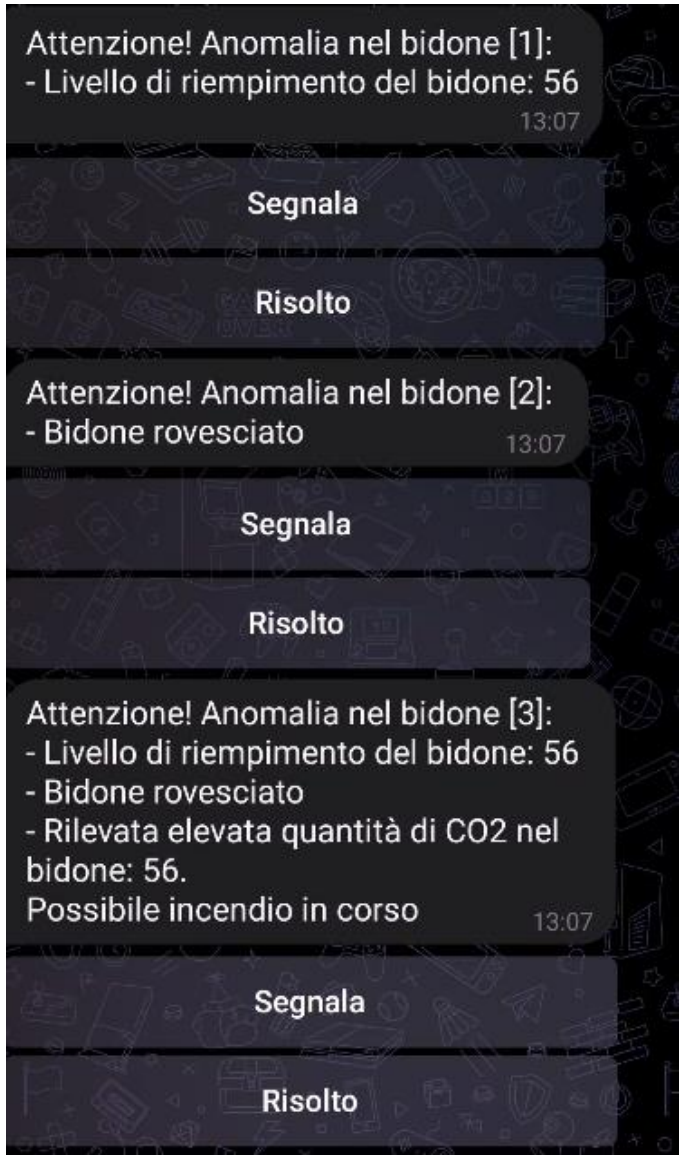
`/set_TelegramSession/<string:usr>&<int:idchat>`

`/getScore/<string:usr>`

TRAP

Send POST Request

AlterationRecord



InlineKeyboardButton

```
query = update.callback_query
await query.answer()

answer = query.data
```

Leaderboard

LOGIN

Why JWT?

1. **Scalability**: cookies can make more complex to maintain the authentication state across different servers.
   JWT can be used to maintain authentication and authorization state in a decentralized way.
2. **Security**: Cookies can be vulnerable to attacks such as Cross-Site Request Forgery (CSRF) and Cross-Site Scripting (XSS). JWT offers a higher level of security thanks to the use of a digital signature to verify the integrity of the token and the possibility of encrypting the data contained in the token itself.
3. **Efficiency**: Using cookies requires transferring data between the client and server on every request. By using JWT, authentication and authorization data can be stored directly in the token, avoiding the need to transfer this data between client and server on every request.


```
"access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
    eyJmcmVzaCI6ZmFsc2UsImlhdCI6MTY3ODkyOTA5MCwianRpIjo
    0I5wz2OBY8Nj9_hHrXb56HlCsZeOx265QP8g7Jlo0kM",
"birth_year": 2000,
"card_number": "d3370a8",
"city": "Modena",
"id": 1,
"name": "Mario",
"surname": "Rossi"
```


POST Request to:
https://flask.gmichele.it/login/loginadmin

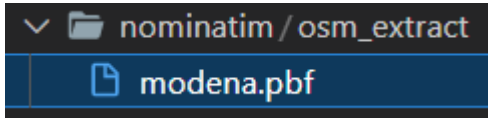
BESTPATH & MAP

- Leaflet is an open-source JavaScript library used to build web mapping applications

```
<head>
  <title>OSM Points</title>
  <link rel="stylesheet" href="{{url_for('.static', filename='leaflet.css')}}" />
</head>

<script src="{{url_for('.static', filename='leaflet.js')}}"></script>
```

https://flask.gmichele.it/map/viewmap/

openroute service

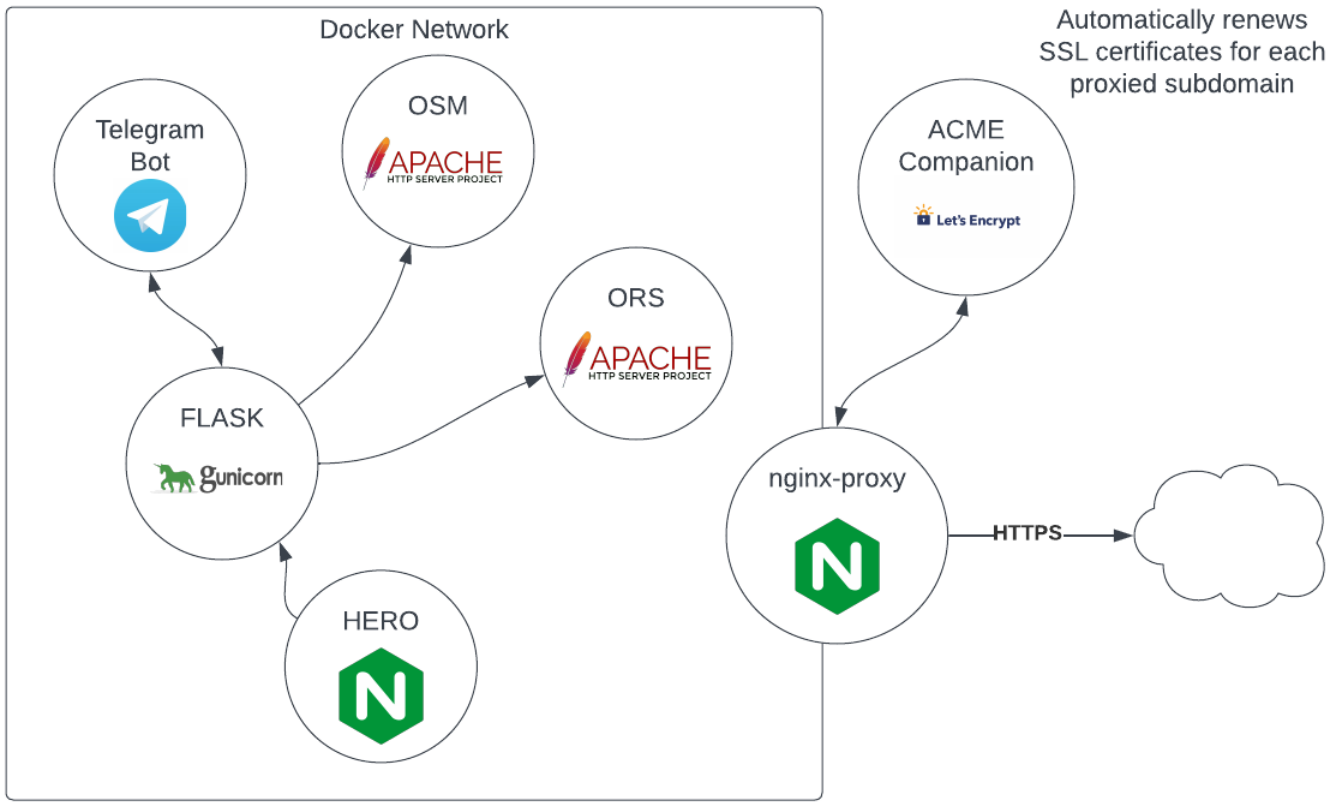


http://download.geofabrik.de/

- OpenStreetMap (OSM) is a free, open geographic database updated and maintained by a community of volunteers via open collaboration.
- OpenRouteService (ORS) is a very useful routing service used for matrix distances or optimal path. It is based on OSM data.

`https://osm.gmichele.it/reverse?lat=<lat>&lon=<lon>&format=json`

DOCKER NETWORK

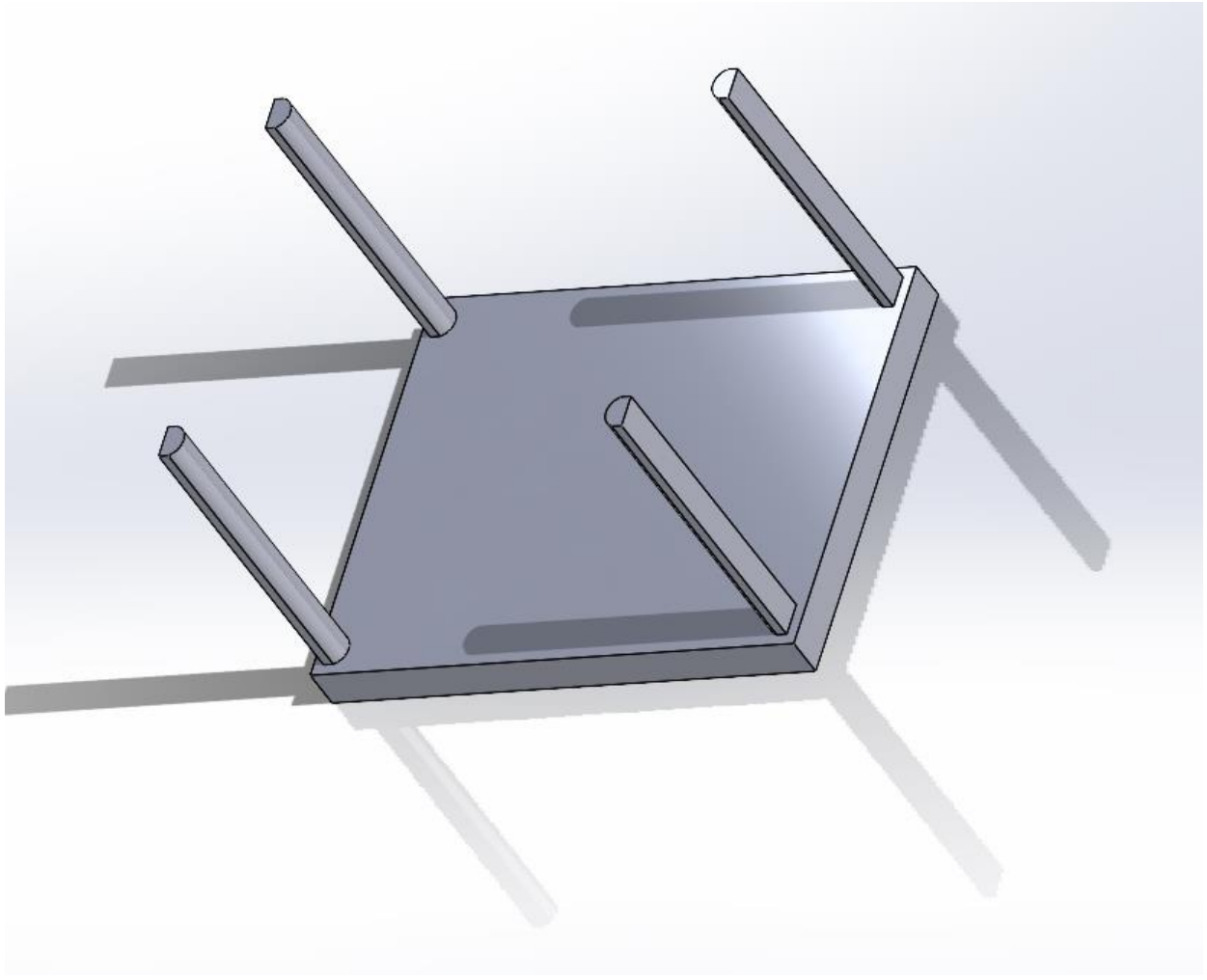


* Self-hosted

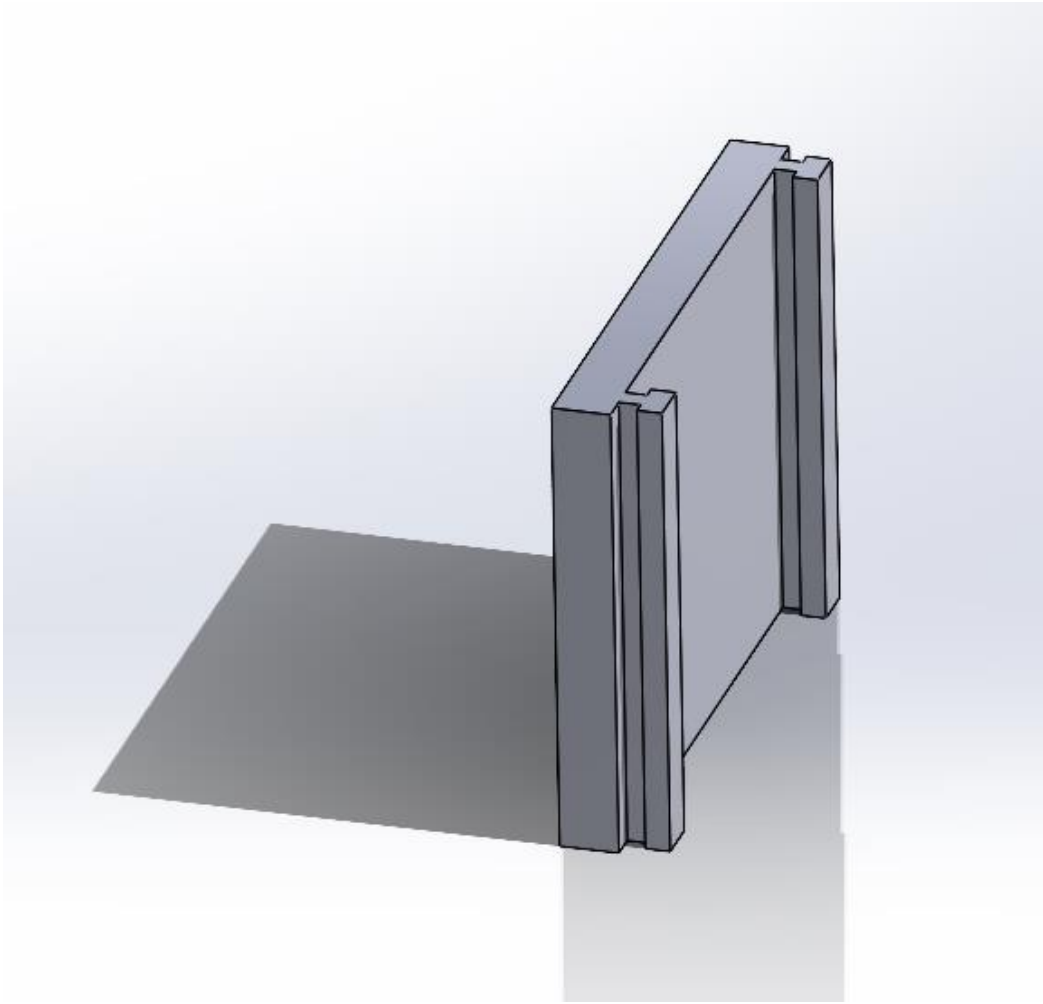
PROTOTYPE DESIGN INTRO

- Material (**polylactic acid**)
- Printer: Flying Bear Ghost 5
- CAD Software: Solidworks 2023
- Slicer: CURA

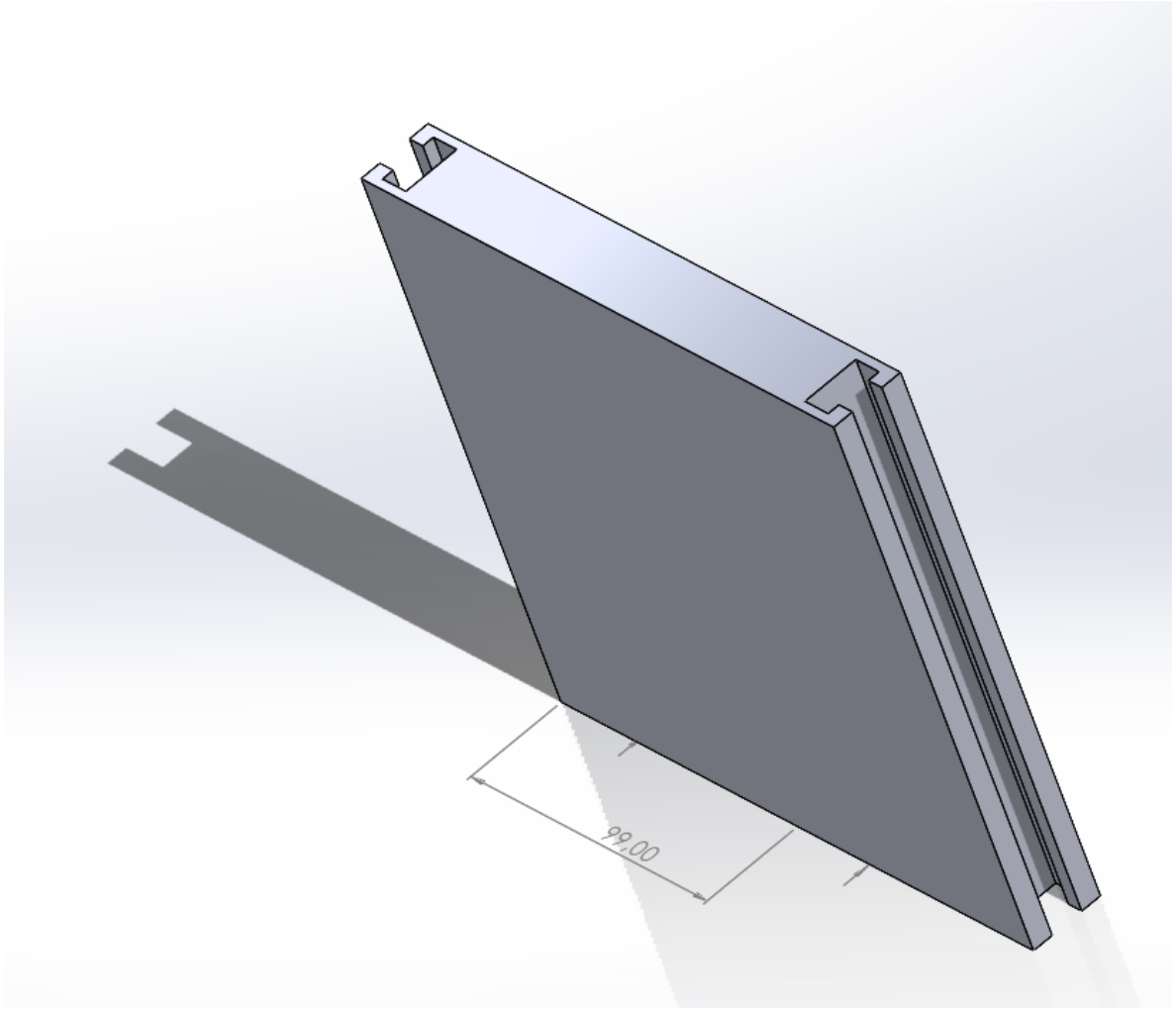
PROTOTYPE DESIGN BASE



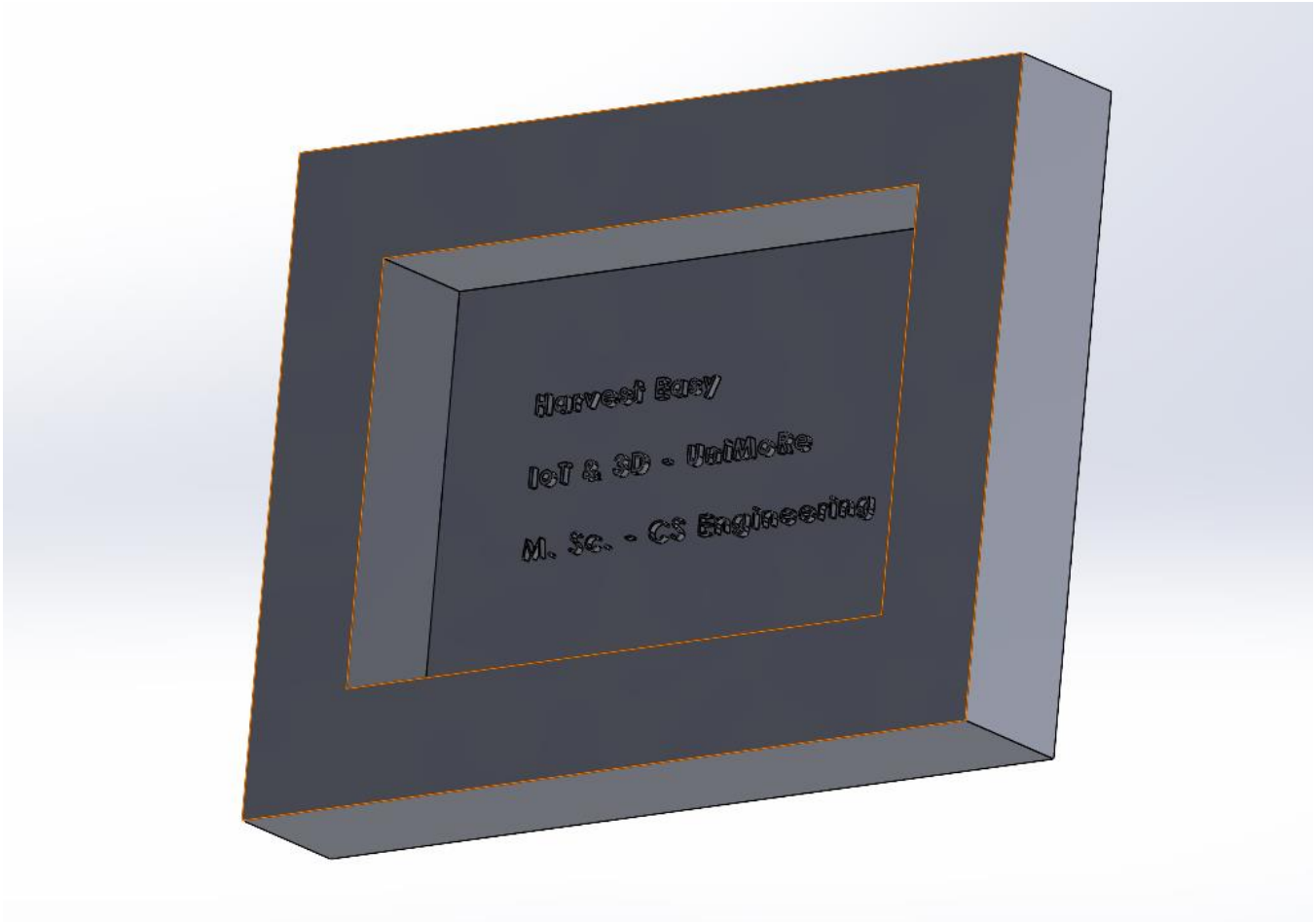
PROTOTYPE DESIGN
MALE-FEM WALL



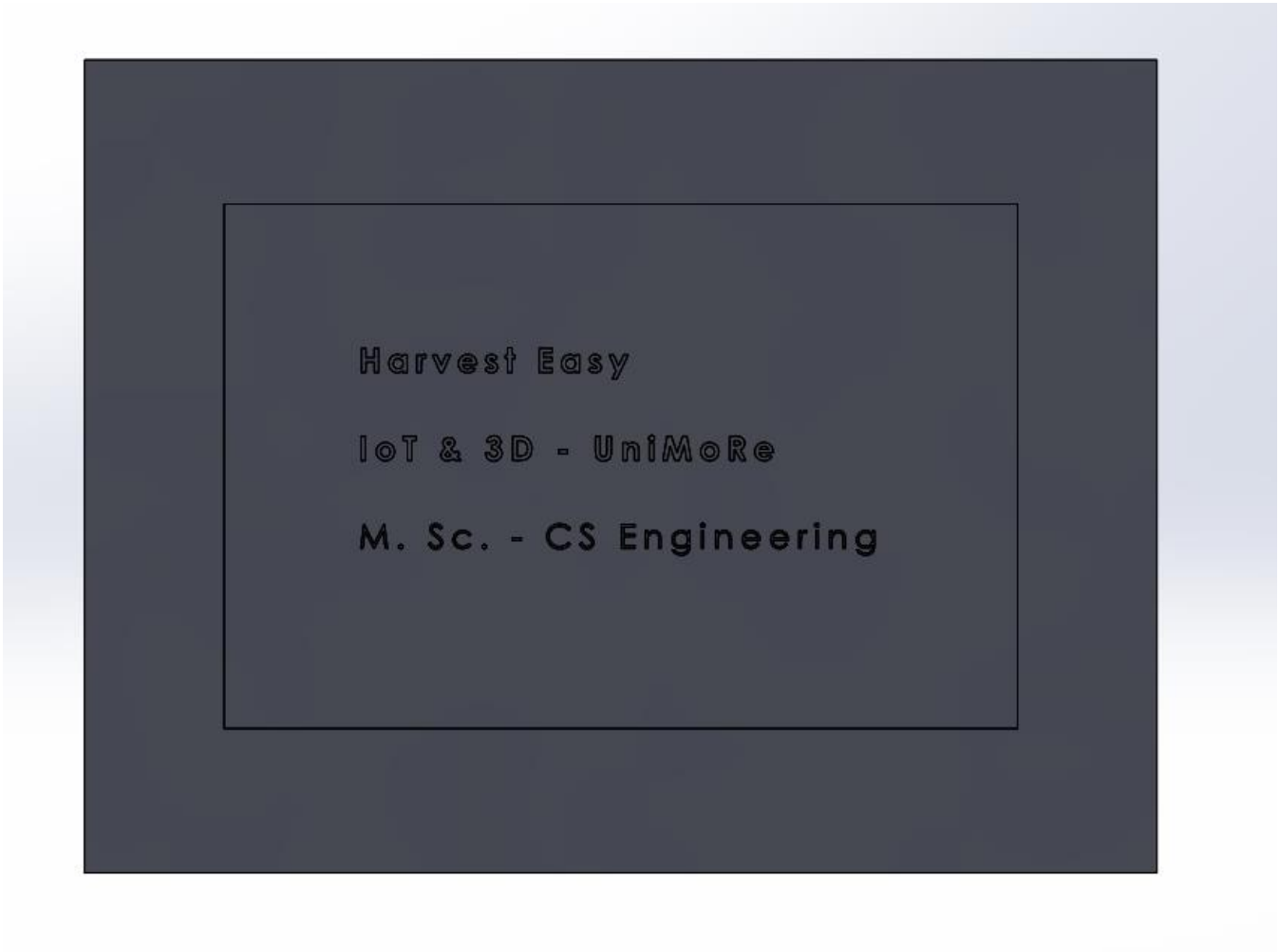
PROTOTYPE DESIGN
FEM WALL



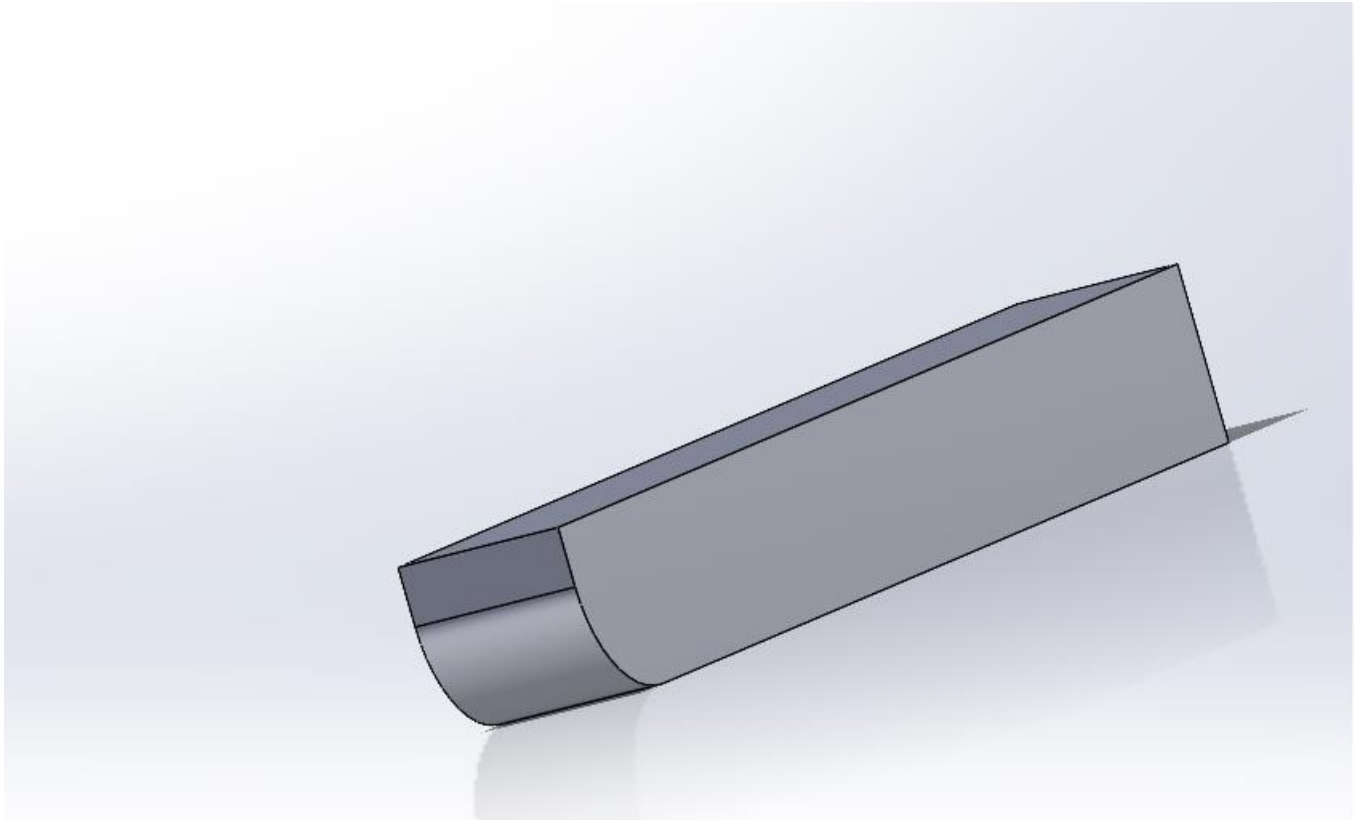
PROTOTYPE DESIGN
TOP



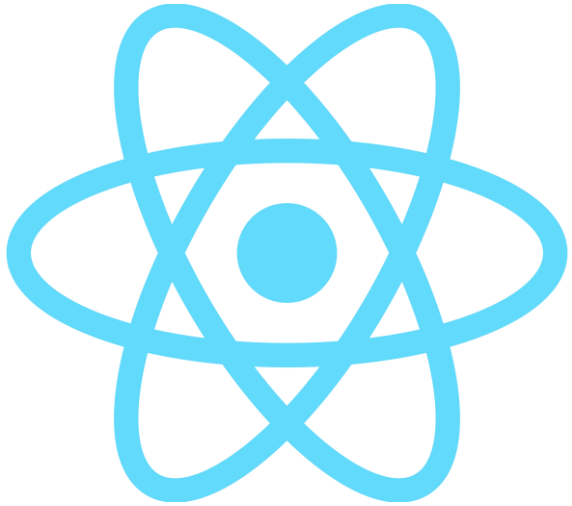
PROTOTYPE DESIGN
TOP

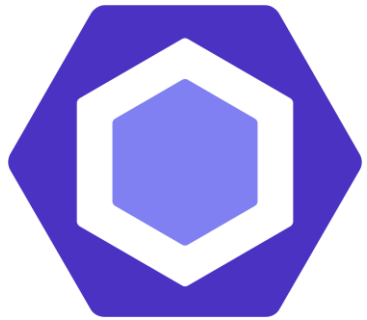


PROTOTYPE DESIGN
ENGINE ARM



FRONTEND TECHNOLOGIES





tailwindcss

FRONTEND

- ATOMIC DESIGN PATTERN
- REUSABLE COMPONENTS
- RESPONSIVE
- MODERN UI

FRONTEND

```
useEffect(() => {

    if (!navigator.geolocation) {
        setGeoSupported(false);
        return;
    }

    navigator.geolocation.getCurrentPosition(function (position) {
        setGeoCoordinates(new Coordinates(position.coords.latitude, position.coords.longitude));
        setGeoAllowanceGiven(true);
    }, function (error) {
        if (error.code == error.PERMISSION_DENIED) {
            console.log("permission denied");
            setGeoAllowanceGiven(false);
        }
    });
})
```

FRONTEND
LOGIN (AXIOS-JWT-BEARER)

```
axios
    .post(url + '/login/loginadmin', data, {
        headers: {
            Accept: "application/json",
            "Content-Type": "application/json;charset=UTF-8",
        },
    })
    .then(({ data }) => {
        //console.log(data);
        if (data['access_token'] === undefined) {
            return <Error header_message="Invalid credentials" body_message="The credentials you have entered are not valid, please try again"></Error>
        } else {
            navigate("/admin_home", {
                state: { coords: deviceGeoCoordinates, access_token: data['access_token'], admin_username: admin_username, apartment_id: apartment_id },
            });
        }

    });
```

FRONTEND
WASTE-LIST COMPONENT

```tsx
const typologyColorMap: TypologyColorMap = { "vetro": "bg-green-600", "plastica": "bg-blue-600", "carta": "bg-yellow-300", "umido": "bg-orange-900", "other":


const WasteListComponent: React.FC<Props> = ({ wasteList }) => {
  if (!wasteList) return (<div className='text-bold'>Loading or not available</div>);

  return (
    <div>
      {Object.keys(wasteList).map((wasteInfoKey, index) => {
        const wasteInfo = wasteList[wasteInfoKey as keyof WasteInfo];
        return (
          <div key={index}>
            <hr></hr>
            <div className='font-medium mb-4 mt-2'>
              Tipologia rifiuto:  <span className='uppercase'>{wasteInfoKey}</span></div>
            <div className='mb-4'>Stato: {wasteInfo.status === 1 ?
                <span> <GreenCircle></GreenCircle> OK!</span> : wasteInfo.status == 2
                ? <span><YellowCircle></YellowCircle> FULL!</span> :
                <span> <RedCircle></RedCircle> DANGER!</span>}</div>
            <BinProphetRecord filling={wasteInfo.riempimento} sort_type={wasteInfoKey} color={typologyColorMap[wasteInfoKey]}
            date={String(wasteInfo.previsione_status)}></BinProphetRecord>
          </div>
        );
      })}
    </div>
  );
};
```

FRONTEND
BASE64 TO IMG COMPONENT

```
const ImmaginiPrevisioni: React.FC<Props> = ({ previsioni }) => {
    if (!previsioni) return (<div className='text-bold'>Loading previsions or not avialable yet for your apartment.</div>);
    return (
        <div>
            {Object.keys(previsioni).map((key: string, index) => (
                <div key={index} className='my-6'>
                    <div className='text-center font-bold my-6'>{capitalizeFirstLetter(key)}</div>
                    <img key={index} src={`${previsioni[key]}`} alt={key} />
                </div>
            ))}
        </div>
    );
};
```

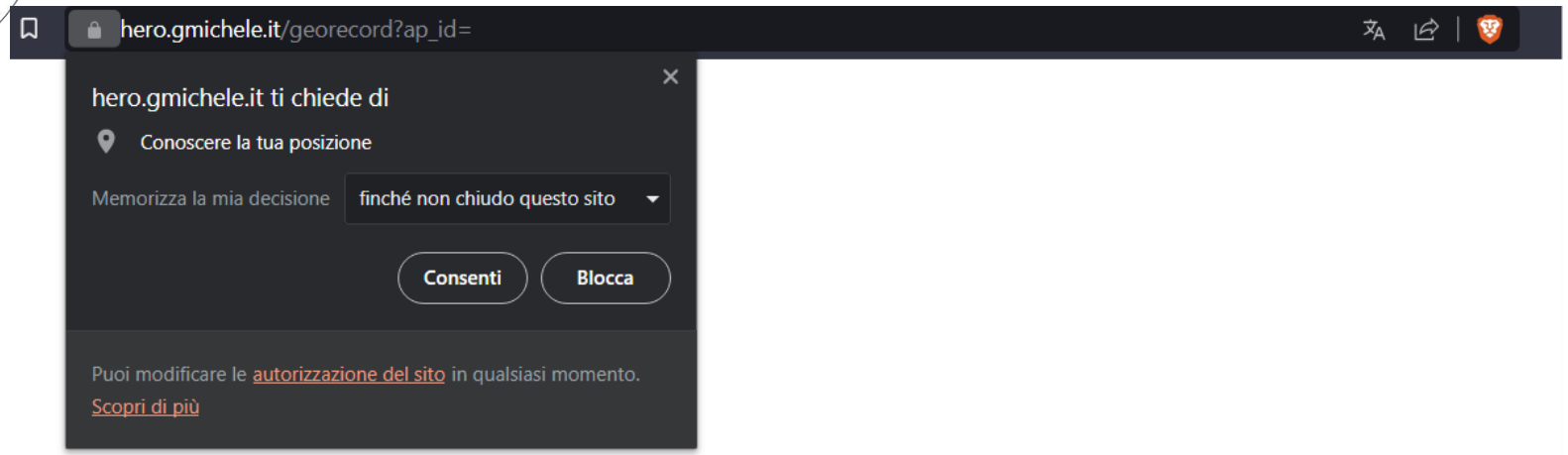
FRONTEND
BASE64 TO IMG COMPONENT

```
class toSendMsg {
  timestamp: string;
  apartment_name: string;
  common_city: string;
  admin_username: string;
  apartment_coords: Coordinates;
  final_people: Person[];
  apartment_waste_sorting: string[];

  constructor(final_people: Person[], apartment_waste_sorting: string[], common_city: string, apartment_name: string, timestamp: string, admin_username: str
    this.final_people = final_people;
    this.apartment_waste_sorting = apartment_waste_sorting;
    this.common_city = common_city;
    this.apartment_name = apartment_name;
    this.timestamp = timestamp;
    this.admin_username = admin_username;
    this.apartment_coords = apartment_coords;
  }
}
```

```
const [apartment_waste_sorting, setApartmentWasteSorting] = useState<Set<string>>(new Set());
```

ADMIN-RECORD AXIOS POST

FRONTEND
SAFETY PAGES



Loading...

FRONTEND
CUSTOM PAGES

404

Apartment ID invalid or not specified or a broken QR Code

Sorry about that! Please visit our hompage to get where you need to go.

Take me there!

Vincenzo Lapadula

Michele Giarletta

Alessia Saporita

https://hero.gmichele.it/

EXTRA