Welcome to the pinewatcher backend recruitment project.

This project is going to present to You all the important aspects of the backend projects in eco-projects.

The recruitment program lasts 3 weeks. Every week You will get a new subset of tasks to be finished in those 7 days, and You will have a chance to iron out some mistakes from the prior week.

There are some important rules:

1. Deadlines are the most important thing along finishing tasks - appropriate commits should be available by 24:00 that day.
2. If for any reason one might need deadline extension it should be reported max. 2 days before the deadline.
3. **Instructions that are unclear should be reported ASAP**
4. Read the instruction in full before starting your work
5. Never store login data in any files stored in the repository
6. Rules can be changed anytime

Instructions

In this chapter You will find all the important information regarding the assigned tasks.

Trivia

The client (company) has robots, that are riding around. Each robot carries a communication device and has build in sensor. Communication device is our gateway to the robot and its data, and can be moved between robots in case of failure. MQTT is used for communication. Robot is sending two types of data: its location and sensor data. Robots are manufactured by different manufacturers (our clients are not manufacturers) and has some distinctive information like serial number, production date and type (4 wheeler, amphibian, tracked, flying). Due to the fact that robots have different sizes, communication devices have different form factors. The platform is organised as a SAAS, and thus requires separation of robots and devices using user (company/client) accounts. Because of external requirements we need to store KRS (krajowy rejestr sądowniczy) data linked to user accounts.

Robots are sending data:

1. telemetry:
    1. timestamp - EPOCH
    2. humidity - int
    3. temperature - int
    4. pressure - int
2. location:
    1. timestamp - EPOCH
    2. latitude - float
    3. longitude - float

Additional information

General folder structure of the django project:

```
dummy_project
├── .env
├── app1
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── migrations
│   ├── models.py
│   ├── tests.py
│   └── views.py
├── app2
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── migrations
│   ├── models.py
│   ├── tests.py
│   └── views.py
├── manage.py
└── project (sometimes named the same as the project - dummy_project)
    ├── __init__.py
    ├── __pycache__
    ├── asgi.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py
```

Tasks

1. Setup your project repository on github as private and follow conventional commits
2. Setup docker containers with django and postgres using docker compose
3. Create models based on the provided story
4. Create endpoints returning the data from the models as JsonResponse
    1. Return all robots (type and owner only)
    2. Return robot data (all data)
    3. Add new client
    4. Return telemetry from selected time period (start, end) for a specified robot
    5. Return location from selected time period (start, end) for a specified robot
    6. Return latest locations from all robots
    7. Modify specified robot brand
5. Register models in django admin and add filters, search fields, list display

Note: data access to specified objects should be made using PK

Useful links

1. https://www.conventionalcommits.org/en/v1.0.0/
2. https://docs.djangoproject.com/pl/4.0/intro/tutorial01/
3. https://docs.docker.com/samples/django/
4. https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods