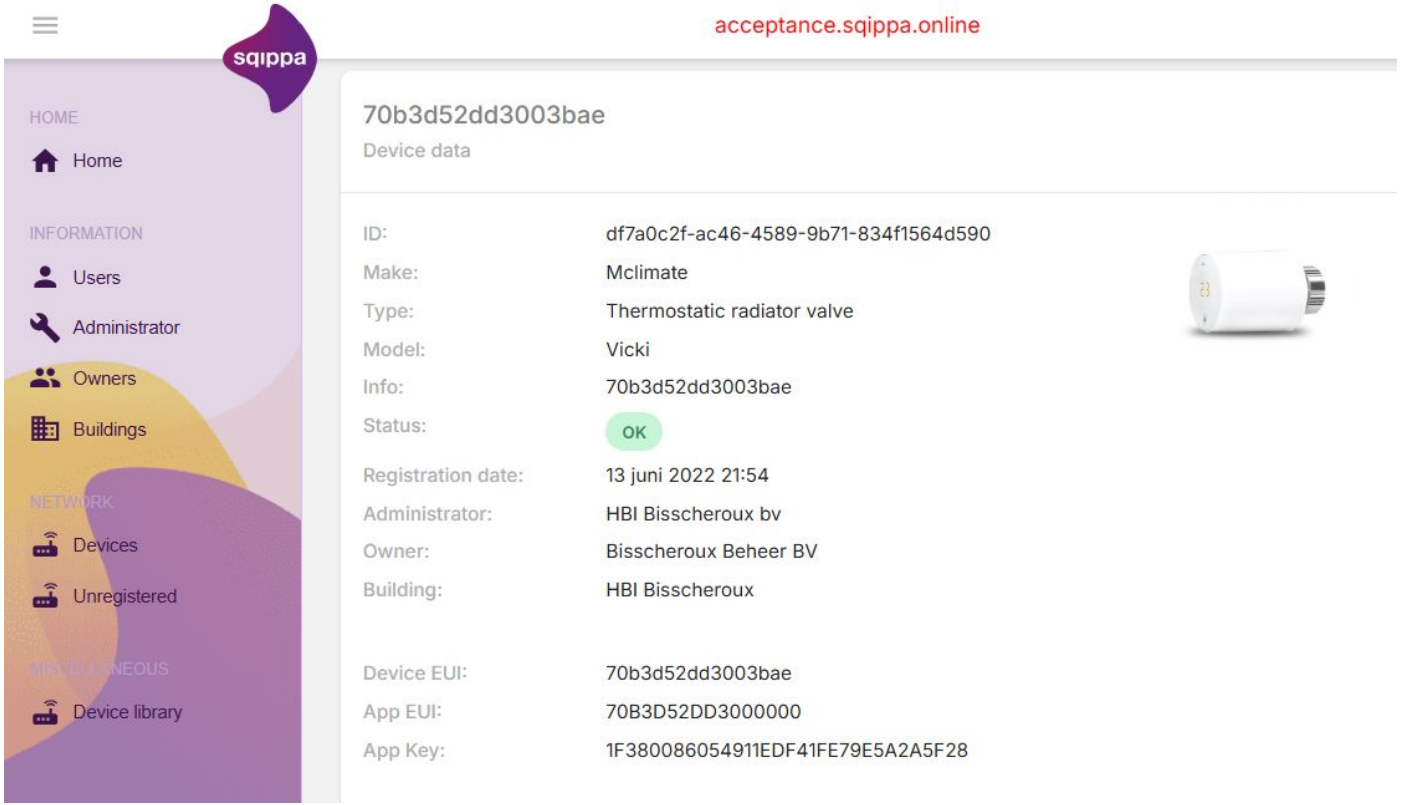
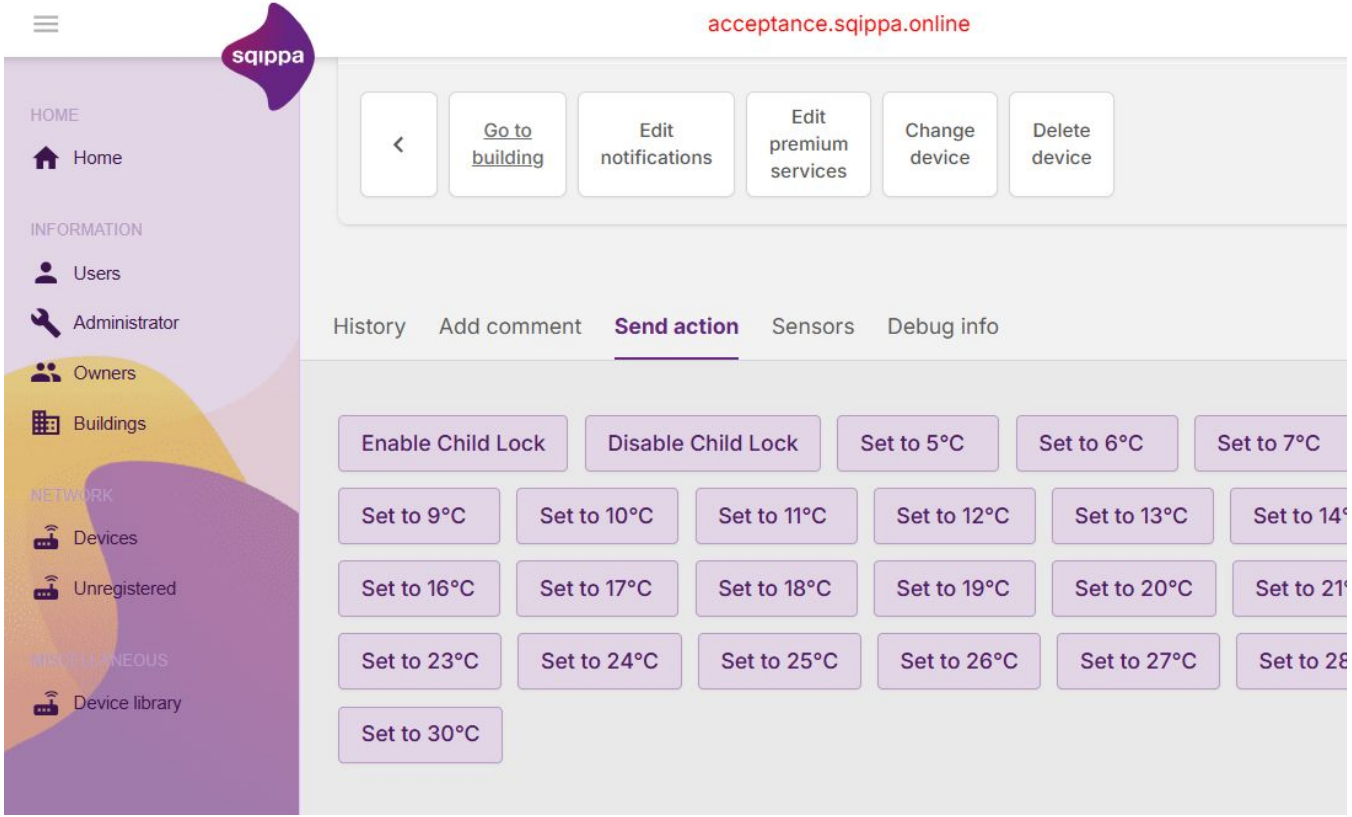
First Project: Sqippa




acceptance.sqippa.online
sqippa
HOME
Home
INFORMATION
Users
Administrator
Owners
Buildings
NETWORK
Devices
Unregistered
MISCELLANEOUS
Device library
Go to building
Edit notifications
Edit premium services
Change device
Delete device
History
Add comment
Send action
Sensors
Debug info
Enable Child Lock
Disable Child Lock
Set to 5°C
Set to 6°C
Set to 7°C
Set to 9°C
Set to 10°C
Set to 11°C
Set to 12°C
Set to 13°C
Set to 16°C
Set to 17°C
Set to 18°C
Set to 19°C
Set to 20°C
Set to 23°C
Set to 24°C
Set to 25°C
Set to 26°C
Set to 27°C
Set to 30°C

Goal

- Way to easily create schedules for devices - aimed at end users

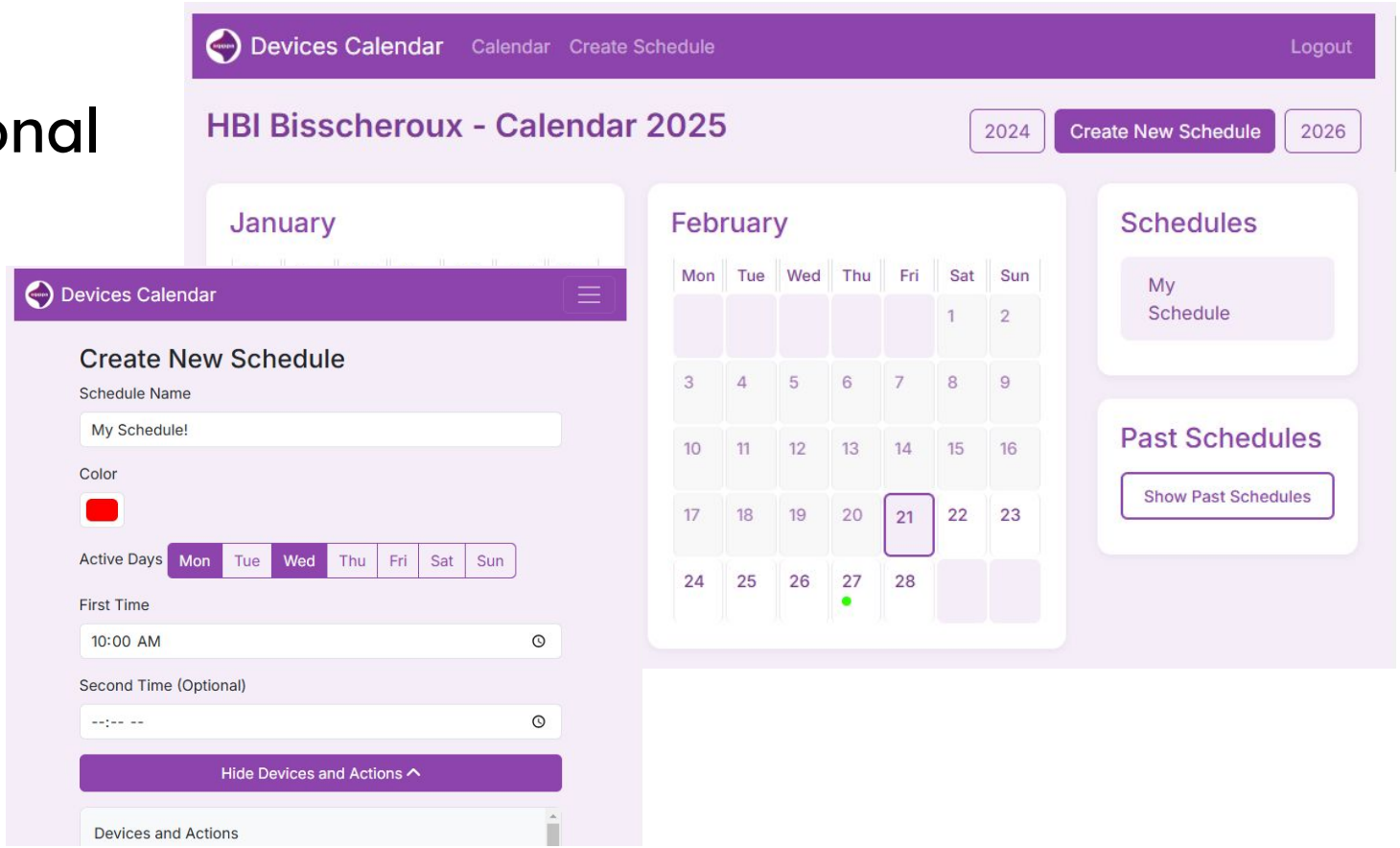
Goal

- Way to easily create schedules for devices - aimed at end users

Work setup:

- Freelancing
- Weekly meetings w/Project Manager
- Occasional meetings w/Software Dev
- Total duration: Sep 2024 - Dec 2024

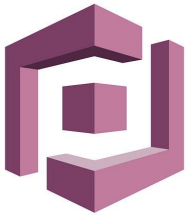
Result:
Fully functional Web app



https://ifthen.sqippa.online/

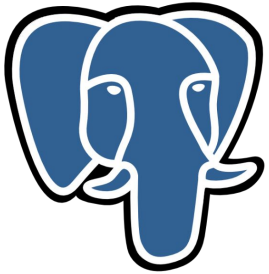
Stack

User authentication



Amazon Cognito

Database



PostgreSQL

Frontend

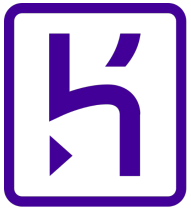
Backend

python

&

django

Deployment MVP



HEROKU

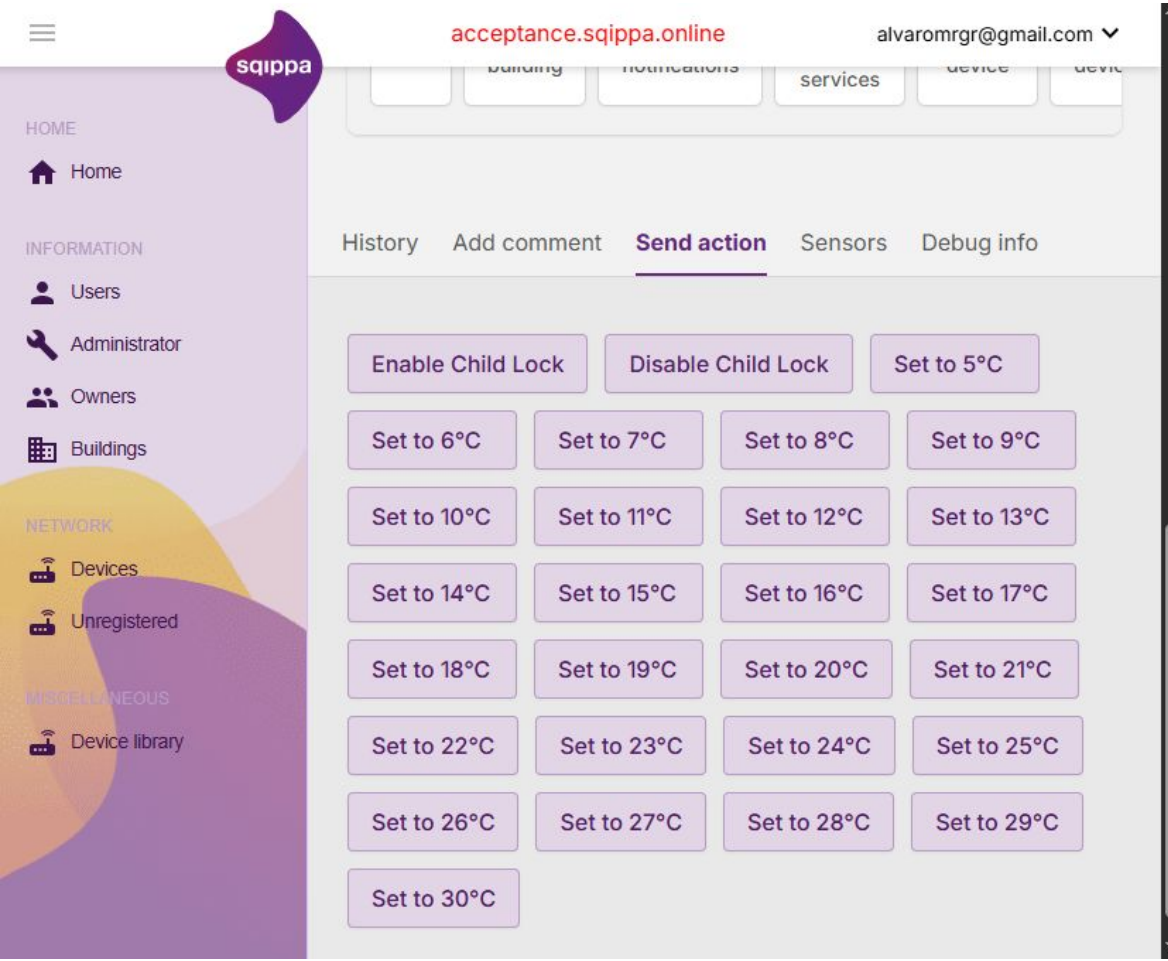
Stable deployment

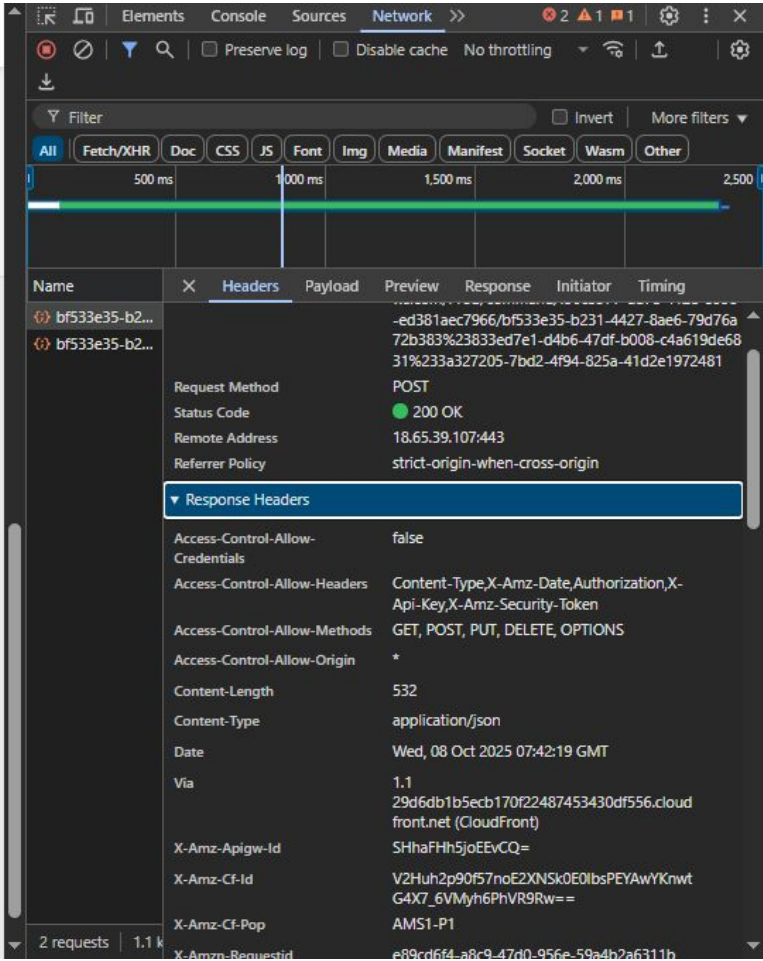
```
Temperature-Calendar/
├── api_calls.py            # Sqippa API calls definition.
├── authentication.py       # Custom Cognito authentication backend.
├── manage.py
├── requirements.txt
├── calendar_app/           # main app.
│   ├── models.py           # DB models (Device, Schedule, etc.).
│   ├── views.py            # App logic (e.g., calendar_view, login_view).
│   ├── urls.py
│   ├── management/
│   │   └── commands/
│   │       └── run_scheduler.py # Cron Job.
│   └── config.py
├── temperature_calendar/
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
├── templates/
│   ├── base.html           .
│   ├── calendar.html       .
│   └── login.html          .
└── static/
    ├── css/
    │   └── calendar.css
    └── js/
        └── calendar.js
```

```python
class Device(models.Model):
    model_name = models.CharField(max_length=100)
    device_id = models.CharField(max_length=100, unique=True)
    location = models.CharField(max_length=100, null=True)
    device_type = models.CharField(
        max_length=20,
        choices=[
            ('RADIATOR', 'Radiator'),
            ('ENERGY_INTENSIVE', 'Energy Intensive Device')
        ]
    )
    available_actions = models.JSONField(default=list)

class Schedule(models.Model):
    template = models.ForeignKey(ScheduleTemplate, on_delete=models.CASCADE, related_name='schedules', null=True)
    device = models.ForeignKey(Device, on_delete=models.CASCADE)
    date = models.DateField()
    scheduled_time = models.TimeField()
    action = models.CharField(max_length=100)
    is_exception = models.BooleanField(default=False)  # True if this is a manual override

    # Authentication fields
    refresh_token = models.CharField(max_length=1024, null=True)  # Store refresh token
    client_id = models.CharField(max_length=100, null=True)  # Store client ID
    region = models.CharField(max_length=50, null=True)  # Store region
    username = models.CharField(max_length=100, null=True)  # Store username for Cognito
```


acceptance.sqippa.online
alvaromrgr@gmail.com
sqippa
services
HOME
Home
INFORMATION
Users
Administrator
Owners
Buildings
NETWORK
Devices
Unregistered
MISCELLANEOUS
Device library
History
Add comment
Send action
Sensors
Debug info
Enable Child Lock
Disable Child Lock
Set to 5°C
Set to 6°C
Set to 7°C
Set to 8°C
Set to 9°C
Set to 10°C
Set to 11°C
Set to 12°C
Set to 13°C
Set to 14°C
Set to 15°C
Set to 16°C
Set to 17°C
Set to 18°C
Set to 19°C
Set to 20°C
Set to 21°C
Set to 22°C
Set to 23°C
Set to 24°C
Set to 25°C
Set to 26°C
Set to 27°C
Set to 28°C
Set to 29°C
Set to 30°C

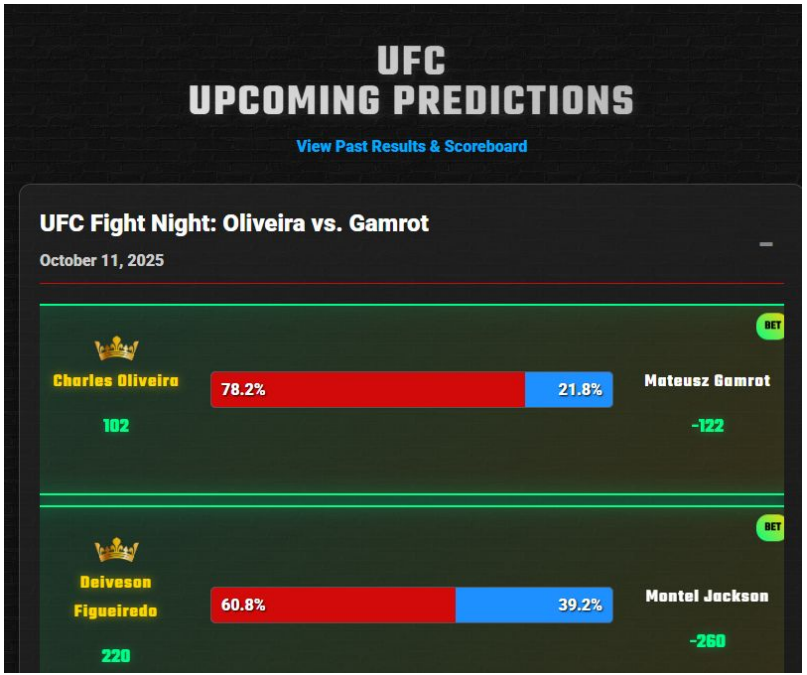

Elements
Console
Sources
Network
Preserve log
Disable cache
No throttling
Filter
Invert
More filters
All
Fetch/XHR
Doc
CSS
JS
Font
Img
Media
Manifest
Socket
Wasm
Other
500 ms
1,000 ms
1,500 ms
2,000 ms
2,500
Name
bf533e35-b2...
bf533e35-b2...
Headers
Payload
Preview
Response
Initiator
Timing
-ed381aec7966/bf533e35-b231-4427-8ae6-79d76a72b383%23833ed7e1-d4b6-47df-b008-c4a619de6831%233a327205-7bd2-4f94-825a-41d2e1972481
Request Method
POST
Status Code
200 OK
Remote Address
18.65.39.107:443
Referrer Policy
strict-origin-when-cross-origin
Response Headers
Access-Control-Allow-Credentials
false
Access-Control-Allow-Headers
Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token
Access-Control-Allow-Methods
GET, POST, PUT, DELETE, OPTIONS
Access-Control-Allow-Origin
*
Content-Length
532
Content-Type
application/json
Date
Wed, 08 Oct 2025 07:42:19 GMT
Via
1.1 29d6db1b5ecb170f22487453430df556.cloudfront.net (CloudFront)
X-Amz-Apigw-Id
SHhaFHh5joEEvCQ=
X-Amz-Cf-Id
V2Huh2p90f57noE2XNSk0E0IbsPEYAwYKnwtG4X7_6VMyh6PhVR9Rw==
X-Amz-Cf-Pop
AMS1-P1
2 requests

Second Project: **UFC predictions**

Goal

- Build predictive models for UFC fights
- Use them to predict future fights
- Automate everything

Result: Webapp that makes predictions and tracks performance automatically



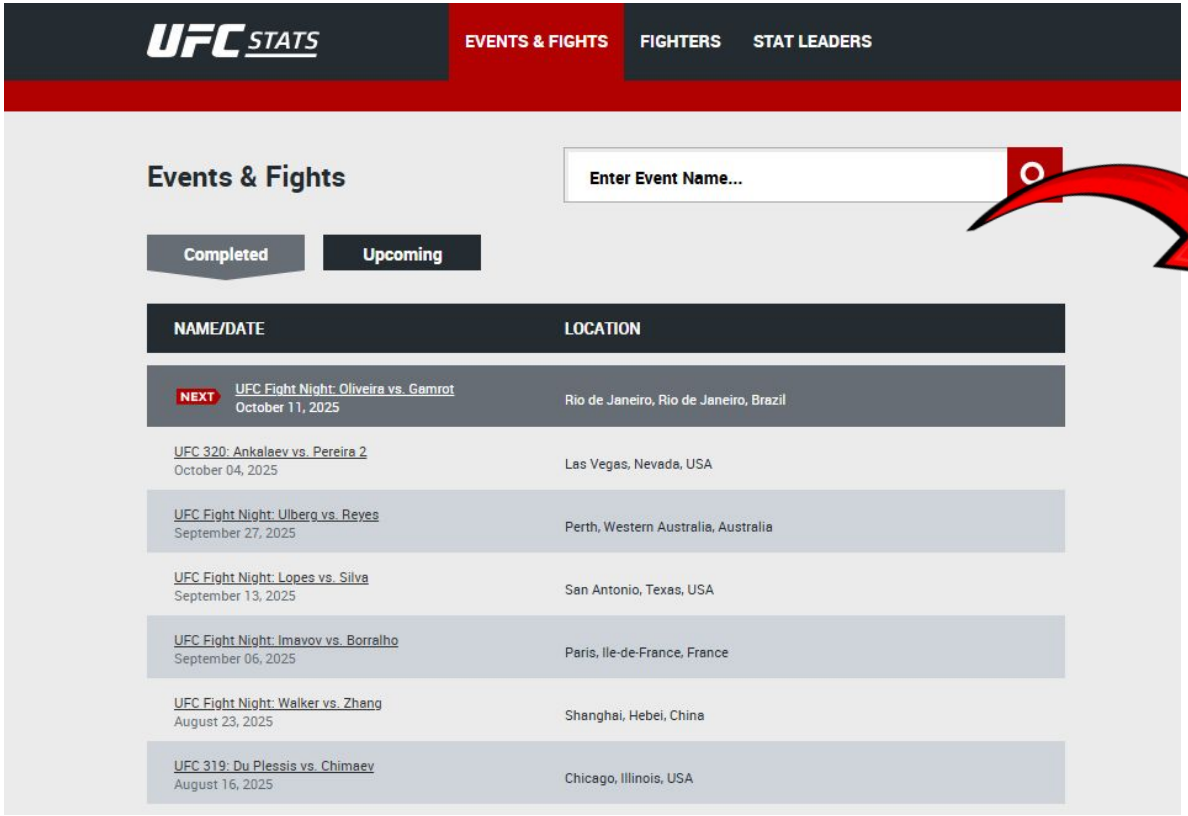
https://alvaromenendez.es/ufc-predictions

https://github.com/DKeAlvaro/UFC-Scraper-ML

Steps

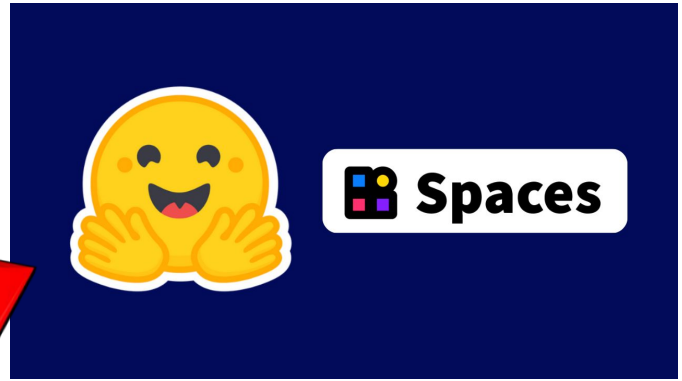
- Scrape data from https://ufcstats.com/
- Build binary classifier models
- Use model's predictions in frontend
- Cron Job to update

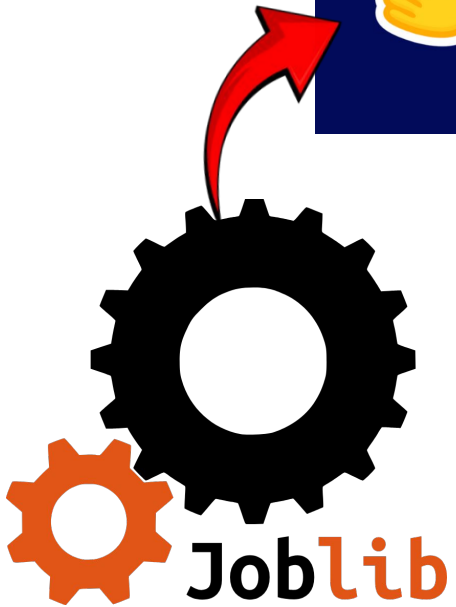
Scraping

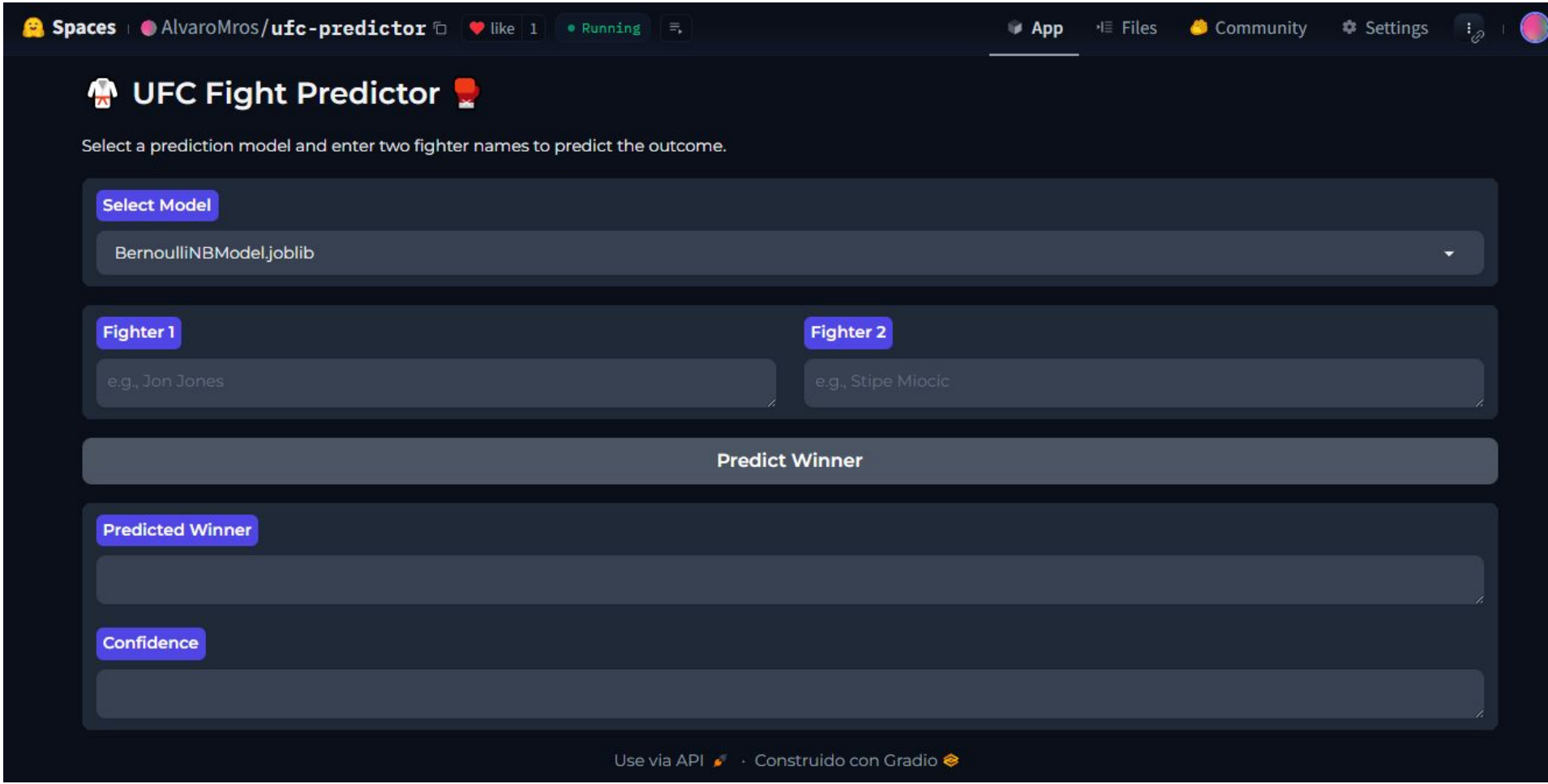


ufc_fights.csv
ufc_fighters.csv

Modelling:






Spaces
AlvaroMros/ufc-predictor
like 1
Running
App
Files
Community
Settings
UFC Fight Predictor
Select a prediction model and enter two fighter names to predict the outcome.
Select Model
BernoulliNBModel.joblib
Fighter 1
e.g., Jon Jones
Fighter 2
e.g., Stipe Miocic
Predict Winner
Predicted Winner
Confidence
Use via API · Construido con Gradio

```python
class BaseModel(ABC):
    """
    Abstract base class for all prediction models.
    Ensures that every model has a standard interface for training and prediction.
    """
    @abstractmethod
    def train(self, train_fights):
        """
        Trains or prepares the model using historical fight data.

        :param train_fights: A list of historical fight data dictionaries.
        """
        pass

    @abstractmethod
    def predict(self, fight):
        """
        Predicts the winner of a single fight.

        :param fight: A dictionary representing a single fight.
        :return: The name of the predicted winning fighter.
        """
        pass
```

```
pipeline = PredictionPipeline(
     models=MODELS_TO_RUN,
     use_existing_models=use_existing_models,
     force_retrain=force_retrain
 )
```