A photograph of a solar panel array in a field of tall grass and wildflowers. In the background, several wind turbines stand against a backdrop of mountains under a clear sky.

CEEX – Clean Energy Exchange

IP5 22HS_IIT21

Raphael Lüthy & George Rowlands

Table of contents

- Who are we?
- What is CEEX
- Initial State
- Problems and their Solutions
- Learnings
- Future Work
- QnA

```
      sion->getExtensions('total');

      354
      355
      356
      357
      358
      359
      360
      361
      362
      363
      364
      365
      366
      367
      368
      369
      370
      371
      372
      373
      374
      375
      376
      377
      378
      379
      380
      381
      382
      383
      384
      385
      386
      387
      388
      389
      390

      value) {
      !])) {
      Carousel.prototype.get = function (item) {
        this.$items = item.parent().children();
        return this.$items.index(item) || this.$active;
      }
    }

    Carousel.prototype.getItemForDirection = function (direction) {
      var delta = direction == 'prev' ? -1 : 1;
      var activeIndex = this.getItemIndex(active);
      var itemIndex = (activeIndex + delta) % this.$items.length;
      return this.$items.eq(itemIndex);
    }

    Carousel.prototype.to = function (pos) {
      var that = this;
      var activeIndex = this.getItemIndex(this.$active);
      if (pos > (this.$items.length - 1) || pos < 0) return;
      if (this.sliding) return this.$element.one('slide');
      if (activeIndex == pos) return this.pause().cycle();
      return this.slide(pos > activeIndex ? 'next' : 'prev',
        that);
    }

    Carousel.prototype.pause = function (e) {
      e || (this.paused = true);
      if (this.$element.find('.next, .prev').length && $.support.transition.end)
        this.$element.trigger($.support.transition.end);
      this.cycle(true);
    }

    this.interval = clearInterval(this.interval);
    return this;
  }
}
```

Who are we

Raphi & Groog

What is CEEX

Initial State

Initial State

Code Base

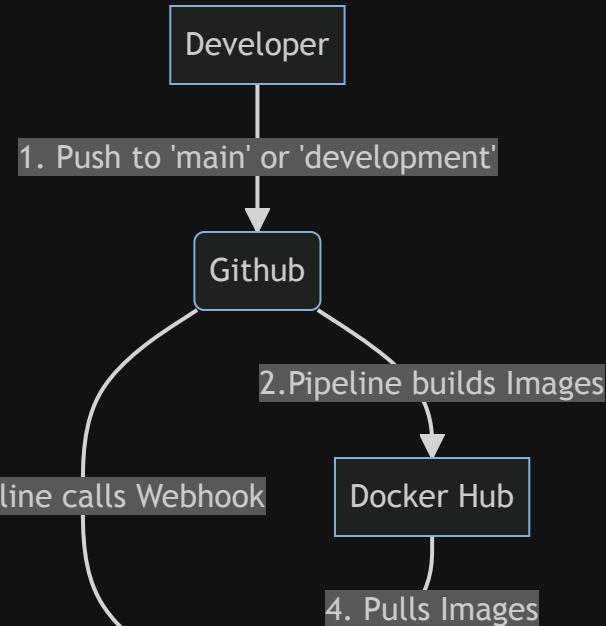
- Hard Coded Components
- Hard Coded Values

```
1 <mat-option value="option1">current Week</mat-option>
2 <mat-option value="option2">current Month</mat-option>
3 <mat-option value="option3">last Month</mat-option>
4 <mat-option value="option4">last 3 Month</mat-option>
5 <mat-option value="option4">last Year</mat-option>
```

```
1 export class DataService {
2     baseUrl = 'http://127.0.0.1:8000/';
3     resetUrl = this.baseUrl + 'auth/request-reset-email';
4     ...
5 }
```

Deployment & Architecture

- Server
- Pipeline
- SSL / Reverse Proxy



Initial State

Hard Coded Components

```
1 <mat-card class="list-element">
2   
3   <div class="list-element__info ">
4     <div class="horizontal_info">
5       <h2>Max Mustermann</h2>
6       
7     </div>
8     <span id="amount">200kWh</span>
9     <div class="horizontal_info last_row ">
10      <div>
11        
12        <span id="distance">300m</span>
13      </div>
14      <span id="horizontal_info">0.18 CHF / kWh</span>
15    </div>
16  </div>
17 </div>
18 </mat-card>
```

Initial State

Hard Coded

```
1  export class DataService {  
2      baseUrl = 'http://127.0.0.1:8000/';  
3      resetUrl = this.baseUrl + 'auth/request-reset-email/';  
4      ...  
5  }
```

Issues

- Maintainability
 - -> Hard to keep track where to change values
- Reusability
 - -> Can't be reused if different values are needed
- Deployability
 - -> Deployments need to change values at runtime

Initial State

Containerization, Pipeline and Deployment

- No Containerization
- No Pipeline
- No Server

📁 backend	Update README.md	5 months ago
📁 frontend	Update README.md	5 months ago
📄 .gitignore	updated gitignore with pycache	6 months ago
📄 README.md	Update README.md	9 months ago

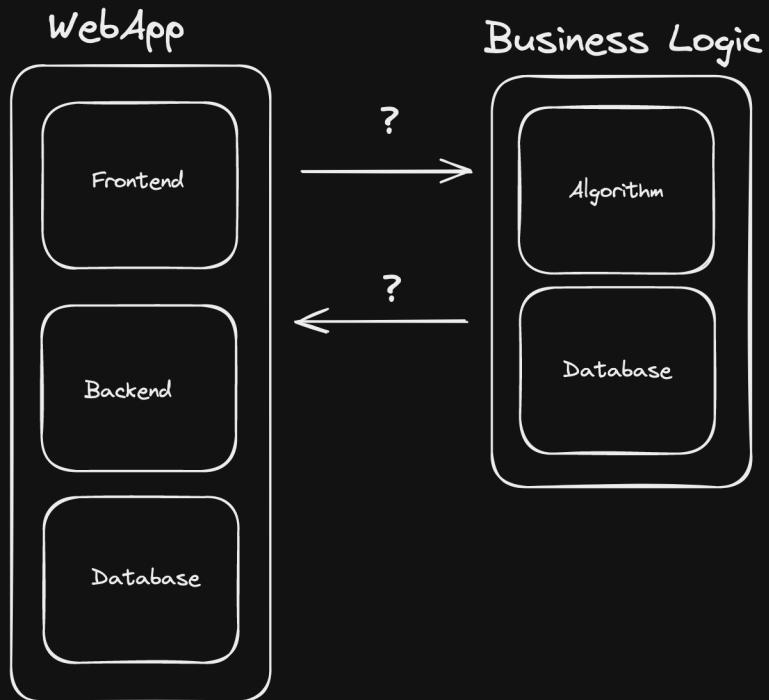


DEVOPS

Initial State

Architecture

- What Side does what?
- Who owns which data?
- How does the communication between Customer and Business Logic work?
- Who owns the Authentication?



Initial State

Usability

- A lot of different fields
- Functionalities with insufficient titles
- No clear distinction of importance
- No clear indication of what interactive

-> Items need more explanation

The screenshot shows the CEEX trading platform interface. At the top, there is a navigation bar with the CEEX logo, a search icon, and tabs for 'TRADING' and 'DASHBOARD'. Below the navigation bar, the main area is divided into several sections:

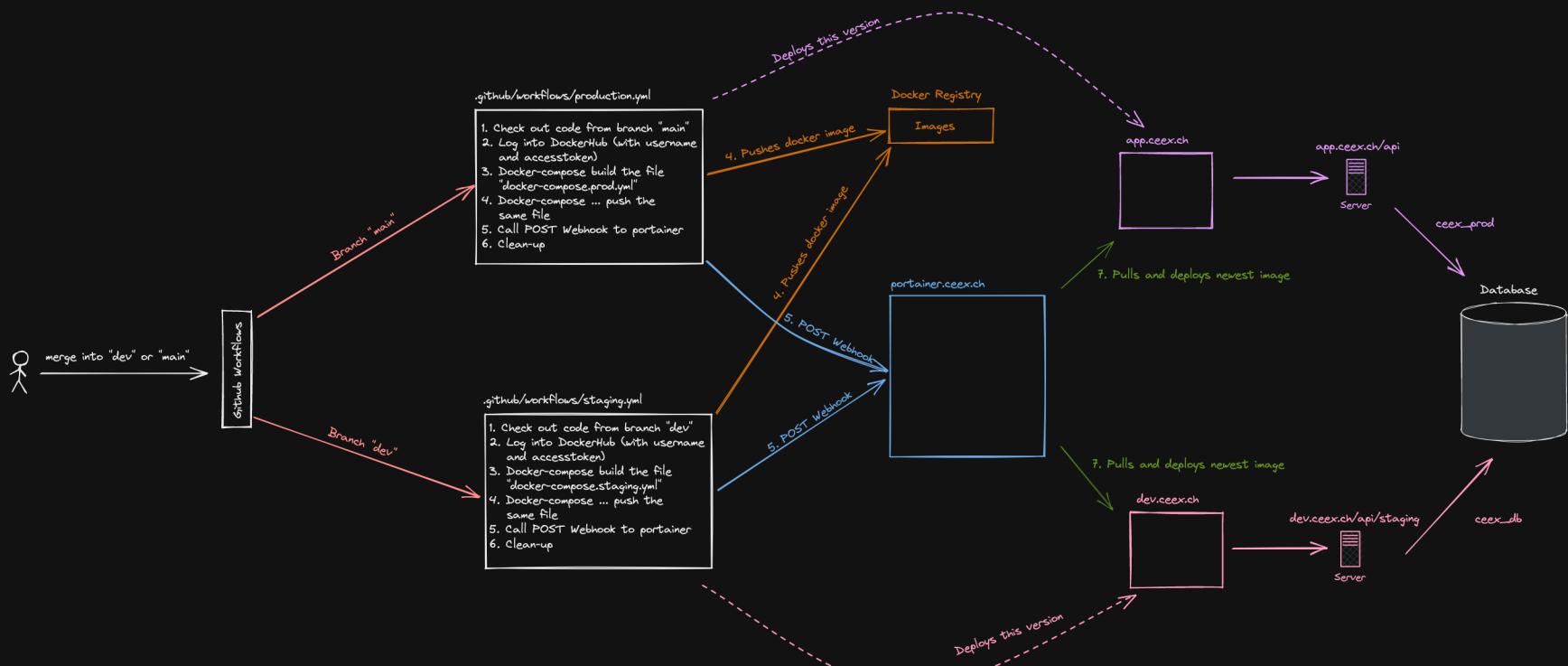
- Your Profile:** This section displays the user's role as 'Consumer / Producer', the distribution system operator as 'IBB Energie AG', the amount of energy required as '300 kWh', and energy surplus as '500 kWh'.
- Price:** A slider for setting the price, ranging from '0 CHF 0.1' to 'CHF 2+'.
- PV system:** A button labeled 'Add' for adding a photovoltaic system.
- Trading Strategy:** This section contains three cards:
 - Market Benefit:** Describes how the user can benefit from a new market entry by IBB Energie AG.
 - Never lose:** Describes the price setting mechanism for the energy produced.
 - Flexibility:** Describes the user's willingness to pay for flexibility services.

Solutions

Deployment & Containerization

Deployment & Containerization

Server setup

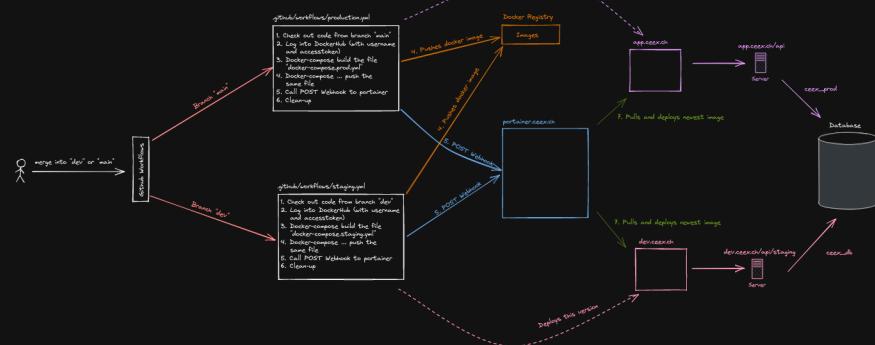


Deployment & Containerization

Server setup

Steps to take:

- Dockerfiles (Containerize Application)
- Server
- Reverse Proxy (traefik)
- Portainer
- Pipeline



Deployment & Containerization

Dockerfiles

Frontend:

```
1 FROM node:14-alpine as build
2 WORKDIR /app
3 RUN npm install -g @angular/cli@13.3.10
4 RUN npm install -g serve@14.1.2
5 COPY ./frontend/package.json .
6 RUN npm install --force
7 COPY ./frontend .
8 RUN ng build --configuration=production
9   --output-path=dist
10 EXPOSE 3000
11 CMD ["serve", "./dist", "-p", "3000"]
```

Backend:

```
1 FROM python:3.10-alpine3.15
2 # stops stdout and stderr streams being buffered
3 ENV PYTHONUNBUFFERED 1
4 WORKDIR /app/backend
5 COPY backend/requirements.txt /app/backend/
6 RUN ["pip", "install", "-r", "requirements.txt"]
7 COPY backend/ceex_api /app/backend/
8 EXPOSE 8000
```

Deployment & Containerization

Server

Actual Server:



[Switch Engines Website](#)

Reverse Proxy:



[Traefik Documentation](#)

Deployment & Containerization

Portainer

- Control of Images / Containers
- Redeployment of latest Image via Webhook
- Stacks

	Name ↓↑ Filter ▾	Type ↓↑	Control	Created ↓↑	Ownership ↓↑
<input type="checkbox"/>	ceex-dev	Compose	Total	2022-11-19 18:01:00 by admin	administrators
<input type="checkbox"/>	ceex-prod	Compose	Total	2022-11-21 12:43:22 by admin	administrators
<input type="checkbox"/>	database	Compose	Total	2022-11-21 13:20:13 by admin	administrators
<input checked="" type="checkbox"/>	portainer	Compose	Limited ⓘ	2022-12-21 11:14:59	administrators
<input checked="" type="checkbox"/>	traefik	Compose	Limited ⓘ	2023-01-13 10:43:43	administrators

Deployment & Containerization

Portainer: "Stack"

- Specifies Docker Images
- Connect the backend to the database
- Expose UI and API to Reverse Proxy

```
1  version: '3.9'
2  services:
3      ui:
4          image: 'raphaelluethy/ceex:ui-prod-latest'
5          networks:
6              - proxy
7      api:
8          image: 'raphaelluethy/ceex:api-prod-latest'
9          networks:
10             - proxy
11             - database
12
13  networks:
14      proxy:
15          name: 'proxy'
16          external: true
17      database:
18          name: 'database'
19          external: true
```

Deployment & Containerization

Pipeline

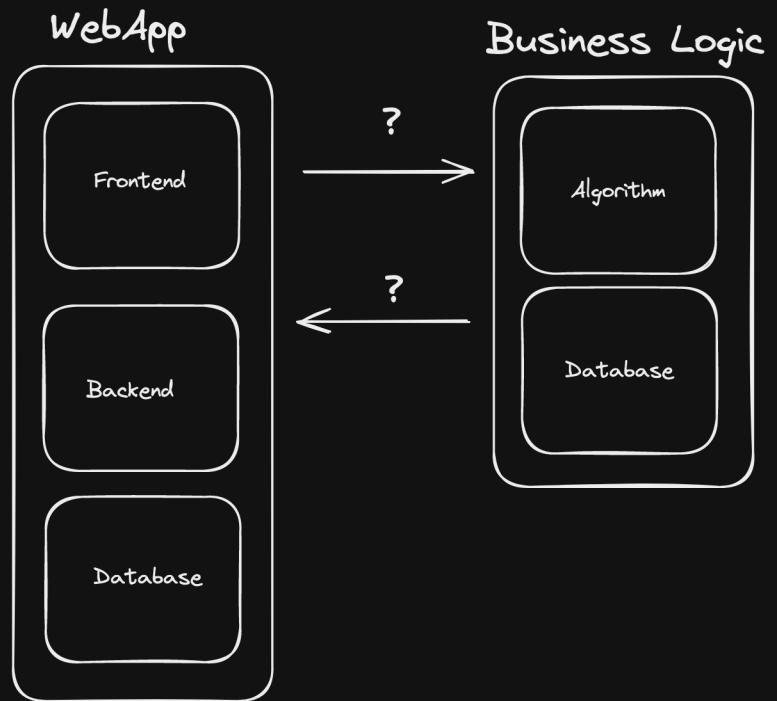
1. Push to Branch "main" (production) or "development" (staging)
2. Spin up Ubuntu VM
3. Checkout the Source Code
4. Login to Docker Hub
5. Build the Docker Image
6. Push the Docker Image to Docker Hub
7. Call Webhook of Portainer
8. Cleanup

```
1  on:  
2    push:  
3      branches:  
4        - main  
5  jobs:  
6    build_push_containers:  
7      runs-on: ubuntu-latest  
8      steps:  
9        - name: Checkout  
10       uses: actions/checkout@v2  
11        - name: Login docker  
12       uses: docker/login-action@v1  
13       with:  
14         username: ${{ secrets.DOCKER_USERNAME }}  
15         password: ${{ secrets.DOCKER_PASSWORD }}  
16        - name: Build Docker image  
17       run: docker compose -f  
18         docker-compose.prod.yml build  
19        - name: Push Docker image  
20       run: docker compose -f  
21         docker-compose.prod.yml push  
22        - name: Call webhook  
23       uses: satak/webrequest-action@master  
24       with:  
25         url: ${{ secrets.WEBHOOK_PROD_URL }}  
26         method: POST  
27       Here comes cleanup...
```

Architecture

Original Problems:

- What Side does what?
- Who owns which data?
- How does the communication between Customer and Business Logic work?
- Who owns the Authentication?



Architecture

Data Agreement

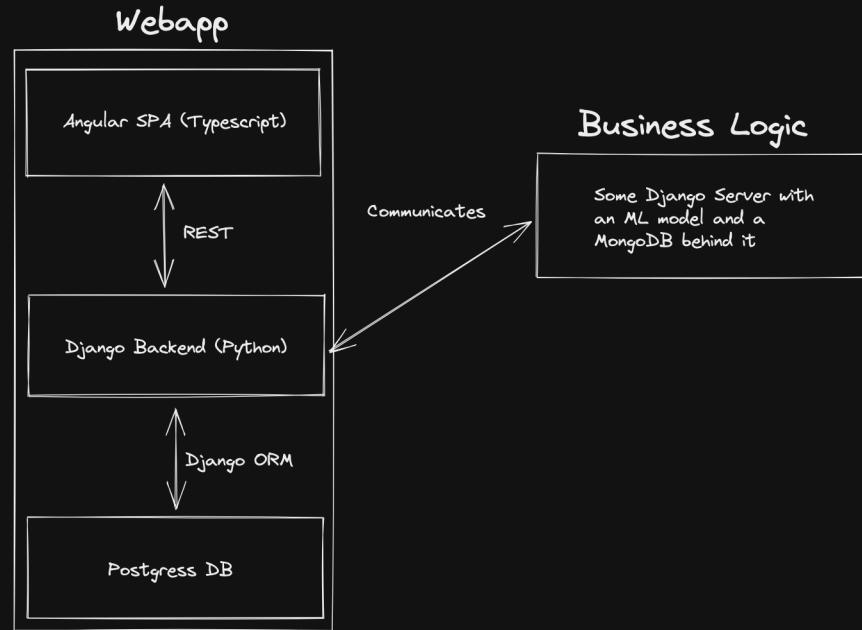
Webapp:

- Own Authentication Layer
- Stores what user belongs to which **Node**
- Provides Proxy for user to communicate with Business Logic

Business Logic:

- Home of the Algorithm
- Stores all the data related to trading

Node: Maps the actual household to the graph of the algorithm



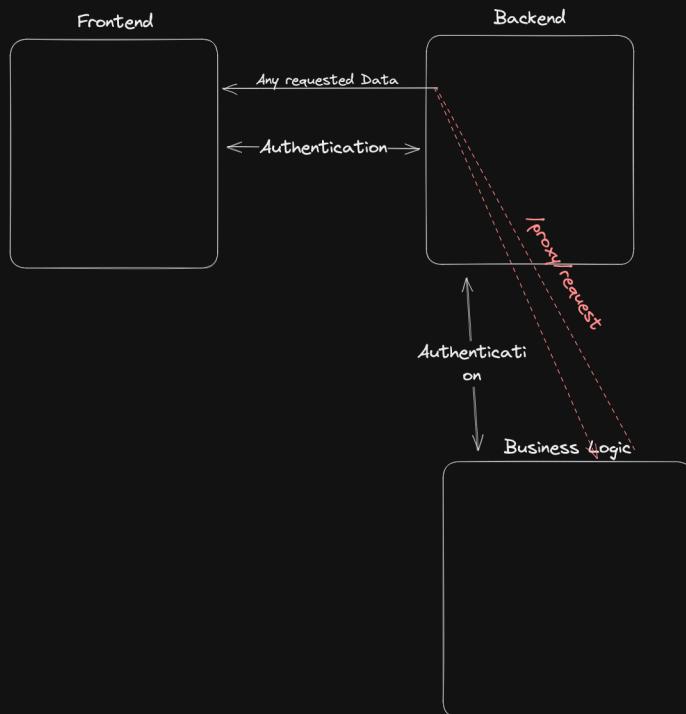
Proxy

Frontend:

```
1  return this.http.post(` ${this.proxyBaseUrl}` , {  
2      url: `node/${nodeId}/`  
3          metrics?startDate=${startDateTime}  
4          &endDate=${endDateTime}` ,  
5      method: 'GET',  
6  });
```

Backend:

```
1  response = requests.request(  
2      method=method,  
3      url=f'{remote_url}{request_url}' ,  
4      allow_redirects=False  
5      )  
6  
7  return JsonResponse(response.json() , safe=False)
```



Thank you for
listening



Any Questions?