

# CEEX – Clean Energy Exchange

IP-5



Students	<i>George Gruffydd Rowlands</i> <i>Raphael Lüthy</i>
Expert	<i>Nitish Patkar</i>
Supervisor	<i>Norbert Seyff</i> <i>Nitish Patkar</i>
Client	<i>Christian Dollfus</i>
Project number	<i>22HS_IIT21</i>

University of Applied Sciences and Arts Northwestern Switzerland, School of Engineering

*Windisch, January 2023*

## **Abstract**

Clean Energy Exchange (CEEX) is a startup supported by the Swiss government to enable customers to trade green, self-produced energy with other customers in their vicinity. This would allow customers to sell their renewable energy to other customers instead of feeding it back into the grid for energy providers to profit off. Lowering the distance from producer to a consumer would allow for a more competitive market and therefore also reduce the basic cost of energy in general. The initial state of the project was a previously finished IP6 thesis, which created a design system and implemented a mostly static web app using Django and Angular.

One of our goals was to make the web app's content more dynamic and create a deployment pipeline, which ensures that the customer always has the most recent stable build. We prepared a proxy so that the business logic, which is being developed at HSLU, can be accessed, and served to the web app. As a temporary placeholder for the data of HSLU a fake data service was implemented and published.

Another goal was to improve the usability of the web app using different tools such as tooltips, adapting the design and layout, adding multilanguage support, and new components that help the user to navigate through the web app more easily.

In summary, our work on the CEEX application has been focused on improving the customer experience, making the website more dynamic and functional, improving the codebase and documentation for future development, and supporting the transition to clean energy for a more sustainable energy system and future.

### **Keywords:**

Angular, CI/CD, Clean Energy, Community, Django, Docker, Exchange, MySQL

# Table of Contents

List of Figures.....	v
Glossary.....	vi
<b>1 Introduction.....</b>	<b>1</b>
1.1 What is CEEEX?.....	1
1.2 What are the goals of this project? .....	1
1.3 What was achieved with this project? .....	1
1.3.1 Deployed Dynamic Web Application .....	1
1.3.2 Enhance UI Look and Feel.....	2
1.4 Structure of This Document.....	2
<b>2 Initial State .....</b>	<b>3</b>
2.1 P1: Hard Coded Components and Values .....	3
2.2 P2: Containerization and Pipeline.....	3
2.3 P3: Server.....	4
2.4 P4: Architecture .....	4
2.5 P5: Usability.....	5
2.6 P6: CSS Styling .....	5
2.7 List of the problems .....	5
2.8 Goals .....	5
2.8.1 Goal 1: Make the website available and usable .....	6
2.8.2 Goal 2: Make the website comprehensive and user-friendly .....	6
<b>3 Solution .....</b>	<b>7</b>
3.1 Dynamic Functionality.....	7
3.2 Data API Agreement .....	7
3.3 Fake Data API.....	8
3.4 Proxy .....	9
3.5 Codebase .....	10
3.6 Deployment.....	10
3.7 Refactoring .....	12
3.8 Libraries .....	13
3.9 Usability .....	13
3.9.1 Internationalization .....	13
3.9.2 Tooltips .....	14
3.9.3 Dialog Windows .....	15
3.9.4 Design .....	18
3.10 Results .....	20
<b>4 Project Management .....</b>	<b>22</b>
4.1 Work packages .....	22
4.1.1 WP1: Analysis work .....	22
4.1.2 WP2: Project Management .....	23
4.1.3 WP3: Prototyping.....	24
4.1.4 WP4: Software Development.....	24
4.1.5 WP5: Validation.....	25
4.1.6 WP6: IP5 Report .....	25
4.1.7 Overall Time Plan .....	26
4.2 Development strategy.....	26
4.3 Requirements Engineering and Validation .....	26
<b>5 Conclusion and Future Work .....</b>	<b>28</b>
5.1 What has been achieved? .....	28

5.1.1	Publicly Available and Interactive Website .....	28
5.1.2	More Comprehensive User Experience .....	28
5.2	<i>Future Work</i> .....	28
5.2.1	Icons .....	28
5.2.2	Addresses .....	29
5.2.3	Map .....	30
5.2.4	Internationalization .....	30
5.2.5	HSLU Data.....	30
5.2.6	Registration .....	30
<b>6</b>	<b>Reflection</b> .....	<b>31</b>
6.1	<i>Teamwork</i> .....	31
6.1.1	Internally .....	31
6.1.2	Externally .....	31
6.2	<i>Learnings</i> .....	31
<b>7</b>	<b>Bibliography</b> .....	<b>33</b>
	<b>Declaration of Authenticity</b> .....	<b>34</b>
	<b>Appendix</b> .....	<b>35</b>
<b>A</b>	<b>Current State of UI</b> .....	<b>36</b>
<b>B</b>	<b>Initial State of UI</b> .....	<b>39</b>

## List of Figures

Figure 1 Screenshot of the initial dashboard code.	3
Figure 2 Overview of the initial architecture.	4
Figure 3 Duplicate code in Dashboard styling.	5
Figure 4 Response from the fake data API.	8
Figure 5 Flow diagram of how a request to the fake data API works.	9
Figure 6 Dockerfile to build the Django backend.	10
Figure 7 Docker Compose file to build and connect the services.	11
Figure 8 Overview Deployment Architecture	12
Figure 9 HTML and BODY elements being used in navbar component	12
Figure 10 Example of not needed div element in login component.	12
Figure 11 Row mixin being used to style the dashboard.	13
Figure 12 Globally defined mixin for working with flex rows and columns.	13
Figure 13 en.json containing keys and value for English content.	14
Figure 14 Navbar where the language can be set.	14
Figure 15 Translation key being used for content in the dashboard.	14
Figure 16 Tooltip in the registration form for Customer ID.	15
Figure 17 Code for the tooltip for the energy required stat on the dashboard.	15
Figure 18 The resulting tooltip on the dashboard.	15
Figure 19 The resulting dialog window.	16
Figure 20 The "learn more" link which opens the dialog window.	16
Figure 21 The Dialog component receiving the reference to itself and injected data.	16
Figure 22 The code to open a dialog window, pass information and handle the "afterClosed" event.	17
Figure 23 The initial state of the profile page.	18
Figure 24 The new implementation of the profile page.	19
Figure 25 Screenshot of Feedback which then was given in a Meeting	20
Figure 26 Short Feedback via Slack	21
Figure 27 Silent / No Feedback from Customer, mostly interacted with Coach / Project Lead	21
Figure 28 Screenshot Workpackage 1.1 from the Project Agreement	22
Figure 29 Screenshot Workpackage 1.2 from the Project Agreement	23
Figure 30 Screenshot Workpackage 2 from the Project Agreement	23
Figure 31 Screenshot Workpackage 3 from the Project Agreement	24
Figure 32 Screenshot Workpackage 4 from the Project Agreement	24
Figure 33 Screenshot Workpackage 5 from the Project Agreement	25
Figure 34 Screenshot Workpackage 6 from the Project Agreement	25
Figure 35 Time Beam of the originally planned time frames	26
Figure 36 Example of a Protocol with Requirements	27
Figure 37 Correct way of using material icons font.	29

## Glossary

Grid	“Electrical grid,” “energy grid” or commonly just the “the grid” refers to the interconnected network that is responsible for delivering electricity from producers to consumers.
PV system	Solar Panel system, used to create electricity out of sunlight.
Portainer	Docker Container Management Solution
GitHub	Hosting Solution for Source Code Version Management
CI/CD	Continuous Integration and Delivery achieved through pipelines
SASS	Extension Library that operates on CSS
CSS	Language used to style websites

# 1 Introduction

## 1.1 What is CEEEX?

In recent years energy prices have risen, according to [1], which made a lot of customers unhappy. These challenges have motivated the development of new technologies and methods for energy generation and distribution, with independent trading communities being a potential solution. These communities will soon be made possible by changes in the law [2] that will open the Swiss electricity market to the public, promoting competition, offering economic advantages to consumers by avoiding additional charges for infrastructure, and strengthening network stability. CEEEX aims to create an open and self-regulated market for communities to exchange clean and renewable energy through its energy trading platform, which enables customers such as private households to buy and sell sustainably produced energy among themselves. CEEEX utilizes a powerful algorithm based on graph technologies<sup>1</sup>, developed with ETH Zurich, to calculate real-time prices based on location, supply, and demand. CEEEX also provides trading strategies that allow consumers and producers to save money by circumventing usage fees from large providers. Using a matching algorithm designed by ETH and implemented at HSLU, the platform matches suppliers to consumers depending on different trading strategies. The focus of this project is on the customer facing application, which contains a web app and a simple backend layer for authentication, and other smaller services that interact with each other.

The initial iteration of CEEEX was a bachelor thesis completed by two students and an intern, which included a style system in Figma, a static Angular<sup>2</sup> prototype for the web app, and a Python backend layer built using the Django<sup>3</sup> framework. However, the focus of the previous project was on design, resulting in a codebase that was not ready for production and not prepared for deployment.

## 1.2 What are the goals of this project?

In the project agreement, we identified two main objectives:

1. To develop a dynamic web application with a robust pipeline for deployment to a publicly accessible website.
2. To enhance the existing user interface by incorporating features that aid navigation within the complex realm of the energy market.

## 1.3 What was achieved with this project?

In this chapter, we will briefly showcase what we accomplished.

### 1.3.1 Deployed Dynamic Web Application

After one month of working our way into the project, we got told by the customer that the highest priority was to have a working website online. Thus, we designed and developed a deployment pipeline to an FHNW owned server. Afterwards, we focused on removing the hard coded content and replaced it with dynamically loaded data.

---

<sup>1</sup> Graph algorithms: <https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/> (19.01.2023)

<sup>2</sup> Angular: <https://angular.io/start> (20.01.2023)

<sup>3</sup> Django: <https://www.djangoproject.com/> (20.01.2023)

### **1.3.2 Enhance UI Look and Feel**

We implemented three important features to enhance the web app's usability: Multilanguage support, tooltips, and dialog windows. We also restructured the web app so that the design was more concise and highlighted some key features to make them more easily accessible and simpler to work with.

## **1.4 Structure of This Document**

- Chapter 2 of this report details the initial conditions of the project and identifies the primary challenges we encountered and identified.
- In Chapter 3, we will examine the strategies and approaches implemented to address these issues.
- Chapter 4 delves into the project management techniques employed during the project.
- Chapter 5 outlines potential areas for future development and expansion and lists what we achieved in this project.
- Finally, Chapter 6 concludes the report with a reflection on the project itself.



## 2 Initial State

After conducting an initial review of the initial state of the project and consulting with our customer, we identified the following problems with the initial state of the application:

### 2.1 P1: Hard Coded Components and Values

All the data and images displayed were hard-coded, instead of being dynamically loaded from a source. This made the web application non-interactive for users and hard to showcase to potential customers and stakeholders. The solution to this problem required all hard-coded values to be identified, rewritten, and abstracted into a new component.

Let's examine the initial state of Figure 1.

```
<div class="numbers">
  <mat-card class="energy-sold dashboard-cards">
    <p>Total energy sold</p>
    <div class="second-row">
      <mat-icon class="card-icon" svgIcon="solar_icon"></mat-icon>
      <h5>1000kWh</h5>
    </div>
  </mat-card>
  <mat-card class="selling-price dashboard-cards">
    <p>Selling price pro kWh</p>
    <div class="second-row">
      <mat-icon class="card-icon" svgIcon="price-change_icon"></mat-icon>
      <h5>0.25 CHF</h5>
    </div>
  </mat-card>
  <mat-card class="energy-bought dashboard-cards">
    <p class="hide-text">Total energy bought</p>
    <div class="second-row">
      <mat-icon class="card-icon hide-icon" svgIcon="solar_icon"></mat-icon>
      <h5 class="hide-text">0 kWh</h5>
    </div>
  </mat-card>
  <mat-card class="purchase-price dashboard-cards">
    <p class="hide-text">Purchase price pro kWh</p>
    <div class="second-row">
      <mat-icon class="card-icon hide-icon" svgIcon="price-change_icon"></mat-icon>
      <h5 class="hide-text">0 CHF</h5>
    </div>
  </mat-card>
</div>
```

Figure 1 Screenshot of the initial dashboard code.

Link to the file at the time of the start of the project: [GitHub Repository](https://github.com/FHNW-CEEX-IP6/clean-energy-exchange-app/blob/ed4705f8335012f253ab7df9ef467efa6378c5ae/frontend/src/app/components/dashboard/dashboard.component.html)<sup>4</sup>

The file shows the code for the cards that are intended to display various metrics on the dashboard. However, these values are not connected to the data that can be retrieved from the backend or business logic, but rather set as text. This issue is found throughout the entire source code, resulting in an excessive amount of code duplication.

### 2.2 P2: Containerization and Pipeline

The initial codebase only contained the basic structures created by the frameworks used, and there were no files for Docker or any other containerization technology. This resulted in several issues,

---

<sup>4</sup> <https://github.com/FHNW-CEEX-IP6/clean-energy-exchange-app/blob/ed4705f8335012f253ab7df9ef467efa6378c5ae/frontend/src/app/components/dashboard/dashboard.component.html> (15.01.2023)

such as compatibility issues which can come up during development due to different devices, different operating systems, or different dependency installations. The initial codebase is also not ready to be deployed as there would be multiple issues when trying to communicate between different services in a different environment, because a lot of environment configurations are hard coded.

### 2.3 P3: Server

No server was used or provided to us in the initial prototype development, which is essential for the final deployment and is necessary for us to acquire and configure.

### 2.4 P4: Architecture

The customer and external teams were not yet certain on how the solution's architecture would appear and had not fully discussed the entire data flow from the client facing application to the back. The initial architecture is shown in Figure 2.

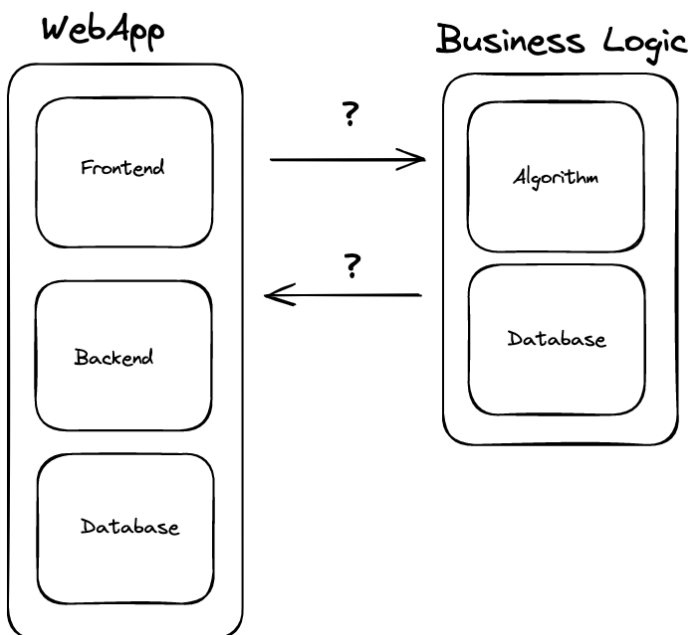


Figure 2 Overview of the initial architecture.

We were initially made aware that there was a web application and a service for business logic. The web application has a frontend and a backend separate from the business logic, which, from an architectural point of view, is less optimal, as the backend usually handles the logic and the permission system together and isn't split in two parts. This approach will lead to issues regarding stability and latency.

Having the logic split from the authentication system also means that there must be another way to be defined for how the business logic will authenticate the requests coming in.

It was also not yet defined which database serves as the source of truth for the data displayed on the webapp. The codebase suggested that the data, such as the data for the graphs, was being duplicated in both databases. This presented a major issue, as the web application would need to synchronize the data between the 2 databases for it to be able to display the correct data.

Generally, having redundant systems like this always add additional complexity for system operations, which is less optimal for this projects' current state.

## 2.5 P5: Usability

The user interface for CEEEX must be clear and self-explanatory, while also providing detailed instructions and information if needed, as the trading of energy is complex, and any inaccuracies may result in financial losses. According to the previous feedback and the clients wishes there was definitely still improvement potential as certain features and concepts are still very confusing for customers.

## 2.6 P6: CSS Styling

A common practice when working with web apps is to use libraries that enhance the styling experience. One such library is SASS which offers some nice features such as variables and functions without changing the overall way CSS is used. This was also included and used in the first increment of the project. However, these features were not made use of which resulted in a lot of duplicate styles across components but also within components. There was also no attempt made of defining global styles that could be used across the entire web app.

```
.dashboard-title {
  display: flex;
  flex-direction: row;
  justify-content: space-between;
  margin: 1rem .5rem 0;
}
.card-title {
  display: flex;
  flex-direction: row;
  justify-content: space-between;
  margin: 0 .5rem 0;
}
```

Figure 3 Duplicate code in Dashboard styling.

## 2.7 List of the problems

The above discussed problems conclude to following list:

- P1: Hardcoded data in components that do not display data from business logic.
- P2: Non-functioning pipeline and lack of preparation for virtualization.
- P3: No server or infrastructure available for deployment.
- P4: Unclear architecture and undefined data ownership.
- P5: Missing information in web application for a complex domain.
- P6: CSS Styling redundancy and SASS features were not used.

## 2.8 Goals

Our main goal was to provide the customer with an application, that was available online and interactive so that it could be showcased to customers, shareholders, and potential investors. Another goal was to improve the usability for users, since the domain of electricity trading is very complex and must be well understood, especially when money is involved. Below we summarize and provide details about the goals we wanted to achieve:

### **2.8.1 Goal 1: Make the website available and usable**

This goal initially did not have the highest priority, however after some initial work on the codebase, it was very apparent that the current state of the web application didn't allow for extensive research on improving the user interfaces usability. Therefore after a few iterations of the project agreement, we decided to make this our highest priority which will also make up the biggest part of our work.

This involves multiple steps:

- Prepare code base for deployment.
- Setup a deployment pipeline and prepare a server to deploy the application to.
- Clean up the codebase:
  - o Replace hard coded values with dynamically loaded values.
  - o Abstract components so that they can be reused.

### **2.8.2 Goal 2: Make the website comprehensive and user-friendly**

The second goal was mostly defined by the initial project proposition, to evaluate and add features that help the user to navigate through the app, such as:

- Tooltips
- Multilanguage
- Modals for more detailed content

The changes are supposed to help the user through the webapp, since energy trading is already very complicated and since money is involved, having the user as good informed as possible is very important.

## 3 Solution

In this chapter we will describe our solution and the methods used to achieve the obtained solution. To solve the before mentioned problems and satisfy our goals we in summary obtained the following list of things to improve:

- Dynamic Functionality, to solve P1 and P4 we agreed upon data ownership with HSLU and made the web app's content more dynamic and functional.
- Usability, to solve P5 we implemented components and changed existing components and design.
- Codebase, to solve P2, P3, and P6 we refactored the code and setup a pipeline which deploys the CEEEX application to a FHNW server.

### 3.1 Dynamic Functionality

In the first implementation almost all the web app's content was statically written in the HTML templates, as described under P1, the only exception to this being the functionality of registering and logging in. In the Django backend, some endpoints had been made with some hard coded data that could be expected to come but from HSLU. However, these endpoints were not hooked up to the web app and the content, structure and formatting of the data had only been based on guesswork, nothing had yet been discussed and defined with HSLU.

### 3.2 Data API Agreement

Once we had a good understanding of the project, what were the problems, what needed to be done etc. we held a meeting with HSLU to discuss the data flow of the application. This included who would own what data, what was to be expected from either parties and in what format. The agreement we came to can be found in reference [3]. We will shortly summarize here what was agreed upon with HSLU:

- We would only be responsible for data related to the user and his authentication, this includes the data that is required to sign up such as passwords, email address etc. but also includes a user's settings in the web app such as language preferences.
- However, we agreed that any data related to the trading process HSLU would be responsible for and we would query via a proxy built into our backend. This includes price histories, information on energy sources such as PV Systems and other metrics.

### 3.3 Fake Data API

Unfortunately, during our project period HSLU was not able to finish their API to provide us the data related to trading. So, we instead implemented a “fake data” API that simulates the data coming from HSLU in the agreed upon format. By implementing this API, we are still able to work with non-static data and implement dynamic functionality into the web app. The RESTful API was created using Next.js<sup>5</sup> and makes use of the Faker<sup>6</sup> library to generate random but meaningful data. For more information on how an API can be implemented using Next.js we refer to the official documentation on Next.js<sup>7</sup>.

```
{
  "averagePrice": 0.243,
  "totalAmount": 564400,
  "amountNetwork": 41,
  "amountCommunity": 59,
  "personalPriceHistory": [
    {
      "timestamp": 1673136000000,
      "price": 0.235
    },
    {
      "timestamp": 1673222400000,
      "price": 0.362421875
    },
    {
      "timestamp": 1673308800000,
      "price": 0.2165625
    },
    ...
  ]
  ...
}
```

Figure 4 Response from the fake data API.

---

<sup>5</sup> Next.js: <https://nextjs.org/> (20.01.2023)

<sup>6</sup> Faker: <https://fakerjs.dev/> (20.01.2023)

<sup>7</sup> Next.js documentation on API routes: <https://nextjs.org/docs/api-routes/introduction> (20.01.2023)

### 3.4 Proxy

We implemented a proxy service in our backend to be able to query the HSLU/fake data API. This is a common and good practice as it decouples the separate services which follows the design principle of “separation of concerns”.

By having a proxy endpoint, we also avoid the need of having to duplicate the HSLU database in our database. If this was not the case the implementation of our service would be much more complex as we would need to always make sure, that the databases are synchronized between each other.

The proxy endpoint is accessed using the “/proxy/requests” path and must contain two fields in the body, one for the desired destination at the fake data API and another one for the HTTP request method. The Django server then builds a new request using the provided information and sends it to the fake data API. The response from the fake data API is then forwarded back to the web app.

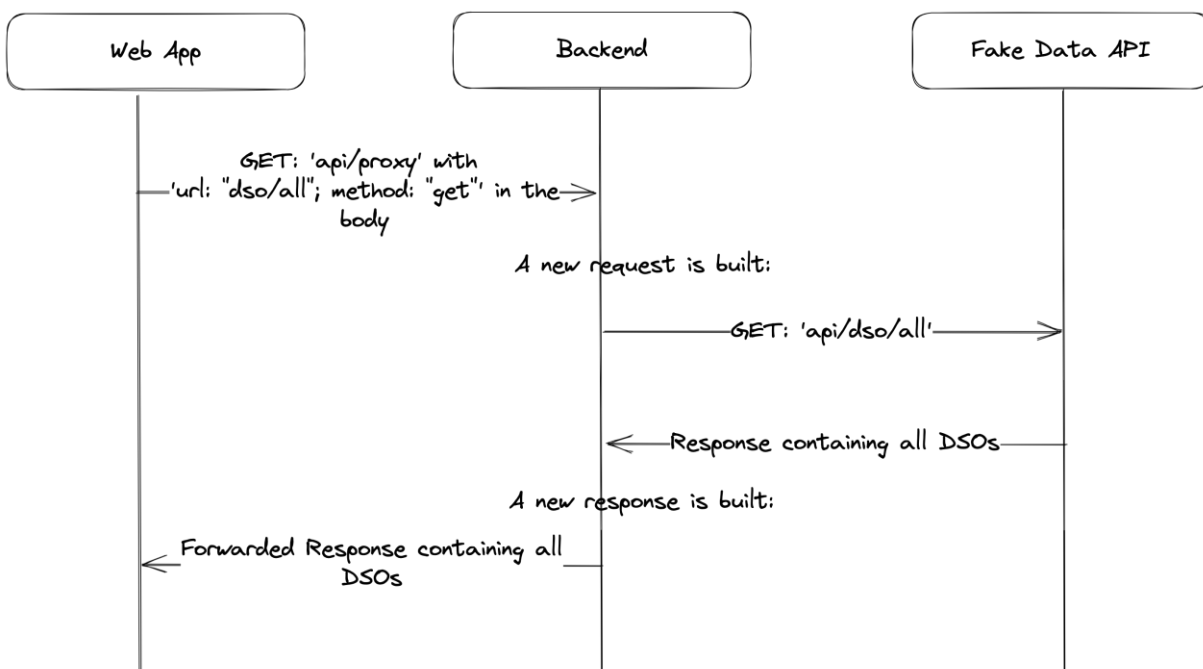


Figure 5 Flow diagram of how a request to the fake data API works.

### 3.5 Codebase

Our work was not only limited to adding new or changing existing features. We also refactored a lot of the code in aid of future development. A pipeline was also implemented to build, containerize, and deploy the application for a better developer experience but also to allow for quick customer feedback on implemented changes and features.

### 3.6 Deployment

As already mentioned, we implanted a pipeline that automatically builds our application consisting of the Angular web app, Django backend and MySQL database. All these services are built within their own container using Docker<sup>8</sup>. This results in less compatibility issues. Compatibility issues can occur during development due to multiple reasons such as: different devices, operating systems, coding setups or dependency installations. Instead, if the application is containerized, each container is its own contained system and is told exactly how and which dependencies and versions to install and how to build and start the service. To further improve the codebase environment variables were also added that are referenced during container creation but also in the services themselves, this avoids redundancy in the code. Environment variables also allow for the services to be easily configured for different environments.

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

Figure 6 Dockerfile to build the Django backend.

The containerized services are then connected using Docker Compose<sup>9</sup>, which also allows us to pass certain environment variables to multiple containers at the same time to configure them and creates a virtual network<sup>10</sup> which all containers are a part of.

---

<sup>8</sup> Docker getting started: <https://docs.docker.com/get-started/> (20.01.2023)

<sup>9</sup> Docker Compose: <https://docs.docker.com/compose/> (20.01.2023)

<sup>10</sup> <https://docs.docker.com/compose/networking/> (20.01.2023)



```

version: '3.9'
services:
  ui:
    platform: linux/amd64
    depends_on:
      - api
    build:
      dockerfile: ./frontend/Dockerfile
    ports:
      - '4200:4200'
    volumes:
      - ./frontend:/app/frontend
      - ut.node_modules:/app/node_modules
  api:
    platform: linux/amd64
    depends_on:
      - db
    build:
      dockerfile: ./backend/Dockerfile
    command: >
      sh -c 'sleep 10 && python manage.py migrate && python manage.py runserver 0.0.0.0:8000'
    environment:
      - DATABASE_HOST=${DATABASE_HOST}
      - DATABASE_NAME=${DATABASE_NAME}
      - DATABASE_USER=${DATABASE_ROOT_USER}
      - DATABASE_PASSWORD=${DATABASE_ROOT_PASSWORD}
      - DEV=${DEV}
      - ENVIRONMENT=dev
    ports:
      - '8000:8000'
    volumes:
      - ./backend/ceex_api:/app/backend
  db:
    ...

```

Figure 7 Docker Compose file to build and connect the services.

This process is a common industry standard as it later also makes deploying the application to a server easier and more flexible. To deploy our application, we implemented the following pipeline, as visualized in Figure 8:

1. A developer merges a new pull request into the development branch and if it is a major version then also merges the development branch later into the main branch.
2. Then the Github workflows<sup>11</sup> build docker images for each of the services and pushes them to Docker Hub<sup>12</sup> at hub.docker.com, a public image registry, using a provided docker account.
3. After pushing the images to Docker Hub, the GitHub workflow calls the Portainer API<sup>13</sup>, which is hosted on a FHNW SwitchEngine<sup>14</sup>, with a webhook to then trigger the deployment of the most recent images. Portainer then pulls the images corresponding to the latest version of the application from the registry and deploys them on the server using Docker Compose, using a specific, for deployment optimized configuration file.
4. Then, depending on which branch was merged or pushed into, the application is either deployed to “app.ceex.ch” (production build) or “dev.ceex.ch” (development preview).

<sup>11</sup> GitHub Workflows: <https://docs.github.com/en/actions/using-workflows> (20.01.2023)

<sup>12</sup> Docker Hub: <https://hub.docker.com/> (20.01.2023)

<sup>13</sup> Portainer Webhook: <https://docs.portainer.io/user/docker/services/webhooks> (20.01.2023)

<sup>14</sup> Switchengine Website: <https://www.switch.ch/engines/> (20.01.2023)

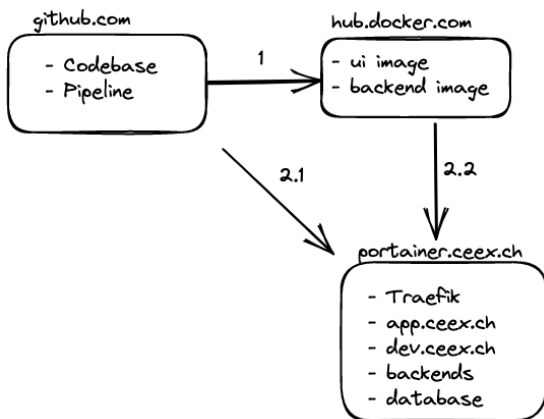


Figure 8 Overview Deployment Architecture

### 3.7 Refactoring

A major part of our work included refactoring the existing code where we followed the camp site rule, to leave the code better than how it was found when making a change.

This resulted in us removing the guessed data structures in the backend, as we had discussed and agreed with HSLU that they would be responsible for those pieces of data, as we later defined in the API Document [3]. We also removed some duplicate and obsolete code along the way whilst trying to keep all endpoints as standardized and simple as possible in the Django server.

However, the main refactoring work was done in the web app. We found that for example a lot of unnecessary DIV elements were being used to just assign a class name to a single element, instead these class names could be assigned straight to the element. Another common mistake made was that the HTML and BODY elements were often included in the HTML template for an Angular component, however, this is not mandatory to add them.

```

<!DOCTYPE html>
<html>
  <body>
    <mat-toolbar class="toolbar" >
      <a routerLink="/">
        <mat-icon class="logo" svgIcon="ceex_logo"></mat-icon>
      </a>
      ...
    </mat-toolbar>
  </body>
</html>
  
```

Figure 9 HTML and BODY elements being used in navbar component

```

<div class="login-screensaver">
  
</div>
  
```

Figure 10 Example of not needed div element in login component.

We also completely changed the styling process by correctly using SASS features and global styles in aid of resolving P6. As mentioned, we found that a lot of styling was being duplicated across elements and components. So, we instead created commonly used classes as part of the global styles, such as card-title and card-value. We also defined SASS variables. As example, for the green accent color, we added such a variable which then can be imported and used throughout the frontend. Using SASS variables is especially handy for theming the web app without having to change the same value

in multiple places. SASS also offers so-called “mixins”<sup>15</sup> which are like functions that can be called/included to further remove duplicate code. These mixins were especially useful for working with flex columns and rows but also for defining fonts.

```
@mixin row($content: flex-start, $items: stretch)
{
  display: flex;
  flex-direction: row;
  justify-content: $content;
  align-items: $items;
}

@mixin col($content: flex-start, $items: stretch)
{
  display: flex;
  flex-direction: column;
  justify-content: $content;
  align-items: $items;
}
```

Figure 12 Globally defined mixin for working with flex rows and columns.

```
@import "styles/mixins";

.stats {
  @include row(space-between);
  gap: 1rem;
}
```

Figure 11 Row mixin being used to style the dashboard.

## 3.8 Libraries

Before we started with development, we made sure that all libraries were up to date. Whilst doing this we realized the map library which was being used had been deprecated for quite a while and therefore needed to be replaced. We also removed some libraries that were added presumably during the experimentation or learning phase of the prototype development and were therefore no longer needed. By updating libraries and removing unused ones we could work with more stable libraries and achieve a slimmer build.

## 3.9 Usability

From previous feedback on the prototype and our own analysis we found that the usability of the web app required improvement. The web app requires improvement in usability, as a lot of users found that the trading of energy between households is a complex task and requires a lot of knowledge in the required fields. However, users of CEEX should not need to know the ins and outs of the system, instead it should be abstracted and kept as simple as possible. The CEEX web app is also not meant to be interacted with as often as a stock exchange, prices and settings do not need to be constantly monitored as they are less volatile, and the trading strategies designed by HSLU take over the trading process for the user.

### 3.9.1 Internationalization

To make the web app more user-friendly and accessible, it offers the user the choice of either English, French, or German. Not only does this aid the user in using the web app more comfortably, as it might be in their mother tongue or a language they are comfortable with, but also opens the possibility for new customers that previously were not interested in CEEX due to the language barrier and the daunting task of working with a tool in a less familiar language.

The internationalization was implemented using the ngx-translate<sup>16</sup> library for Angular. The library can be used to define content in multiple different languages and dynamically change the content of the frontend at runtime unlike the Locales<sup>17</sup> library which builds a separate web app for each language.

---

<sup>15</sup> <https://sass-lang.com/documentation/at-rules/mixin> (20.01.2023)

<sup>16</sup> ngx-translate: <http://www.ngx-translate.com/> (20.01.2023)

<sup>17</sup> Locales by Angular: <https://Angular.io/guide/i18n-common-overview> (20.01.2023)

To internationalize the web app of CEEX, a JSON file is defined for each language: “en.json”, “de.json”, and “fr.json”. Then the so-called TranslationModule is added to the root of the application. The JSON files contain the same keys but have different values corresponding to the translation of the content in that language. Because the translations are written in JSON, they can be nicely structured to make the code more readable.

```
{
  "dashboard": {
    "stats": {
      "energy_required": "Energy Required",
      ...
    },
    ...
  },
  ...
}
```

Figure 13 en.json containing keys and value for English content.

The keys pointing to the translations can then be referenced in the HTML templates of the web app components and fed into a pipe<sup>18</sup> function to translate them using the currently active language setting, which is set using the dropdown in the navigation bar. However, making the web application suitable for multilanguage support did have the consequence of having to change the font, as the originally picked font, Dongle, did not support the special symbols “öäü”, which are commonly used in German. Instead, we decided to use the Roboto font as this is the standard material theme font<sup>19</sup> and was therefore already being used in the application.



Figure 14 Navbar where the language can be set.

```
<span class="card-title">
  {{ "dashboard.stats.energy_required" | translate }}
</span>
```

Figure 15 Translation key being used for content in the dashboard.

### 3.9.2 Tooltips

Tooltips are a great way to give a user more information about a feature without crowding the web app with information and then decreasing the usability again by making things hard to find. For example, in user tests with the prototype users gave the feedback that certain fields were hard to understand, such as the “Customer ID” field in the registration form. They were confused as to what was required to be entered and were unsure as where they could find the required information. By adding a tooltip here, we can quickly and elegantly provide with a short explanation.

<sup>18</sup> <https://www.knowledgehut.com/blog/web-development/pipes-in-Angular> (20.01.2023)

<sup>19</sup> <https://m3.material.io/styles/typography/overview> (20.01.2023)

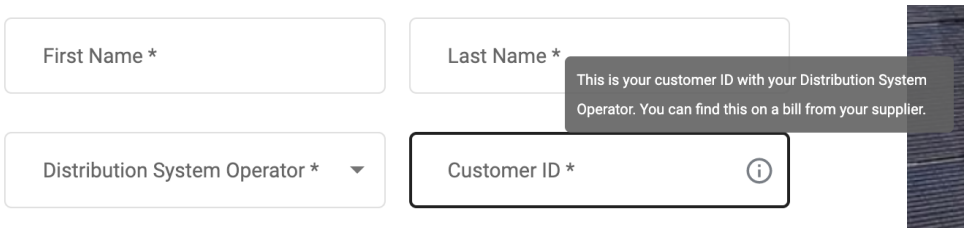


Figure 16 Tooltip in the registration form for Customer ID.

The Tooltips were implemented as a component, so that they could be reused throughout the web app. The material theme already offers tooltips, however these were not quite satisfactory, as our tooltips need to have multi-language support. So, we based our component off the material theme tooltip but enhanced it according to our requirements. The component can then be used like any other component in the html template. It takes the contents translation key as an attribute so that it can be dynamically translated.

```
<tooltip
  [translationId]=" 'dashboard.stats.energy_required_tooltip' "
></tooltip>
```

Figure 17 Code for the tooltip for the energy required stat on the dashboard.

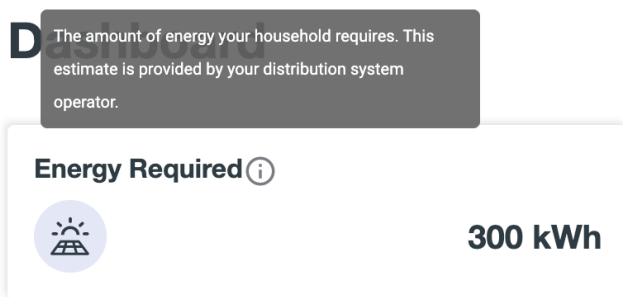


Figure 18 The resulting tooltip on the dashboard.

For more examples of where tooltips were used, we refer to Appendix A, that shows the current state of the web app's UI.

### 3.9.3 Dialog Windows

In certain circumstances a tooltip is not enough to convey information to the user. This could be the case if the text is too long or if the user would benefit of a visual explanation such as an image or video. This for example is the case for the trading strategies, as they are all very different and can be very interesting to people that would like to know the ins and outs of the system but also for someone who would just like some general information on CEEX. A dialog window, or also commonly called modal or popup, is a small window, that overlays the current page and contains additional content.

Currently the dialogs are only used to give additional information for the trading strategies which can be opened with the “learn more” link.

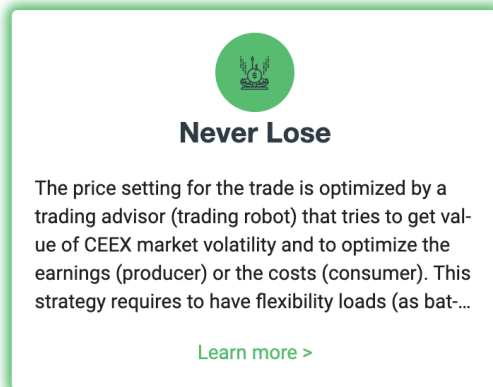


Figure 20 The "learn more" link which opens the dialog window.

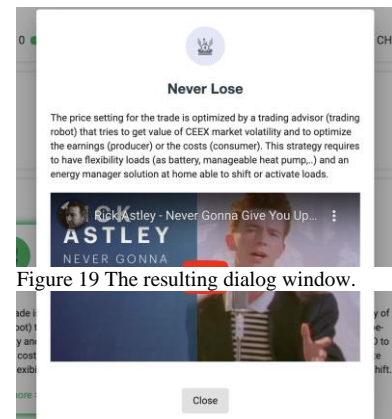


Figure 19 The resulting dialog window.

However, in the future dialog windows could also be used in the previously mentioned example for the customer ID field in the registration to show an example image of a bill and where the customer ID is located. They are also commonly used as containers for input forms. This could be applied for example to the form that is required to put information about an energy source in if a user would like to be a producer, or for a user to change his password or address.

To implement the dialog windows, we used the built-in dialogs from the material theme, as it is well implemented, and it already fulfills all our needs. The material theme dialog windows make use of the observable pattern<sup>20</sup> like many other implementations in Angular. So, knowing this design pattern is highly suggested.

Each dialog window is a separate component, just like any other Angular component with SASS styling and a HTML template for structure. In addition, the dialog component receives a “MatDialogRef<DialogComponentName>” parameter, a reference to itself and it also can be passed different content to make the displayed content dynamic.

```
export class TradingStrategyDialogComponent {
  constructor(
    @Inject(MAT_DIALOG_DATA)
    public data: { title: string; description: string; icon: string },
    public dialogRef: MatDialogRef<TradingStrategyDialogComponent>
  ) {}

  onClose() {
    this.dialogRef.close(Dialog was closed!);
  }
}
```

Figure 21 The Dialog component receiving the reference to itself and injected data.

The reference to itself is then used to close the window and send back any information to the callee. In order to display the modal, the callee needs to pass the dialog component to the “dialog.open()”

<sup>20</sup> [https://en.wikipedia.org/wiki/Observer\\_pattern](https://en.wikipedia.org/wiki/Observer_pattern) (20.01.2023)

function with any data to be passed. This function then returns an observable object, where we can subscribe to a callback function for a specific event, which is called when the user closes the dialog window. This can be used to receive any data from the closed modal.

```
openDialog(event: MouseEvent) {  
  const dialogRef = this.dialog.open(TradingStrategyDialogComponent, {  
    width: '30rem',  
    data: {  
      ...  
    },  
  });  
  dialogRef.afterClosed().subscribe((result) => {  
    console.log(`Dialog result: ${result}`);  
  });  
}
```

Figure 22 The code to open a dialog window, pass information and handle the "afterClosed" event.

### 3.9.4 Design

The design of a website can have a great effect on its usability<sup>21</sup>. The previous implementation already had a good design system which was created in Figma. However, the implementation of the design was lacking as it was not consistent across pages or components and certain features could be highlighted better.

Consistency in design such as spacing, coloring and layout can aid the user when using the website, as they become more familiar with it and find information where they expect it to be. Using consistent typesetting (font and font sizing) does not only look better but if done correctly, small things such as headings can lead the users view to the correct place. It is also important that key features are highlighted either with icons or other means such as shadows or color. These features then stand out more and can be found easier.

In our implementation of the design, we laid heavy focus on making everything as consistent as possible. This allowed us to refactor the CSS styling and HTML templates heavily, so that they can be reused in multiple occasions and do not require each component to define its own styling. For further details on the refactoring process, we refer to the previous chapter on Codebase.

We also decided to use the CEEX green color more often throughout the web app to highlight key features in order to aid the user in finding key features quicker.

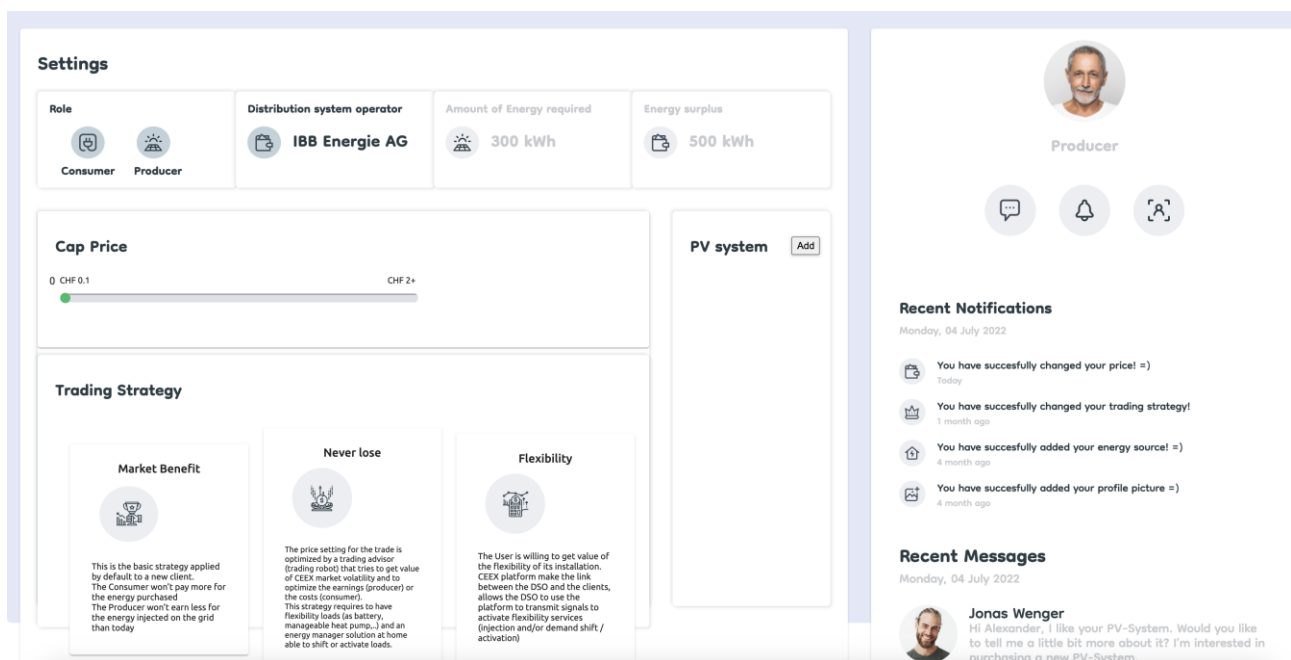


Figure 23 The initial state of the profile page.

<sup>21</sup> Adobe Design Principles: <https://helpx.adobe.com/indesign/how-to/design-principles.html>



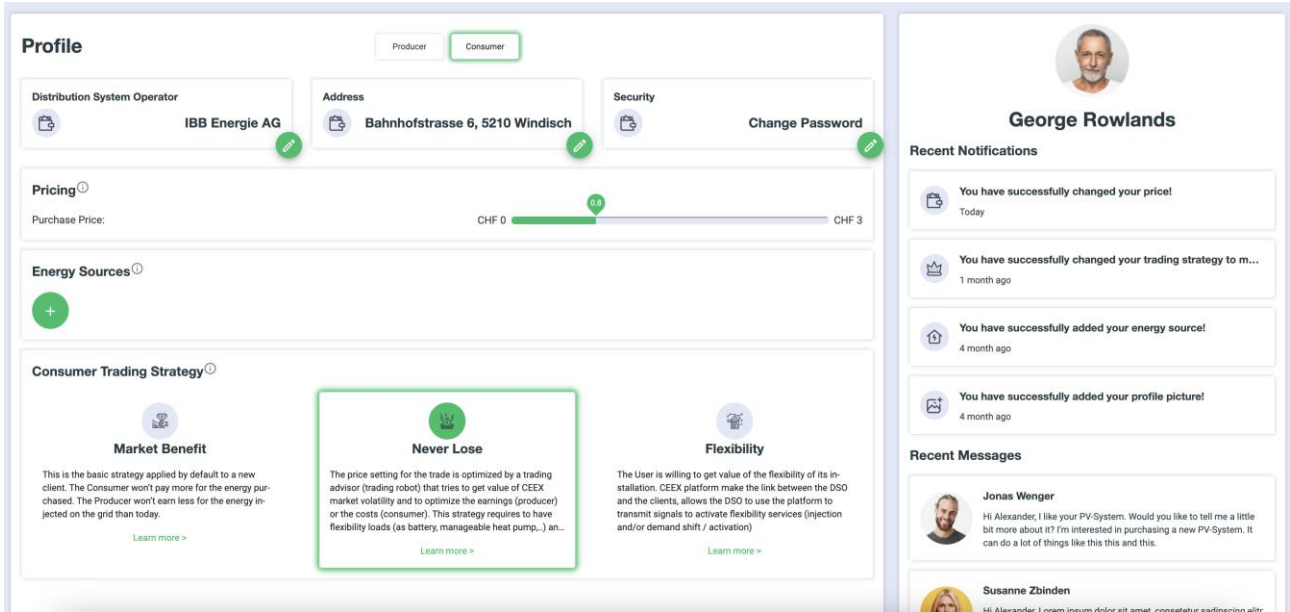


Figure 24 The new implementation of the profile page.

To see more differences in the design we refer to comparing Appendix A, the current UI to Appendix B the old UI.

### 3.10 Results

We were easily able to validate our work thanks to the implemented live deployment. Once the new features were implemented, we could see our changes at dev.ceex.ch and receive feedback from the client via meetings or slack messaging.

As shown in Figure 25, if there was any more detailed feedback to be given, a spontaneous meeting was scheduled.

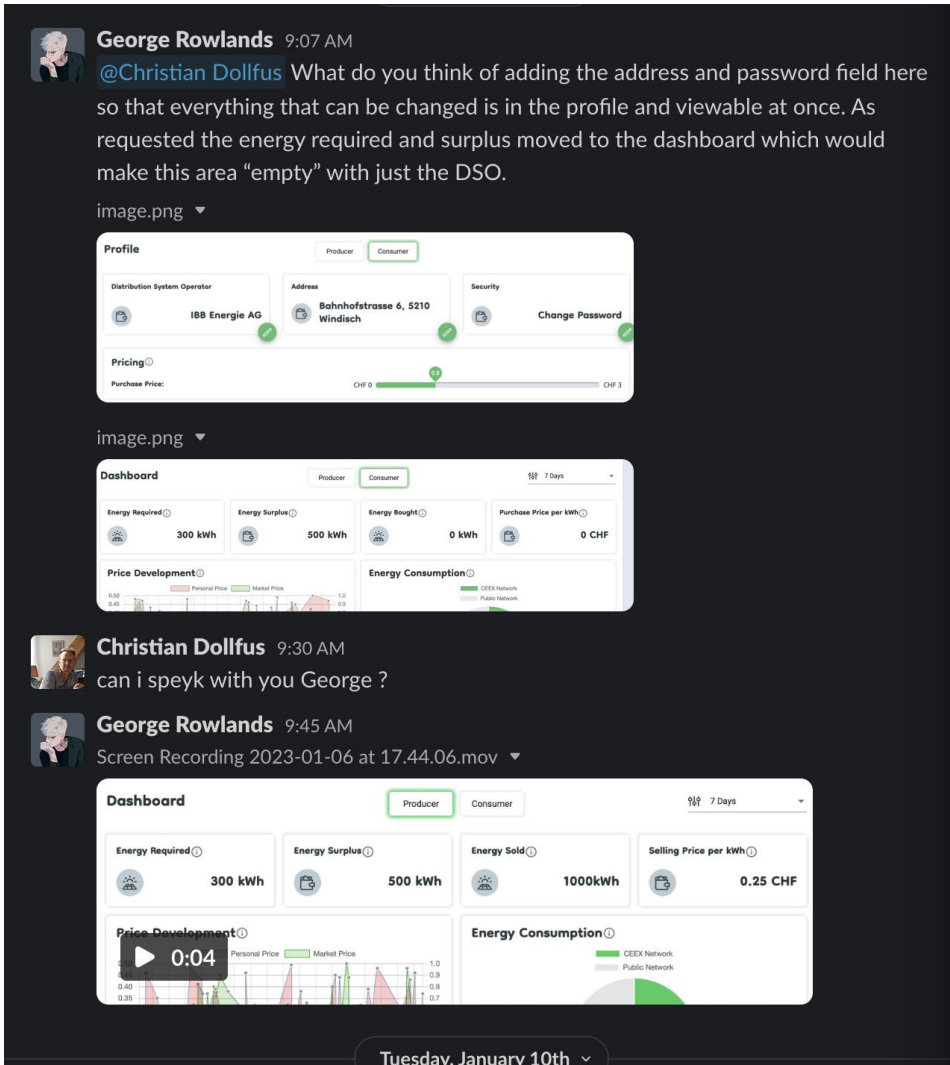


Figure 25 Screenshot of Feedback which then was given in a Meeting

Normally, we received either very short answers (Figure 26) or just silent agreement (Figure 27). When the customer was asked to give feedback, during meetings, we hadn't received any negative responses.

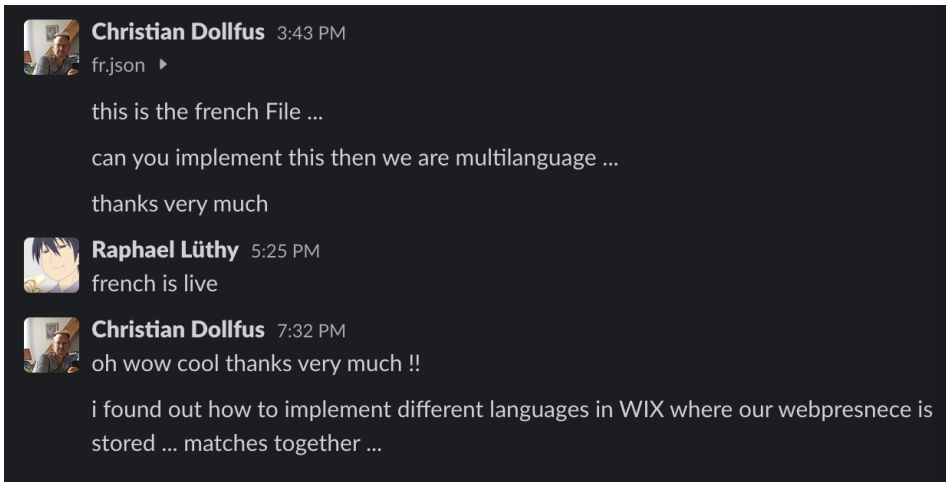


Figure 26 Short Feedback via Slack

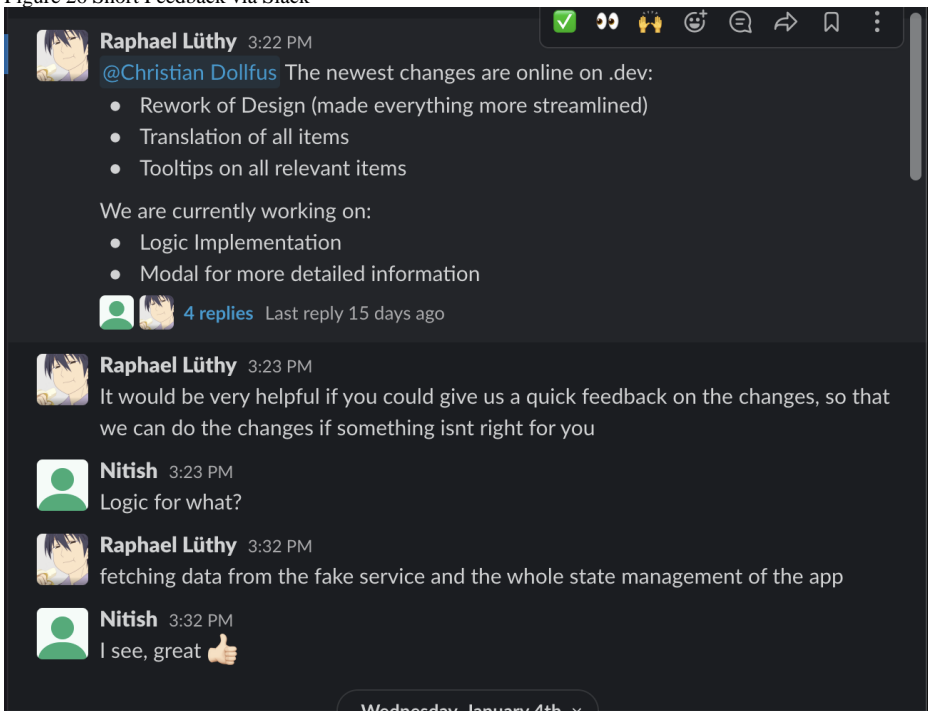


Figure 27 Silent / No Feedback from Customer, mostly interacted with Coach / Project Lead

We did not manage to perform any in-depth used testing due to time constraints and lack of users, as the website has no clients onboarded yet.

## 4 Project Management

Even though we had originally planned and defined the timeframes for each work package, we found that it was hard to stick to these timeframes. This is due to the fact that one of us was doing an exchange semester at Seoul National University and the required coursework and time difference made it difficult to stay within the set boundaries. This resulted in an asynchronous workflow, which we managed to overcome by separating tasks in a way that allowed us to work independently. We often met and discussed the projects and the tasks to be done, but we minimized dependencies between each other, which ultimately led to a successful outcome.

### 4.1 Work packages

During the project, we revised and ultimately settled on several work packages, which we will now discuss:

#### 4.1.1 WP1: Analysis work

<b>AP 1.1: State of the art analysis</b>	
Ziele	Getting a better understanding of what the current state of the art is
Aktivitäten	<ul style="list-style-type: none"><li>• Research on other websites</li><li>• Comparing findings</li><li>• Analysing findings</li><li>• Create learning from the analysed content</li></ul>
Deliverables	<ul style="list-style-type: none"><li>• Report from the research</li><li>• List of found points</li><li>• Implemented modules</li></ul>
Start	26.09.2022
Ende	21.11.2022
Aufwand	10 h

Figure 28 Screenshot Workpackage 1.1 from the Project Agreement

We were able to minimize time for this work package as we were able to use our previous work experience to conclude the necessary decisions that would have been impacted by this research.

<b>AP 1.2: Current State of the App analysis</b>	
Ziele	Getting a better understanding of what the current state of the application is
Aktivitäten	<ul style="list-style-type: none"> <li>• Check Documentation from previous teams</li> <li>• Check out source code</li> <li>• Check current state of deployment</li> <li>• Analyse current state of architecture</li> </ul>
Deliverables	<ul style="list-style-type: none"> <li>• List of issues and features (requirements)</li> <li>• Documentation (if missing)</li> </ul>
Start	26.09.2022
Ende	21.11.2022
Aufwand	20 h

Figure 29 Screenshot Workpackage 1.2 from the Project Agreement

This work package took longer than we had originally allocated for it, as the initial codebase was difficult to understand. This was also affected by the initially unclear project goal and the complex domain of electricity trading. Despite this, we were able to deliver the planned deliverables and we believe that allocating more time to this package was the right decision as it saved us time in the long run.

#### 4.1.2 WP2: Project Management

<b>AP 2: Project Management</b>	
Ziele	Having a clear overview on where the project stands at any time
Aktivitäten	<ul style="list-style-type: none"> <li>• Manage Jira</li> <li>• Log time</li> <li>• Conduct meetings with supervisors</li> <li>• Create a roadmap</li> <li>• Coordinate with external teams: <ul style="list-style-type: none"> <li>○ Marketing</li> <li>○ HSLU Algorithm Team</li> </ul> </li> </ul>
Deliverables	<ul style="list-style-type: none"> <li>• Roadmap</li> <li>• Meeting Notes</li> <li>• List of open Jira issues</li> </ul>
Start	19.09.2022
Ende	20.01.2023
Aufwand	30 h

Figure 30 Screenshot Workpackage 2 from the Project Agreement

We tried to run with the set activities and deliverables, but due to the turbulent start and our special circumstances we decided to minimize any work on project management as the roadmap and the general management changed frequently enough to warrant not spending any more time on this matter.

We attempted to integrate Jira into our workflow, but as a two-person team, we decided to abandon it as it seemed to be an inefficient use of our already limited time and since it was not specifically required by the IP5 coach team or the customer, it was really low on our priority list. Instead, we

managed our workflow by creating meeting protocols and having more detailed discussions over WhatsApp. As experienced developers, we concluded that this approach would suffice as we trust each other's skills.

### 4.1.3 WP3: Prototyping

<b>AP 3: Prototyping</b>	
Ziele	Creating an initial proposal for a feature to implement
Aktivitäten	<ul style="list-style-type: none"> <li>• Create Wireframes / Concepts</li> <li>• Hold Meetings with customer</li> <li>• Adapt Wireframe / Concepts</li> </ul>
Deliverables	<ul style="list-style-type: none"> <li>• Wireframes / Concepts</li> </ul>
Start	19.09.2022
Ende	31.12.2022
Aufwand	10 h

Figure 31 Screenshot Workpackage 3 from the Project Agreement

We spent way less time on prototyping as we primarily conducted it during meetings with the customer or by sending screenshots of the current website with a placeholder for the proposed change. With some answered questions through slack the prototypes were accepted without a lot of issues. Overall, we agree with our decision to minimize prototype work, as we focused mostly on functionality then design.

### 4.1.4 WP4: Software Development

<b>AP 4: Software Development</b>	
Ziele	Create an accessible webpage and implement features which were agreed on with the customer
Aktivitäten	<ul style="list-style-type: none"> <li>• Create concepts for features with customers</li> <li>• Setup CI/CD</li> <li>• Create proper mock data for showcase</li> <li>• Implement new features</li> <li>• Make architectural changes</li> <li>• Verify changes by testing</li> <li>• Refactor code</li> </ul>
Deliverables	<ul style="list-style-type: none"> <li>• Pipeline with reports</li> <li>• Publicly available Website</li> <li>• Source Code</li> </ul>
Start	24.09.2022
Ende	13.01.2023
Aufwand	140 h

Figure 32 Screenshot Workpackage 4 from the Project Agreement

The software development took up the majority of our allocated time. We were able to complete our deliverables and carry out the planned activities.

#### 4.1.5 WP5: Validation

<b>AP 5: Validation</b>	
Ziele	Validating implementation with the customer
Aktivitäten	<ul style="list-style-type: none"> <li>• Repeating process:               <ul style="list-style-type: none"> <li>- Deploy changes to development deployment</li> <li>- Get feedback on changes</li> <li>- Adapt according to the feedback</li> </ul> </li> <li>• Deploy finished change to production deployment</li> <li>• Get acceptance from customer for the changes</li> </ul>
Deliverables	<ul style="list-style-type: none"> <li>• Wireframes / Concepts</li> <li>• Live Previews</li> <li>• Analysed answers with new learnings</li> </ul>
Start	19.09.2022
Ende	31.12.2022
Aufwand	30 h

Figure 33 Screenshot Workpackage 5 from the Project Agreement

During the project, we were able to significantly reduce the validation time as the customer only had to provide minimal feedback on the latest changes. Additionally, much of the work was focused on architecture, making it challenging to validate the technical details. Despite this, the customer was satisfied with the outcome of our pipeline and we determined that further validation was not necessary. We also did not conduct validation with end customers as we primarily focused on implementing functionality, and the UI changes we made were mainly to correct logical errors and adhere to common practices such as including tooltips. We concluded that the customer would suffice as validation.

#### 4.1.6 WP6: IP5 Report

<b>AP 6: IP5 Report</b>	
Ziele	Delivering a proper academic report
Aktivitäten	<ul style="list-style-type: none"> <li>• Take notes for the report in the semester</li> <li>• Writing the report</li> </ul>
Deliverables	<ul style="list-style-type: none"> <li>• Report</li> <li>• Appendix</li> </ul>
Start	19.09.2022
Ende	13.01.2023
Aufwand	120 h

Figure 34 Screenshot Workpackage 6 from the Project Agreement

For the report, we had to do the most part at the end of the project, since a lot of key factors changed suddenly during this project, which could have led to a lot of work becoming outdated. We still kept notes, mostly for technical documentation, which we then could polish up and add to the report. Due to planning mistakes regarding the winter break and upcoming exams, we were forced to work on the report way later than we should have. This resulted in a hectic writing and review process which was preventable.

### 4.1.7 Overall Time Plan

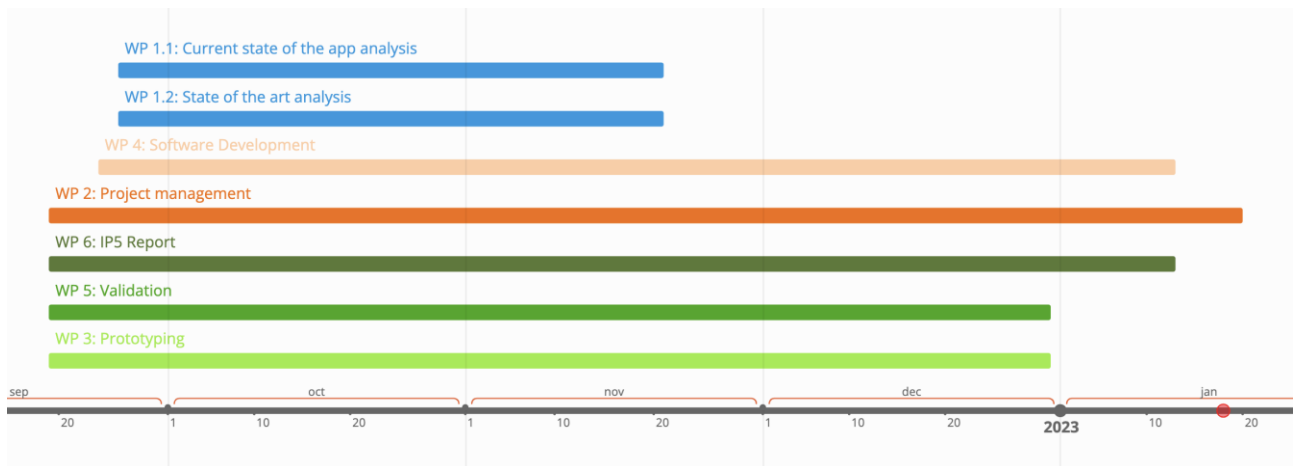


Figure 35 Time Beam of the originally planned time frames

The above are the original timeframes that we planned. We were mostly able to meet the deadlines, but occasionally had to extend the end date slightly.

## 4.2 Development strategy

Initially, we decided to use the agile development approach as we were both familiar with it from the IP34 project. However, as the initial phase of the project progressed, it became increasingly difficult to achieve this approach due to the circumstances we faced. One major factor was the time spent trying to identify the problem we needed to solve; we realized we needed to fix the underlying structure of the project before making any improvements to the user interface. This resulted in the development period starting much later than originally planned. Additionally, constant changes in requirements, priorities and the sudden need for a deployed website added to the challenges. Furthermore, the time zone difference between team members and the workload of the exchange semester made it difficult to conduct effective stand-up meetings and led to initially uneven distribution of work. All these issues prompted us to adopt a more iterative approach, which allowed us to have a flexible time schedule, easily showcase changes and make adjustments through spontaneous meetings and the pipeline we set up.

## 4.3 Requirements Engineering and Validation

Our requirements gathering and validation were mostly done in an ad-hoc manner, as we had a direct line of communication with our customer with Slack. This allowed us to receive fast feedback and clarify any questions we had as we did not have to wait until the end of the sprint to add a new requirement. Furthermore, we were mostly provided with goals to achieve by our customer and were given a great deal of freedom in how we approached each solution.

We also incorporated feedback from a previous project into our backlog.

A protocol from such a meeting would look like Figure 36.



## Notes






- Rename to Settings -> ✓
- Change Seller/Buyer Button to TAB based system, since price etc are for both types of users
  - Es gibt 2 versch. Anmeldungen, erst wenn diese Durchlaufen sind kann verkauft werden
  - Auf Settingspage/oder Registrierungspage müssen die Daten eingegeben werden -> Rollen Buttons auf Settings Page bleiben vorhanden um den Registrierungsprozess einzuleiten
- #task Define Register Processes and then Implement them with data
- PV Management with add Button at the bottom right -> ✓
  - ✓ #task Talk to Remo for the data definitions ✓ 2022-12-13
    -  **Christian Dollfus** 9:48 AM  
we only need the max Peak Information of the panels (edited)  
 1 
    -   2 replies Last reply 12 days ago
  - open modal with explanation
- Remove text from strategies and put them at a better place -> ✓
- ask why energy surplus and energy required exist on profile/settings page -> ✓
  - Not needed, can go on Dashboard
- ✓ #task Todo: Create Wireframes for those items on Dashboard ✓ 2023-01-18

Figure 36 Example of a Protocol with Requirements

## 5 Conclusion and Future Work

### 5.1 What has been achieved?

The following to general goals will be discussed: “Publicly Available and Interactive Website” and “More Comprehensive User Experience”.

#### 5.1.1 Publicly Available and Interactive Website

We believe that we were able to fully achieve the first goal of making the publicly available and usable with our implementation of the deployment pipeline and replacing static data with data provided by the fake data API.

Since we set up the pipeline and the server, the website has since been accessible without any big issues. Since we never got a message, that the website is unreachable from our customer via Slack or Email, we are satisfied with our solution.

We also are now loading the data for the graphs from our external service, and we prepared the other sections, so that they can be connected as well.

However, we would have preferred it if we could have used HSLU’s API as this would have saved us time and we could have achieved more functionality.

#### 5.1.2 More Comprehensive User Experience

With the addition of features like Multi-Language, Tooltips and Modals we were able to provide the customer with more concise information. By letting users interact with the webapp in whatever language they prefer we achieve a more welcoming user interface. And by allowing them to get more information by hovering an icon or clicking a “Learn More” button, we increase the user experience should increase considerably, according to [4] (Point 3, “Provide In-Depth Information”) by Adobe, a leader in the design space.

We would have liked to validate our usability changes with more user testing but unfortunately run out of time.

The refactoring of the code and additional documentation we wrote should also help future development to be more efficient and effective.

### 5.2 Future Work

The application is not finished and still has improvement potential. In this chapter we discuss what could be further implemented.

#### 5.2.1 Icons

Issue: Apache License 2.0 and incorrect use of icons.

Throughout the CEEX web app icons are used. These Icons aid usability as they can communicate meaning either on their own or in combination with a label. Icons also benefit of being understood globally without having to be translated or changed for certain languages or regions, everybody knows that a cog wheel icon means will lead to a settings page.

The material theme by Google which was used for the CEEEX web app offers many icons that can be used under the Apache License 2.0. This however means that due to the following clause: "You must give any other recipients of the Work or Derivative Works a copy of this License" the license must be publicly accessible somewhere on the website.

The Icons were also incorrectly added to the web app in the prototype development. The SVG files were downloaded and added to the Angular project under the assets folder and then imported into the application as if they were a custom icon. This however is not necessary as the Angular material theme supports these icons out of the box<sup>22</sup>.

```
<mat-icon aria-label="Example home icon">sentiment_very_dissatisfied</mat-  
icon>
```

Figure 37 Correct way of using material icons font.

This could be changed in the future as using the material icons the correct way keeps the website lighter as there are no potentially unused assets being loaded.

Additionally, to adapt the colors of the icons they were manually changed using image processing software. This led to their being duplicate icons just with different colors and the icons not being customizable using CSS styles. By using the icons, the intended way this could be fixed. However, there are some custom icons for the trading strategies, ideally these should also be changed so that they can be recolored using CSS styles.

## 5.2.2 Addresses

Issue: Unverified textual addresses without geolocation.

In the current implementation a user's address is just entered as text and stored in the same way. This will in the future not be enough as it is important that the addresses are real as a lot of features of CEEEX rely on this. To ensure the entered address is real the Google Map API<sup>23</sup> could be used. This would also enable CEEEX to get the geolocation (longitude and latitude) of the address, which is in the end the essential part, essential because the distance to other users is used to determine the price of energy amongst other things.

A user should also be able to change his address if he moves house. However, this could be abused to get better energy prices therefore it may be smart to in the future verify addresses with a letter and limit the number of times a user can change his address in a year.

---

<sup>22</sup> Material Icons Font: <https://www.developer.com/languages/javascript/using-material-font-icons-in-your-Angular-11-projects/> (20.01.2023)

<sup>23</sup> Google Map API: <https://developers.google.com/maps/documentation/places/web-service> (20.01.2023)

### 5.2.3 Map

Issue: Insufficient library

The trading page features a map showing the surrounding consumers/producers. The goal of this map is to give the sense of being part of a community whilst also giving the user information on potential consumers/producers and their prices. We were able to draw custom markers on the map to show other users locations. However, after trying two different libraires from Google and multiple different approaches we were not able to draw lines on the map to connect users and show the energy flow as was initially designed. If this feature is truly relevant for CEEEX then we suggest that in the future a different map library is used rather than Google Maps. This may allow for more customization. Possible libraries could be Leaflet<sup>24</sup> or OpenLayers<sup>25</sup>.

### 5.2.4 Internationalization

Issue: Italian is missing.

Italian is one of the key languages in Switzerland and we feel like it would be advantageous to add the translation as well.

### 5.2.5 HSLU Data

Issue: Using fake data API instead of HSLU data.

As we have already mentioned the HSLU API has not been implemented yet. Therefore, once it has been implemented our fake data API should be replaced with the business logics' API.

### 5.2.6 Registration

Issue: Unverified information is being used.

Currently, when a user registers an account with CEEEX, there is a predetermined list of distribution service providers. However, this list is not in sync with all the providers in Switzerland. The customer ID that is entered is also not verified whether it belongs to the selected provider and belongs to a person with the same name and address. Additionally, CEEEX could gain more information from the user during the registration process such as their usage and cost to double check this information with the estimates provided by the provider.

---

<sup>24</sup> Leaflet: <https://leafletjs.com/> (20.01.2023)

<sup>25</sup> OpenLayers: <https://openlayers.org/> (20.01.2023)

## 6 Reflection

### 6.1 Teamwork

#### 6.1.1 Internally

The biggest hurdle internally was the difference of the time zones, since we had different work hours and different exam/school schedules. We solved this by dividing the work in a way that we were mostly independent from each other.

There was also a developer from FHNW involved, which in the start was still included in the development but since we had no clearly defined requirements while refactoring the code base, it was difficult to delegate tasks.

#### 6.1.2 Externally

Through this project we experienced what it means to work with external teams and trying to communicate with them. We also had some quite heavy dependencies on some parties, as we will discuss later.

#### Marketing

At the start, we learned that there was a marketing team from HSLU also working on CEEEX and we decided to try to initiate an exchange, since they had a more direct connection to potential customers of CEEEX. We sent them a document, in which we asked if they could evaluate our solutions with the users, but we got a negative response, since making user tests was not feasible for them. Afterwards there was no further interaction.

#### Customer

The communication with the customer was done through Slack and some meetings, that were scheduled mostly spontaneous when something was hard to explain in writing or if something larger had to be discussed. This was very different than what we hoped to at the project, as we thought of continuing working with an agile approach as we did in Ip4. Instead of a meeting at the end of a sprint, we had way shorter feedback loop which in turn also meant, that we had to answer more quickly. In the end, we think that the customer was well informed, and it was also more realistic to how development work was done in a company.

#### Algorithm Team

The interaction with the algorithm team, meaning Remo Kälin, was the most critical and important part of our project. Since we will be consuming data from his server, we had to be sure to have all the details correctly defined. This meant that we created an API document, which clearly defined on how we expect to call the Backend and what kind of response we expect. This meant that we had to work with a black box scenario, since we had no info on how exactly the data was stored on his side. Also, we had a critical dependency, since we needed his server to be available to consume the data that we had to display. Unfortunately, this dependency resulted in us having no data until the end of the project and we had to create a service ourselves, that creates some displayable data for us.

### 6.2 Learnings

Before this project neither of us had worked with Angular or Django. However, we had both already used Python and React with TypeScript, so we were aware of most of the ideas and concepts. We did

find that Angular was less enjoyable to work with compared to the more popular and documented framework, React, Svelte or Vue, as the libraries are held up to date and there is more documentation on how things have or could be implemented.

We had also never worked on a project where team members were in different locations and even time zones. At the beginning it was hard to find a system and schedule to meet up, but over time we became more efficient and used to the process.

## 7 Bibliography

- [1] Bruna Alves, "statista.com," statista.com, 12 1 2023. [Online]. Available: <https://www.statista.com/statistics/1271511/switzerland-monthly-wholesale-electricity-price/>. [Accessed 20 1 2023].
- [2] BFE, «Bundesrat bekräftigt vollständige Öffnung des Strommarktes,» BFE, 27 09 2019. [Online]. Available: <https://www.admin.ch/gov/de/start/dokumentation/medienmitteilungen.msg-id-76564.html>. [Zugriff am 19 1 2023].
- [3] R. Lüthy, "Figshare," 19 1 2023. [Online]. Available: [https://figshare.com/articles/book/API\\_Structure\\_Final\\_pdf/21923286](https://figshare.com/articles/book/API_Structure_Final_pdf/21923286). [Accessed 19 1 2023].
- [4] Adobe, "Adobe," Adobe Experience Cloud Blog, 21 6 2018. [Online]. Available: <https://business.adobe.com/blog/basics/8-ways-to-make-your-website-more-user-friendly>. [Accessed 20 01 2023].

## Declaration of Authenticity

### Declaration of Authenticity

We the undersigned declare that all material presented in this project report is our own work and written independently only using the indicated sources. The passages taken verbatim or in content from the listed sources are marked as a quotation or paraphrased. We declare that all statements and information contained herein are true, correct and accurate to the best of our knowledge and belief. This report or part of it have not been published to date. It has thus not been made available to other interested parties or examination boards.

Windisch, 20.01.2023

**Name:** George Gruffydd Rowlands

**Signature:** 

**Name:** Raphael Lüthy

**Signature:** 



# Appendix

# A Current State of UI

Login:

## Hi, Welcome Back!

Enter the World of Clean Energy Exchange

[Forgot password?](#)

Not registered yet? [Create an account](#)



Register:

## Get Started Now!

It's Free to Join - Enter the New World of Clean Energy Exchange

I agree to the [Terms of Service](#) and [Privacy Policy](#).

Already have an account? [Sign In](#)



## Trading:

**Trading** | Producer | Consumer

Map | Satellite

**Purchase Price:** 0.25 CHF

**Start Trading**

**Trading Strategy:** Never Lose

**Suppliers**

- John Doe**: 10 km, 100 kWh, CHF 0,10 / kWh
- Peter Lustig**: 19 km, 110 kWh, CHF 0,16 / kWh
- Max Mustermann**: 80 km, 122 kWh, CHF 0,20 / kWh
- Hans Muster**: 18 km, 120 kWh, CHF 0,18 / kWh
- Hans Muster**: 18 km, 120 kWh, CHF 0,18 / kWh
- Hans Muster**: 18 km, 120 kWh, CHF 0,18 / kWh

## Dashboard:

**Dashboard** | Producer | Consumer | 7 Days

**Energy Required:** 300 kWh

**Energy Bought:** 0 kWh

**Purchase Price per kWh:** 0 CHF

**Price Development:** Personal Price, Market Price

**Energy Consumption:** CEEX Network, Public Network

**George Rowlands**

**Recent Notifications:**

- You have successfully changed your price! Today
- You have successfully changed your trading strategy to m... 1 month ago
- You have successfully added your energy source! 4 month ago
- You have successfully added your profile picture! 4 month ago

**Recent Messages:**

- Jonas Wenger**: Hi Alexander, I like your PV-System. Would you like to tell me a little bit more about it? I'm interested in purchasing a new PV-System. It can do a lot of things like this and this.
- Susanne Zbinden**: Hi Alexander. Lorem ipsum dolor sit amet, consetetur sadipscing elitr.

# Profile:

CEEX TRADING DASHBOARD English

## Profile

Producer **Consumer**

**Distribution System Operator**  
IBB Energie AG

**Address**  
Bahnhofstrasse 6, 5210 Windisch

**Security**  
Change Password

**Pricing**  
Purchase Price: CHF 0 0.4 CHF 3

**Energy Sources**

**Consumer Trading Strategy**

**Market Benefit**  
This is the basic strategy applied by default to a new client. The Consumer won't pay more for the energy purchased. The Producer won't earn less for the energy injected on the grid than today.  
[Learn more >](#)

**Never Lose**  
The price setting for the trade is optimized by a trading advisor (trading robot) that tries to get value of CEEX market volatility and to optimize the earnings (producer) or the costs (consumer). This strategy requires to have flexibility loads (as battery, manageable heat pump...) and...  
[Learn more >](#)

**Flexibility**  
The User is willing to get value of the flexibility of its installation. CEEX platform make the link between the DSO and the clients, allows the DSO to use the platform to transmit signals to activate flexibility services (injection and/or demand shift / activation)  
[Learn more >](#)

**George Rowlands**

### Recent Notifications

- You have successfully changed your price!  
Today
- You have successfully changed your trading strategy to m...  
1 month ago
- You have successfully added your energy source!  
4 month ago
- You have successfully added your profile picture!  
4 month ago

### Recent Messages

**Jonas Wenger**  
Hi Alexander, I like your PV-System. Would you like to tell me a little bit more about it? I'm interested in purchasing a new PV-System. It can do a lot of things like this and this.

**Susanne Zbinden**  
Hi Alexander. Lorem ipsum dolor sit amet. consetetur sadiosino elit.

## B Initial State of UI

Login:

### Hi, Welcome Back!

Enter the world of Clean Energy Exchange

[Forgot password?](#)

Not registered yet? [Create an account](#)



Register:

### Get Started Now!

It's free to join - enter the new world of Clean Energy Exchange

email is required

Hiermit erkläre ich mich mit den [Allgemeinen Geschäftsbedingungen](#) einverstanden.

Already have an account? [Sign In](#)



# Trading:

The trading interface features a map of the Aare region with a red location pin. Below the map is a 'Place your Order' section with the following details:

- current selected price: **0,25 CHF**
- your energy consumption: **kWh / Month**
- current trading strategy: **Never Lose**
- energy received by CEEX Community: **100%**
- A green button labeled 'deactivate trade' is visible.

On the right side, there are two buttons: 'I'm a Consumer' and 'I'm a Producer'. Below them, a list of suppliers is shown, all named 'Max Mustermann' with a 200kWh capacity and a price of 0.18 CHF/kWh. Each entry includes a '300m' distance indicator.

# Dashboard:

The dashboard provides a comprehensive overview of energy trading performance. Key metrics include:

- Total energy sold:** 1000kWh
- Selling price pro kWh:** 0.25 CHF
- Total energy bought:** 0 kWh
- Purchase price pro kWh:** 0 CHF

The 'Price Development' chart shows 'Personal price', 'Market price', and 'Floor Price' in CHF/kWh over the last 3 months. The 'Energy sale' donut chart shows the distribution of energy sold: 'to the community' (green) and 'To the network' (grey).

Recent Notifications and Messages are listed on the right side, including updates on price changes, trading strategies, and profile pictures.

# Profile:



## Your Profile

<b>Your Role</b> Consumer / Producer	<b>Distribution system operator</b> IBB Energie AG	<b>Amount of Energy required</b> 300 kWh	<b>Energy surplus</b> 500 kWh
---	---	---	----------------------------------

**Price**

0 CHF 0.1 CHF 2+

PV system

## Trading Strategy

<b>Market Benefit</b>  <small>This is the basic strategy applied by default to a new client. The consumer won't pay more for the energy purchased as compared to the market.</small>	<b>Never lose</b>  <small>The price setting for the trade is supported by a trading advisor that aims to ensure that the client gets the best value of CEEX market conditions and to improve the earnings (profit) at the end of the transaction.</small>	<b>Flexibility</b>  <small>The user is willing to get value of the flexibility of its installation. CEEX platform makes the link between the DSO and the client, allows the DSO to use the platform to connect with its own flexibility services (storage and/or demand-side management).</small>
--	---	---



Test



## Recent Notifications

Monday, 04 July 2022

- You have successfully changed your price! =)  
Today
- You have successfully changed your trading strategy!  
1 month ago
- You have successfully added your energy source! =)  
4 month ago
- You have successfully added your profile picture =)  
4 month ago

## Recent Messages

Monday, 04 July 2022

Jonas Wenger