


Department of Informatics of the Technical University of Munich in cooperation with fortiss

Project Report in Advanced Practical Course Blockchain Technology for Public Sector Innovation (IN2106, IN4212)

Best Energy

Distributed Ledger based Decentralized Energy Market Design and Management Structures

Can Bayraktaroğlu* Can Bayraktaroğlu† İrem Gökçek‡
Haris Durrani§ Lukas Bernwald¶

Submission Date: July 16, 2021

Supervisors

Prof. Dr. Helmut Krcmar (TUM - Chair for Information Systems)

Advisors

Dian Balta (fortiss)
Anastasios Kalogeropoulos (fortiss)
Matthias Buchinger (fortiss)

*Computer Science, M.Sc., can.bayraktaroglu@tum.de
†Robotics, Cognition, Intelligence, M.Sc., canbay96@gmail.com
‡Data Engineering & Analytics, M.Sc., irem.goekcek@tum.de
§Computer Science, M.Sc., haris.durrani@tum.de
¶Computer Science, M.Sc., lukas.bernwald@tum.de

Abstract

The Best Energy project is a Distributed Ledger Technology (DLT) based peer to peer energy market platform which can be used for trading and distribution of electricity. The framework provides a secure and efficient marketplace for the combination of supply and demand in electricity trading, offers transparency and reduces the use of intermediaries. The platform simulates a market by having time intervals during which the producers and consumers can post their listings. At the end of this interval the market is cleared by a matching authority in the network which takes into account the listing preferences and creates listing pairs, known henceforth as matches, between the producers and consumers listings. In case of an energy sum mismatch within 1 market clock interval, the surplus or deficient energy is either sold to or bought from a retailer entity, with infinite capacity, which ensures 100% matching. The market is then reset to start a new market interval. Even though, to construct this energy market, the DLT provides us many advantages, the challenges such as scalability, interoperability and privacy are still present.

Contents

List of Figures

1. Introduction

As the recent trends show, there has been a sharp growth in small-scale distributed energy resources as the traditional residential and commercial consumer have taken an increasing important role in the energy transition i.e. generating Renewable Energy (RE). For instance the global rooftop solar Photovoltaic (PV) panels is expected to grow by 11 % over the next five years [PW17]. This uptick in RE is also necessary to reduce greenhouse emissions for power production. These small-scale resources can be utilized to manage the energy demand more efficiently as well as to enable the mix of clean energy into the grid. Based on this, we developed an electricity market trading platform for the distribution and the trading of different types of electricity employing a DLT. We provide a secure and efficient electricity trading marketplace by enabling transparent presentation of information and the derivation of new high-frequency products. By peer-to-peer trading, we are aiming that participants in the market have the possibility to reduce their electricity costs by up to 34 % to 60 %, according to the studies [Tus+18]. The Best Energy Project is developed to satisfy the growing need of energy sharing.

Best Energy uses smart meter identification for identifying unique users and unique energy blocks. We need each user to register into the market once with their smart meter identification number. Another important feature is verification. We need to be able to provide guarantees of origin for traded and stored electricity, verify customer and energy source preferences. Additionally, transparency of billing will support the evidence of price calculation and the traded electricity prices and amounts.

There are also several challenges that we hope to solve. One challenge is the integration of distributed energy systems which adds additional burden to the power grid. Currently, the burden of the power grid is out of the scope of the project at this state. However, various solutions based on the studies is analyzed during the project. Another challenge is the optimal integration of prosumers (see definition in section 2) into the energy system. In our design of the Best Energy, one market participant can either be a consumer or a producer at this state of development, which is currently left as an open point for the future work. In this report, our solutions to these challenges will be explained in detail.

A DLT based solution is developed to evaluate how it can be used to enable new forms of de-central, local energy trading. DLT is needed because of its decentralized structure where we use the immutable registry of transactions distributed across a peer-to-peer network. The smart contracts helps verify transactions and flows manage the energy trading with a defined transaction logic. It also increases the security and efficiency, provides transparency and reduces the use of intermediaries. We will use DLT for its advantages but also hope to overcome the challenges that arise from it. The solution we have provided has been developed in collaboration with OLI Systems and Reiner Lemoine Institute (RLI). The use cases, requirements, the architecture has been discussed in depth within the following sections.

2. Terms and Definitions

In the following we are going to define certain terms that are relevant to understand our system.

2.1. Stakeholders

First we introduce our stakeholders in the system.

Consumer A consumer is a market participant who will buy energy. They need energy and want to buy it a the cheapest price possible. A consumer might have specific preferences regarding the type of energy production. In this project possible consumers are businesses, companies, and organizations.

Producer A producer is a market participant who will sell energy. They produce a surplus of energy that they wants to sell. In this project possible producers are businesses, companies, and organizations.

Prosumer Prosumer stands for the ability to acts as both Producer and Consumer. Every participating node in the network can be classified as Prosumer.

Energy Supplier An energy supplier owns facilities to produce energy. They offset missing amounts of energy in the network. They want to make as much money as possible through selling energy.

Network Operator A network operator is an entity that is responsible for managing the network infrastructure. In our case entry to Corda Best Energy network can only be achieved through a permit, for example an electricity meter, from the network operator. A network operator can also be an energy supplier.

2.2. Additional Terms

After introducing our stakeholders we define some additional terms.

Listing A listing is an indication by a prosumer to the market whether they want to sell or buy energy and with which conditions. There are two different listing types.

- **Consumer Listing** A consumer listing is a listing where the consumer provides the details of its needed energy amount, type and the unit price that they are willing to pay for it.
- **Producer Listing** A producer listing is a listing where the producer provides the details of its surplus energy amount, type and the unit price that they are willing to sell it for.

Market The market is the platform for the consumers and the producers to be matched with each other in order to sell or buy different energy types. Every matching node can be seen as a separate market

Market Clock The market clock is the time interval in which the market participants enter their listings, get matched and where the energy transfer takes place. Every matching node has its own Market Clock

Market Time The market time describes the stages of the market. It is composed of three stages: initiation stage where the participants enter their listings, clearing stage where the matching takes place and reset stage where the energy transfer takes place. One market clock has to successfully go through with these three market time stages. Every matching node has its own Market Time

Matching System The matching system encompasses all techniques and participants related to the matching of producer and consumer listings.

Matcher A matcher is a network participant that matches consumer and producer listings according to a predefined rule based on the conditions that are provided in the listings while also determining the market price. Each "matcher" node on the network can be seen as an individual marketplace, with a different set of matching and pricing criteria. Any network participant can request a permit for the creation of a new matching node at any time. Matching nodes which are not directly created by Network Manager will need to be self-hosted by individuals who have applied for the permit.

3. Related Work and Technologies

Obviously most systems are not built from the ground up with no external input and technologies. Therefore, in this section we want to highlight some of this external input.

3.1. Related Work

Regarding the goal to build a distributed ledger-based decentralized energy market, we agreed on creating a Peer-to-Peer (P2P) trading platform.

P2P trading platforms provide information about energy trading among buyers and sellers, as they match their offers using the agreed or accepted regulations [Okw+20]. Through their free market aspect, although regulated, they incentivize prosumers of a region to manage their local funds efficiently within the local economy and provide them the opportunity to maximize their profits from electricity trading.

This allows the local prosumers to have control over their distributed energy resources, helping them to realise their domestic economic potential through increased intraconnectivity. Through the double-sidedness of this platform, the buyers and sellers have the

opportunity to trade their commodities through the market [Okw+20], accounting for a more efficient and cheaper distribution of energy resources in the region.

In order to account for the decentralization of the energy distribution process and transparency in terms of energy exchange, we decided to build our market on a platform based on DLT, specifically a consortium Blockchain (BC) technology. As Okwuibe et al. [Okw+20] describe it, BC technology is a decentralized networking infrastructure that allows the execution and inspection of transactions in a decentralized and transparent manner.

Following the inherent semi-private characteristics of a trading market, all processes and stages must occur between market's participants transparently: requiring them to be preselected. Based on the mentioned aspect, we decided to use a consortium BC, which is a permissioned semi-private BC technology where multiple preselected organizations/participants govern the platform.

3.2. Technologies

Now that we provided some of the basic terms and the theoretical background, we briefly go over the implementational dependencies of our application. Certainly every software project uses some technologies that it is built upon. In this section, we introduce five technologies that underlay the final project.

Corda The first technology used in this project is Corda. Corda is an open source distributed ledger platform. It is not a public blockchain, in which every participant can see every transaction. It rather relies on direct point to point communication between network participants on a need-to-know basis. In Corda, States represent on-ledger facts, Transactions are proposals to update the ledger, Contracts are for checking whether a transaction is valid for all of its input and output states that acceptable according to the requirements and Flows automate the process of agreeing ledger updates [Cor].

Kotlin The second technology used in this project is Kotlin. The Corda platform can be programmed with any Java Virtual Machine language, in particular, Java and Kotlin. In our project, we opted for Kotlin, because Java is already a quite old programming language and Kotlin can be seen as an improved version of Java with additional features, faster compile time and less verbose [Kot]. An indication for its adaptation is that it is the preferred language for Android development according to Google, the developer of Android [Lar19].

Gradle The third technology used in this project is Gradle. Gradle is a build automation tool [Gra]. Similar to Maven[1] Gradle can be used to manage the dependencies of a software component.

[1] See `https://maven.apache.org/`

Spring The fourth technology used in this project is Spring. Spring is a popular Java framework that is often used for web development [Spr]. It bootstraps the process of developing a backend for an application by dependency injection. In our application we use the Spring Boot project to create HTTP endpoints for accessing the Corda platform via RPCs, such as starting flows.

NuxtJS The fifth technology used in this project is NuxtJS. NuxtJS is an open source framework for frontend web development [Nux]. It is similar to other frontend web frameworks in a way that component-based websites can be created and information can be injected via JavaScript.

4. Development Approach

Regarding the development approach, we went for a rather traditional technique. First we discussed in several meetings with our project partners what the scope of our project is. Based on that we analyzed our stakeholders and the requirements of this project. After the initial design of our system architecture we started working on the system while updating us in the team regularly via meetings.

After designing our system we had to opt for certain technologies to use in our project. Because of the reasons outlined in section 3, we decided to use a consortium BC. We opted for the Corda system due to the reasons specified in the following.

Permissioned Corda is a permissioned distributed ledger platform. For our use case that means that the consumers and the producers need to obtain the permission of the market owner in order to participate which is in alignment with our goals.

Privacy The communication in Corda is point-to-point on a need-to-know basis, which accounts for better privacy measures and is in alignment with our peer to peer market approach which also uses peer to peer communication. In Corda there is no global central ledger in the network. Every node respectively participant has its private subjective view of the ledger.

Horizontal Scaling Linear horizontal scaling enables the transactions to execute in parallel if needed be.

Easier Programming Corda eases the programming since it uses the already adopted technologies such as JVM and SQL which simplifies the programming.

5. Scenario and Requirements

After we explained some background about the technologies and how we approached the development process, we want to present our design of the system.

Abbreviation	Name	Description
PRO	Producer	• Produces a surplus of energy for sale • In this project possibly businesses, companies, and organizations
CON	Consumer	• Requires energy • Wants to have energy for the cheapest price possible • Has specific preferences for produced energy • In this project possibly businesses, companies, and organizations
PROS	Prosumer	• Can be Producer or Consumer
NO	Network Operator	• Provides the network infrastructure • Wants the networkload distributed evenly • Can be an Energy Supplier
ES	Energy Supplier	• Balances Energy deficit/surplus in the market • Possesses facilities for Energy Production • Possesses facilities for Energy Consumption • Wishes to maximize gains from Energy Trade

Table 1: Stakeholder descriptions

5.1. Scenarios

We have five scenarios for the stakeholders in our systems: consumers, producers, prosumers, energy providers and network operators in one Market Clock. Those stakeholders are again shown in table 1. Following this idea we now present their use cases.

5.1.1. Scenario for Consumers

1. Registration: The consumer has to register to the marketplace, that is at minimum to provide the system their smart meter information.

2. Create a consumer listing to buy energy: The consumer specifies the amount of energy they need, the type of energy (renewable or non-renewable), the unit price that they are willing to pay for it and the matching node that they want to participate in with the given listing.

3. Enter into the Market accepting the market regulations and wait for matching.

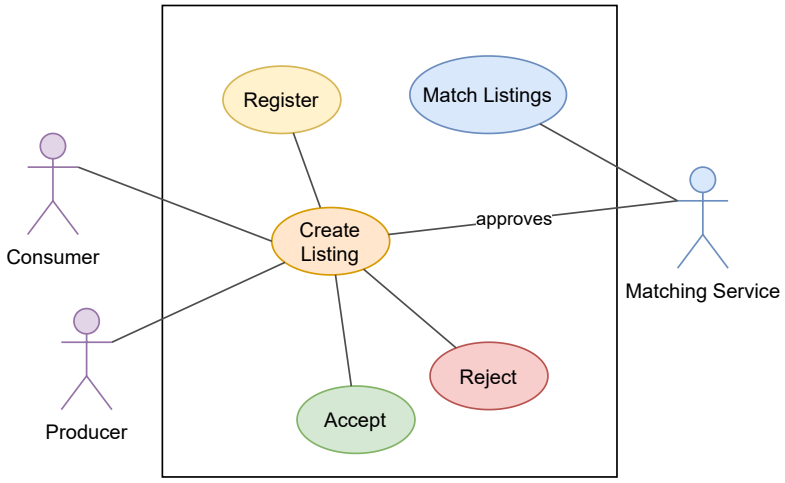


Figure 1: Use Case Diagram

4. After the market clearing signal, the consumer gets matched to a producer. Consumer receives the matching transaction from the matching node. If the consumer gets matched to a retailer/energy supplier, they have to pay a penalty as well.

5.1.2. Scenario for Producers

1. Registration: The producer has to register to the marketplace, that is at minimum to provide the system their smart meter information.
2. Create a producer listing to sell energy: The producer specifies the amount of energy they have, the type of energy (renewable or non-renewable), the unit price that they are willing to sell for it and the matching node that they want to participate in with the given listing.
3. Enter into the Market accepting the market regulations and wait for matching.
4. After the market clearing signal, the producer gets matched to a consumer. The producer receives the matching transaction from the matching node. If the producer gets matched to a retailer/energy supplier, they have to pay a penalty as well.

Based on these explanations, the use case diagram can be seen in fig. 1.

5.1.3. Scenario for Prosumers

As every market participant can be classified as prosumer depending on their listing types, the scenario for prosumers is not different than the ones for producers and consumers. To be both producer and consumer, it is enough to create at least one producer listing and one consumer listing in one market clock.

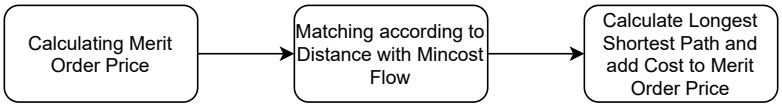


Figure 2: Phases of the Matching Process

5.1.4. Scenario for Energy Suppliers

The energy supplier acts as a figurative form of a central energy grid in our system. In order to have the market cleared up in each market loop, there should be no energy deficit or surplus in the network after the matchings are due.

In case there exists a deficit of energy after the clearing stage of the market, the consumer, whose request for energy cannot be provided by other producers in the network; is matched with the energy supplier and thus the required amount of energy is provided to the consumer via the central grid. Analogously, for a producer with energy surplus in the network; the energy supplier is matched with the producer and buys/consumes its energy over the central energy grid, leading to a cleared market.

Since the energy supplier offsets energy in the network, if there is a surplus or a deficit, and in our system the matcher is responsible for that, the use case for the energy supplier is to run a matching node in our system. The energy supplier would then in a part of the complete network be responsible for the following.

- Matching of the producers and consumers
- Regulating the market time

5.1.5. Scenario for Network Operator

The network operator is responsible for the maintenance of the physical network. In our project this would mainly entail the administration of the network participants. In our case the network operator is for example responsible for handing out the identities of the network participants, i.e. the smart meters, and admitting people to the network. Because a network operator can also be an energy supplier in our system it also has the same use cases as the energy supplier.

5.2. Requirements

For simplicity, we have to assume that every market participant already has a smarty meter installed and running correctly. We also cannot approximate the energy loss that is caused during the energy transfer and the number of grid infrastructures between participants. Thus, we assume that the smart meter identification numbers are correct and the distances between the participants is not affecting our calculations. Moreover, for the matching, we planned to use the Merit Order Pricing, see fig. 2.

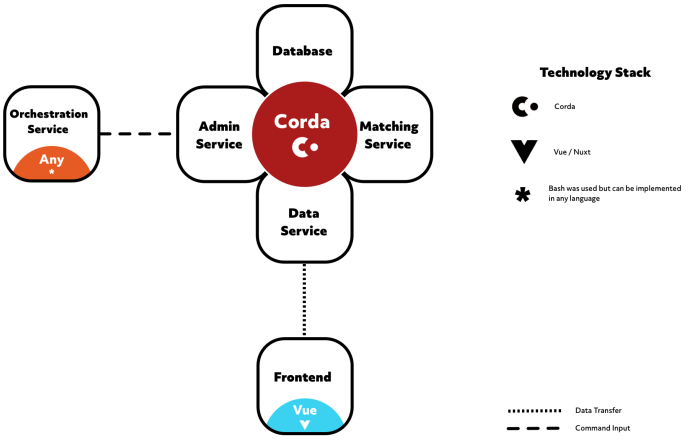


Figure 3: System Architecture

6. Architecture

For the architecture of the project, the contracts and the flows are developed on Corda, the frontend on Vue.js and the orchestration service as a script. For a better explanation, a visualization is shown in fig. 3.

6.1. Listing

The listing state has the variables listing type, electricity type, unit price, sender, matcher and market clock. The listing type is to define if the listing is for a consumer or for a producer. The electricity type is for the producers' energy type, 0 for renewable, 1 for otherwise and -1 for consumers since the listing states are defined same for both participants. Sender is for storing the ID/Address of the node creating the listing and matcher ID/Adress of the node which will be responsible for matching the market participants. Market clock is the current market clock that comes from the Market time state.

The corresponding contract performs the sanity check on the data that has been provided by the user to ensure that all inputs are valid. After the listing object has been cleared by the contract, it is saved into the vaults of both the chosen matching node and the node that initiated the listing creation.

Figure 4: Market Time phases in one Market Clock

6.2. Market Time

The market time state has four variables: market clock, market time, sender and receiver. The market clock is for keeping track of the which market time cycle the market is in. The market time is for limiting the capabilities of the participants, monitor the time intervals in the market. For this, we have came up with an enumeration system as follows:

Market time is 0 Market is initiated. The consumers and producers can enter into the market by creating listings.

Market time is 1 Market clearing takes place. The consumers and producers will be matched through the matching system and the energy transfer takes place.

Market time is set from 2 Market is reset and market clock is incremented by one creating a new market cycle.

The corresponding contract makes sure that market time value must be greater than or equal to 0, market time value must be lower than 3 and market clock value must be non-negative integer.

There are five flows corresponding to the market time state, visualised in fig. 4:

Initiate Market time Initiate Market time flow checks that the Market time was 0 and sets it to 1.

Clear Market time Clear Market time flow checks that the Market time was 1 and sets it to 2.

Reset Market time Reset Market time flow checks that the Market time was 2 and sets it to 0 and increments the Market clock by one.

Broadcast Transaction Broadcast Transaction for broadcasting the current Market time state to all the other nodes in the network.

Record Transaction Observer Record Transaction as Observer for receiving the more recent Market time state from other nodes of the network and record it in the vault.

6.3. Matching System

The matching system has the responsibility to create matches between the posted consumer and producer listings in the current market clock. Our initial idea was more complex than the approach we ultimate went for because of time and resources constraints.

6.3.1. Initial Idea

During the initial design phase our main focus for designing the system lay on constructing a meaningful matching algorithm for the system. In this approach we also still had a graph of the network in mind which would contain the distances between the point to point connection of the participants in the energy grid. An example of such a graph is shown in fig. 5. The idea consisted of three aspects.

1. We would calculate the merit order price. Here we sort the producer listings in ascending order according to their price. The producer listings are included in the current market time from lowest highest price until the total energy demand form the consumer listings on the market is reached. The producer listing with the highest price that is still in included in the market time sets the market price which is also known as the merit order price. The idea here is that everybody on the market pays the same price.

2. We would match the producers and consumers according to their distance to each other in the network. There should be no excess or deficiency of energy in the market, i.e. 100 % of the energy should be matched. Since the energy market is not an accurate representation of the real world, we make some theoretical assumptions. We use a special consumer respectively producer called the retailer, that can provide infinite amounts of energy and can consume infinite amounts of energy. This market participant is also added to the network graph and thus 100 % matching is possible.

   The initial idea for calculating the the distances was to use a min-cost flow algorithm. In the min-cost problem nodes in a graph have demands for units, that also can be negative. Edges have cost per flow of unit sent across that edge. The goal is to minimize the overall cost of the flow while meeting the node demands. In our case the cost of the flow across an edge would be the distance of the nodes and the demand of a node would be the actual energy demand of a node, which as mentioned might be positive or negative. This min-cost flow problem can for example be solve via the simplex algorithm for linear programs.

   However, solving the min-cost problem on the initial graph would only give information about energy transportation in the network, but not how the nodes are actually matched to each other. Therefore, another idea we had was to first calculate the all pairs shortest path in the network. This can be done using the Dijkstra or Floyd Warshall algorithm. Then we could create a dense bipartite graph of the producers

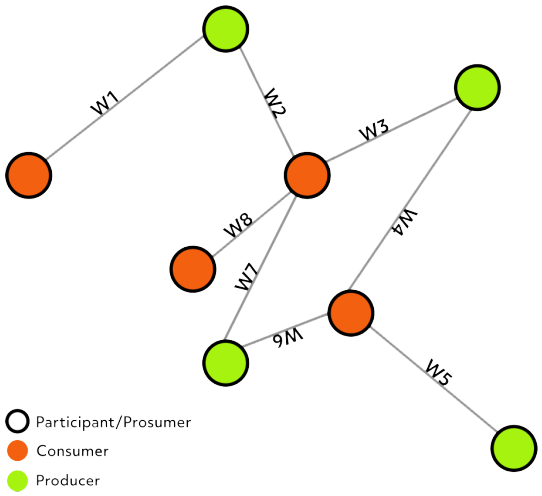


Figure 5: Network graph of the energy grid

and consumers using these calculated shortest paths and finally on this modified graph we could perform the min-cost flow algorithm. The flow in this graph then tells us how much energy is matched from a producer to a consumer.

3. Additional we would calculate the longest shortest path in the original graph. We would add this cost of this path to the merit order price. Since the longest shortest path gets shorter, the more local structures there are, this would overall nudge the market participants to build local structures to reduce the market price. Since one of the additional goals of this system was to enhance local structure this would be in alignment with our objectives.

6.3.2. Actual Implementation

Because of time and resource restrictions, we could not go for our initial idea for matching. Instead we applied the following more simple scheme. First of all the matching is initiated following the execution of clear market time flow.

The matching state has five variables: unit price, unit amount, buyer, seller and matcher. The unit price is the price per unit that will be billed to the buyer and paid to the seller. The unit amount is the integer amount of electricity units sold/bought. The buyer and seller are ID/Address of nodes that created the consumer and producer listings respectively. The matcher is the identity of the matching node which creates this new matching state.

The corresponding contract ensures that the matching transaction has two inputs, the buyer from the consumer listing and the seller from the producer listing, and one output.

The unit amounts and the matcher identity is also verified.

There are three flows corresponding to the matching state:

Matching Flow The Matching Flow, shown in fig. 6a, retrieves all the unconsumed listings from the vault for the current clock time and segregates them according to the electricity types. The merit order price is then calculated for each type. After this, the producers are sorted in ascending and the consumers in descending order so that the lowest producer is matched with the highest consumer in terms of unit price. In case of a mismatch of the unit amounts between listings, the listing with the greater amount is split utilizing the *Split Flow*, following which a new match with equal unit amounts is created and added to a matchings hash set.

All listings that are not matched, in case of an energy sum mismatch between the producer and consumer listings, are matched with the retailer, with the initial assumption that the retailer has an infinite amount of production as well as consumption capacity. However, for this type of matching a penalty amount is added to the merit order price such as to incentivize the producers to offer a lower unit price and the consumers to bid a higher unit price.

After all listings have been matched, either between the market participants or the retailer, they're iteratively passed to the *Single Matching Flow*.

Splitting Flow Splitting flow splits a listing into two listings with a given required unit amount as a parameter.

Single Matching Flow Single matching flow, depicted in fig. 6b, receives a producer and a consumer listing as well as the unit price and amount as inputs. It creates a new transaction with inputs from these listing states and outputs a new matching state. The concerned nodes are notified and their vaults updated with this matching transaction.

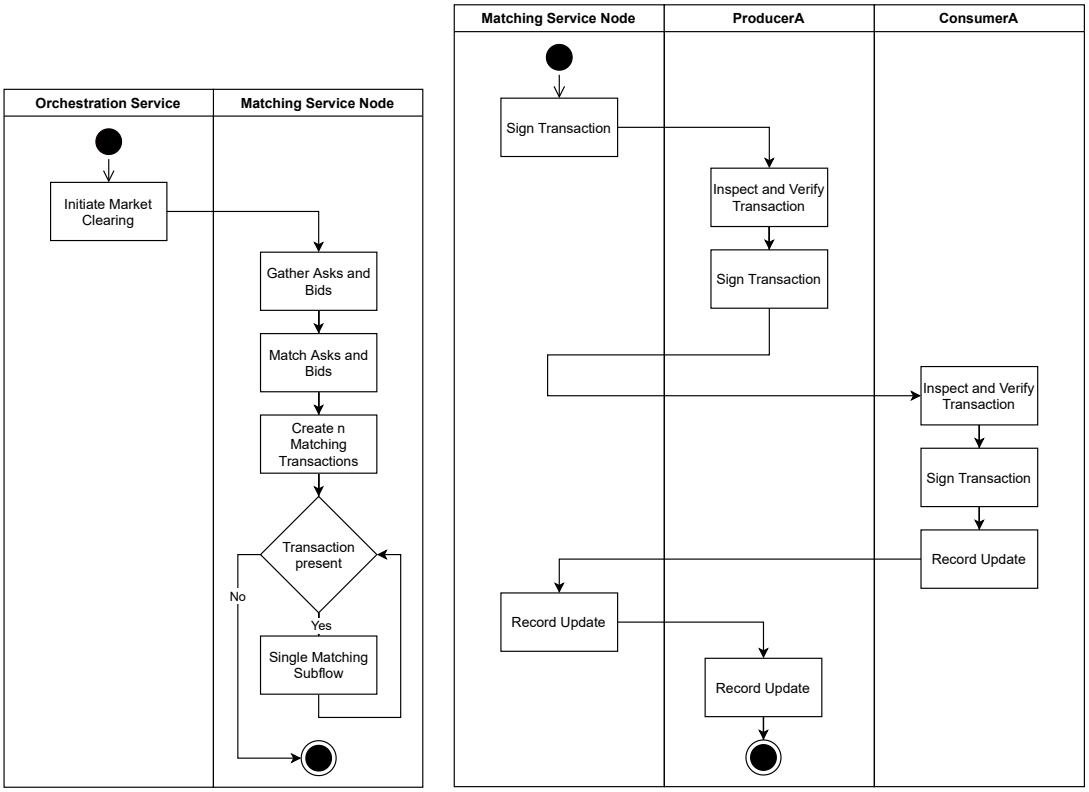
6.4. Orchestration Service

In the current setup the orchestration services shown in fig. 3 is a simple Bash[2] respectively Batch script, that can invoke the market time flows and the matching at the matching node.

6.5. Frontend

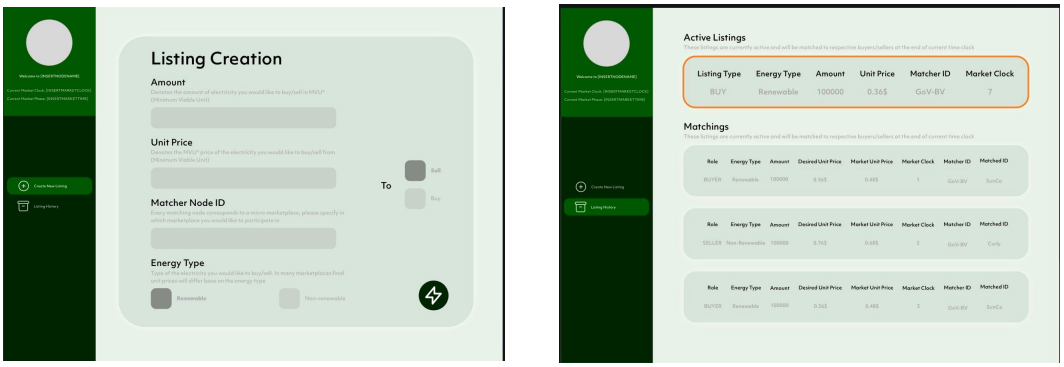
The user interface of this project consists of two pages: Create Listing and History.

[2]See `https://www.gnu.org/software/bash/`



(a) General Matching Flow (b) Single Matching Flow

Figure 6: Matching Flows



(a) Create Listing page (b) History Page with Active Listing & Matchings

Figure 7: Graphical User Interface Mockups

- Create Listing page is a form filled by consumers/producers, referred as market participant. The market participant enters the amount of energy and its price and submits it. fig. 7a is the example visualisation of this page.
- History page is for viewing the current listing and previously matched listings. The market participant can see the current listing waiting to be matched and pending energy transfer, as well as the old listings that have been matched in the previous market clocks with the information of the concerned participants, energy amount and price.

The navigation bar on the left side of the page is for navigating between the above mentioned two pages. It also has the information regarding the current market time and market clock. The navigation bar can be viewed in both fig. 7a and fig. 7b.

7. Prototype and Evaluation

In this Prototype and Evaluation we want to introduce our prototype and evaluate the development.

7.1. Prototype

We have run the project with mock data, created listings between different parties and we have successfully matched them. The User Interface is also functional, we can create listings and view the previous listings and matchings. For more please see the demonstration video we have prepared: Best Energy Prototype

7.2. Evaluation

In this section we are evaluating the capabilities of our prototype and discuss some up and down sights.

7.2.1. Features

There are three main features built into the system to achieve the mentioned properties of a decentralized market place.

Identification We can identify unique users and energy blocks to prevent possible cunning behaviour of participants and to ensure the integrity and privacy of the participants of our network. The unique identification of every user as well as the energy blocks is ensured by the smart meter, which is issued by a trusted authority. Each listing posted by a market participant has to be signed with their key which means a user can be identified by the trusted authority.

Billing Transparency Each user has access to their list of matching's where they can view previous listings that were matched. Every match has details to the matched node, the merit order price per unit, market time as well as energy type among other key information. This suffices for evidence of traded energy. Furthermore, if the matching node allows, a participant can access the unit price's of all listings for a certain market time. Using this, the merit order price can be calculated for that time slot. This serves as evidence for the merit order price.

Verification Through a well-established verification process with guarantees of origin, we plan to avoid overlooking on individual preferences of participants, in terms of energy source. The energy preference can be verified in the matchings list for every user.

7.2.2. Tests

In Corda, you can test all parts of the application within code. It possible to write tests for the contracts, flows, and even integration tests that setup a network and then can run flows in that network using RPCs. There are three main techniques for testing a CorDapp.

1. We can start a so called `MockNetwork` and start flows using the nodes in that network. This can be done in code, but there is no RPC interface provided.

2. We can use the so called `DriverDSL` to start nodes in the network. This can be also done in code and here there is also a RPC interface provided.

3. We can manually start a network to run tests. It also can somewhat be done in code, not in the Corda code however. Instead we can use the so-called Cordform plugin to define nodes in the Gradle build script. This requires a lot of manual input, but here also a RPC interface is provided.

This process of defining a network with coda is vaguely related to the Infrastructure as Code (IaC) principle [Art+17], in which the deployment of a computer system is specified via program code. In the case of our application this helps to specify tests in an as close to production environment as possible. This is very beneficial for the automation of tests, because on every change made to the code that change can be potentially automatically tested in an environment of how it is actually used.

7.2.3. Lessons Learned

During the process of this project there were some lessons we learned. They are not necessarily about the content, but also about the general development process. It seems appropriate to mention these in a report of a practical course.

Lacking Documentation The Corda documentation, in particular the Application Programming Interface (API) documentation, is not as in depth as it seemed to be according to our initial research. There are some modules such as the `net.corda.testing.core` module that can be found in the most up to date version of the source code on GitHub[3], but are not even listed in the official API docs and only briefly explained in the tutorials for Corda. Since in general the API docs also mostly only have very short descriptions and explanations compared to the official Java docs and the fact that there is no search functionality in the API docs, finding out how certain functions worked was rather challenging. Furthermore, the error messages of Corda are not easily searchable in the internet in case something goes wrong. We did not find many tutorials on the internet and also Stackoverflow[4] did not contain that much useful information from our perspective. Thus, it was rather difficult to develop for the Corda platform.

Corda also has multiple ways to test the code that were mentioned in the previous subsection and according to our research it is not clear which one is the recommend or best way. From our experience Corda might be suited for large scale enterprise production environments where developers have time to get acquainted with the system, which is also indicated by availability of an enterprise version of the platform, but it might not be the most optimal for rapid prototyping of ideas in the format of the practical course, since it requires a lot of different parts to be configured and worked on until it reaches a production ready state.

[3] See `https://github.com/corda/corda`
[4] See `https://stackoverflow.com/`

Outdated and Deprecated Versions Corda uses older versions of some the technologies it depends on. They still use Kotlin version 1.2 which was released in 2017. According to our research[5] they cannot switch to newer versions of Kotlin because of various issues. For this build system Gradle version 5.6.4 which was released in 2019 instead of the newest Gradle 7. For testing their code and for code in the tutorials Corda still uses JUnit 4 instead of JUnit 5 although JUnit 5 is even backwards compatible with JUnit 4. They upgraded some plugins to use JUnit 5[6], but that is not reflected in the tutorials and templates. Albeit using old technology is not inherently bad, the possibility of dropped support in the future and the missing out of potentially helpful features is still there.

Need for powerful hardware Development with Corda is not easy, if you do not have the required hardware for it. Using an old and not so powerful laptop for development leads to simple tests needing one to two minutes to run and leaving the the laptop unusable. This is due to the fact, that for the tests Corda spins up a complete network. This is great for testing your application in an environment as close to production as possible, but hurts productivity when rapidly developing a prototype without the required resources. According to our knowledge there does not seem to be a way to use some online tools to alleviate the local computer from this computational load.

8. Conclusion

Concluding this report we want to give a short summary of the achievements and look into the future for further development opportunities.

8.1. Summary

The prototype we have developed for Best Energy creates a market for electricity trading with the most essential functionalities satisfying the needs of the consumers and producers. At this state, our design can be used for local markets, as it would be easier to track the users. Due to the limited time and resources, we had to reduced the scope of the project and thus there are still some ideas and functionalities on the table that all can be integrated with our implementation.

8.2. Future Work

As we have only provided the essential functionalities, there are still more that can be added into the system. In the following paragraphs our additional ideas and how we thought of adding them are explained.

[5] See `https://r3-cev.atlassian.net/browse/CORDA-3115`
[6] See `https://r3-cev.atlassian.net/browse/CORDA-2917`

Register and Login One of those features is a "Register and Login" functionality. We did not focus on implementing one, because managing logins in general is not very easy. For the time being, the uniqueness of a user is guaranteed by the cryptographic identity of a deployed node. This node in our case would be a smart meter in the home of an owner. Nodes in Corda do actually already have some user authentication capabilities. Thus, in the future either these capabilities should be utilized or another additional login system for potential remote access of the nodes should be embedded into the system.

Better Matching Capabilities Another area that might need further refinement is the matching process. The current matching capabilities are quite simple since it only calculates the merit order price. For the future the matching should be enhanced to allow for more intelligent matching of consumers and producers. For example the calculations can be defined by another authority in the market allowing the change of the price calculation system by an external authorized party.

More Flexibility for Market Participants The next point is rather a technical detail than an actual needed improvement, but the current prototype lets market participant to be only either producer or consumer during one market time slot. Market participant can switch between roles between market time slots. This makes theoretical sense, since in one time slot a participant either produces more energy than he needs or consumes more energy than he produces. So he either wants to sell or buy energy in a time slot. For future versions the system should be adapted so it is possible to be both, a producer and consumer, in a market time. The advantage of this feature is still open to discussion, but this would allow for more flexibility and maybe will reveal potential unseen use cases.

Invoice Generation Another useful functionality can be creating invoices. Currently the market is more of theoretical nature and not necessarily directly implementable in practice, among others because there is no real money being transferred. Thus, to help the integration of the market into the real world in the future the generation of invoices should be made possible. The idea is that the market participants can generate the invoices of the energy they sold or bought with a button after energy transactions have taken place. It can be useful for tax records, bookkeeping, legal protection, tracking inventory and even for marketing purposes.

Deployment using Containers Furthermore, currently the nodes in the network are deployed using the JVM on a local machine. This requires dependencies to be installed on the machine where the node is to be deployed and thus requires additional configuration. Because it is possible to run Corda inside of Docker containers [Meh20], for the future, the deployment should be performed using containers, for example by utilizing Docker. This makes the deployment overall easier since all the dependencies for the application

are included in container image and don't have to be manually installed. Deploying the CordApps using containers can also lead to more isolation.

Production Ready Configuration Another point concerning the deployment is that currently some configuration, for example how to deploy the nodes or which notary to choose for notarization of a transaction, are only configured for development use and should not be used like this in practice. The nodes are currently started with the included Gradle script. This results in the nodes running in development mode which is not safe for a production environment. For future production developments a strategy for deploying the CorDapp to a real network should be established. This goal also ties in with the usage of containers and the employment of DevOps strategies. Furthermore the notary, the node that notarises the transactions, is chose arbitrarily. Since there is only notary in the development setup, it does not matter for development. However, according to comments in the sample projects[7] this is not the recommended approach and only applicable for test and single notary environments. Instead a notary should be chosen by its name that is either coded into the flow or even better supplied by a configuration. Therefore for the future, a proper deployment strategy for real life production use has to be developed.

DevOps Also among others related to the deployment there is the general development approach that should be taken in such a project. When scaling up this project from a prototype to a production ready project it will be advantageous to utilize a DevOps approach. This would entail the automated execution of tests in the cloud and the automated deployment of the system to the network using Continuous Integration (CI)/Continuous Deployment (CD) pipelines. Wrapping the application into containers as mentioned would support this process.

[7] See `https://github.com/corda/samples-kotlin`

References

[Art+17] M. Artac et al. "DevOps: Introducing Infrastructure-as-Code". In: *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. 2017, pp. 497–498. DOI: 10.1109/ICSE-C.2017.162.

[Cor] *Documentation and training for Corda developers and operators*. URL: https://docs.corda.net/.

[Gra] *Gradle User Manual*. URL: https://docs.gradle.org/current/userguide/userguide.pdf.

[Kot] *Kotlin Docs*. URL: https://kotlinlang.org/docs/home.html.

[Lar19] F. Lardinois. *Kotlin is now Google's preferred language for Android app development*. 2019. URL: https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/.

[Meh20] A. Meher. *Containerizing Corda with Corda Docker Image and Docker Compose: Corda Blog*. 2020. URL: https://www.corda.net/blog/containerizing-corda-with-corda-docker-image-and-docker-compose/.

[Nux] *Nuxt.js - The Intuitive Vue Framework*. URL: https://nuxtjs.org/.

[Okw+20] G. C. Okwuibe et al. "A Blockchain-based Double-sided Auction Peer-to-peer Electricity Market Framework". In: *2020 IEEE Electric Power and Energy Conference (EPEC)*. 2020, pp. 1–8. DOI: 10.1109/EPEC48502.2020.9320030.

[PW17] M. Peck and D. Wagman. "Energy trading for fun and profit buy your neighbor's rooftop solar power or sell your own-it'll all be on a blockchain". In: *IEEE Spectrum* 54 (Oct. 2017), pp. 56–61. DOI: 10.1109/MSPEC.2017.8048842.

[Spr] *Spring*. URL: https://spring.io/.

[Tus+18] W. Tushar et al. "Transforming Energy Networks via Peer-to-Peer Energy Trading: The Potential of Game-Theoretic Approaches". In: *IEEE Signal Processing Magazine* 35.4 (2018), pp. 90–111. DOI: 10.1109/MSP.2018.2818327.

Glossary

Notation	Description
API	Application Programming Interface.
BC	Blockchain.
CD	Continuous Deployment.
CI	Continuous Integration.
DLT	Distributed Ledger Technology.
IaC	Infrastructure as Code.
P2P	Peer-to-Peer.
PV	Photovoltaic.
RE	Renewable Energy.
RLI	Reiner Lemoine Institute.