INSTITUTO POLITÉCNICO
DE VIANA DO CASTELO

ESTG

Platform for peer-to-peer energy trading with a smart contract framework

Plataforma para comércio de energia peer-to-peer usando uma infraestrutura baseada em smart contract

Kévin Carvas Domingues

Platform for peer-to-peer energy trading with a smart contract framework
Plataforma para comércio de energia peer-to-peer usando uma infraestrutura baseada em smart contract

2024

Escola Superior de Tecnologia e Gestão

INSTITUTO POLITÉCNICO
DE VIANA DO CASTELO

INSTITUTO POLITÉCNICO DE VIANA DO CASTELO

Platform for peer-to-peer energy trading with a smart contract framework

Plataforma para comércio de energia peer-to-peer usando uma infraestrutura baseada em smart contract

Kévin Carvas Domingues

MSC PROJECT WORK

Mestrado em Engenharia informática

Escola Superior de Tecnologia e Gestão

Orientador: Professor Doutor António Miguel Cruz

Orientador: Professora Doutora Maria Estrela Cruz

February 27, 2025

Resumo

Nos últimos anos, o setor energético passou por uma transformação impulsionada por avanços tecnológicos e uma crescente preocupação com a sustentabilidade. Essa mudança despertou um interesse crescente no comércio de energia ponto-a-ponto (peer-to-peer (P2P)) como estratégia promissora. Os Certificados de Energia Renovável (Renewable Energy Certificates, REC, ou Garantias de Origem, GO) desempenham um papel crucial na garantia da rastreabilidade de fontes de energia sustentáveis, fornecendo registos auditados para produtores de energia com baixo impacto ambiental. A tecnologia *blockchain* surge como uma ferramenta fundamental para os mercados de certificados de Energia Renovável, oferecendo uma abordagem descentralizada que reduz a dependência de intermediários, corta custos e aumenta a transparência. O registo distribuído e imutável da *blockchain* garante transações transparentes e acessíveis a todos os participantes autorizados, abordando preocupações de confiança e eficiência nos processos convencionais de certificados de Energia Renovável. Este relatório de projeto de mestrado propõe uma solução para automatizar o processo de criação e negociação de certificados de energia renovável, baseada em *blockchain* e contratos inteligentes, que se liga a sistemas de medição de energia e a autoridades de certificação independentes. A emissão e negociação de certificados de Energia Renovável é registada de forma permanente e imutável na *blockchain*.

Abstract

In recent years, the energy sector has been transformed due to technological advances and a growing focus on sustainability. This shift has led to increased interest in peer-to-peer (P2P) energy trading as a promising strategy. Renewable Energy Certificates (REC) or Guarantees of Origin (GO) play a vital role in ensuring the traceability of sustainable energy sources, providing audited records for low-impact producers. Blockchain technology has emerged as a key enabler for REC/GO markets, offering a decentralized approach that reduces reliance on intermediaries, cuts costs, and enhances transparency. The blockchain's distributed and immutable record ensures transparent transactions accessible to all authorized participants, addressing trust and efficiency concerns in conventional REC/GO processes. This master's project report proposes a solution to automate the process of creating and trading REC/GO certificates, based on blockchain and smart contracts, that connect to energy measurement systems and independent certification authorities. The issuance and trading of REC/GO certificates are recorded permanently and immutably on the blockchain.

Agradecimentos

É com imensa alegria e um profundo sentimento de realização que concluo esta tese, encerrando assim uma etapa marcante da minha trajetória académica e pessoal. Ao longo deste caminho, que foi repleto de desafios, crescimento e aprendizagem, contei com o apoio indispensável de muitas pessoas, às quais quero dedicar este espaço de agradecimento.

Em primeiro lugar, expresso a minha mais sincera gratidão à minha família, que sempre foi o meu porto seguro. Aos meus pais, pelo amor incondicional, incentivo constante e apoio em cada decisão que tomei. À minha irmã, por acreditar em mim e comemorar comigo cada pequena vitória.

Agradeço aos meus amigos, que estiveram ao meu lado nesta jornada, trazendo leveza e força nos momentos mais difíceis.

Deixo também um agradecimento especial aos meus orientadores, António Miguel Cruz e Maria Estrela Cruz, pela paciência, orientação e sabedoria. As suas contribuições e sugestões foram fundamentais para o aprimoramento deste trabalho. Este processo não teria sido o mesmo sem a sua dedicação.

Por fim, mas com igual importância, agradeço à minha namorada, pela paciência, compreensão e apoio irrestrito ao longo de todas as fases deste percurso. O seu carinho e incentivo foram essenciais para que eu pudesse alcançar este objetivo.

A todos que, de alguma forma, fizeram parte desta jornada, o meu mais sincero obrigado. Esta conquista é um reflexo do impacto positivo que cada um teve na minha vida.

Kévin Carvas Domingues

"You should be glad that bridge fell down.
I was planning to build thirteen more to that same design"

Isambard Kingdom Brunel

Contents

List of Figures

List of Abbreviations and Acronyms

API	Application Programming Interface
JWT	Json Web Token
REST	Representational State Transfer
HTTP	Hypertext Transfer Protocol
APP	Application
Go	Golang
GO	Guarantee or Origin
REC	Renewable Energy Certificate
DSR	Design Science Research
P2P	Peer-to-peer
RPS	Renewable Portfolio Standards
ECDSA	Elliptic Curve Digital Signature Algorithm
POW	Proof of Work
FBA	Federated Byzantine Agreement
DOM	Document Object Model
VM	Virtual Machine
CI	Continuous integration
CD	Continuous deployment
FIT	Feed-in Tariffs
FIP	Feed-in Premiums

Chapter 1

Introduction

The transition to renewable energy sources has fundamentally reshaped the global energy landscape, creating opportunities and challenges in ensuring efficient energy management and distribution. As the adoption of decentralized energy production methods, such as solar panels and wind turbines, continues to grow, traditional systems struggle to adapt to the dynamic needs of modern energy markets. This thesis explores the development of a blockchain-based energy trading platform aimed at addressing these challenges by enabling secure, transparent, and efficient transactions of energy certificates.

Blockchain technology has emerged as a promising solution for decentralization and trust, providing immutable and transparent records of transactions without relying on a central authority. By leveraging blockchain in the energy sector, this thesis aims to create a system where energy certificates—representing units of renewable energy production—can be traded efficiently while ensuring trust and integrity. The proposed solution integrates smart contracts to automate complex workflows, including energy certificate ownership transfers, price calculations, and consumption tracking.

The motivation for this project stems from the need to streamline energy trading processes and reduce inefficiencies in traditional systems. Current methods often involve intermediaries, manual verification, and lack transparency, which can lead to delays and increased costs. By integrating blockchain with a user-friendly API, the system ensures rapid and reliable operations, empowering users to trade energy certificates with confidence.

This thesis also addresses critical challenges encountered during the development process, such as the performance limitations of blockchain-based REST APIs and the need for efficient data management. A mock server was employed to simulate regulatory authority calls, ensuring robust testing of the platform under real-world-like conditions.

By addressing the inefficiencies of traditional energy trading systems, this project demonstrates the transformative potential of blockchain in fostering a decentralized, transparent, and efficient energy marketplace.

1.1 Contextualization and motivation

In recent years, the energy sector has undergone a significant transformation driven by technological advancements and an increasing awareness of the importance of sustainability. The shift to renewable energy sources and the decentralization of energy generation have created a demand for more efficient and adaptable commercialization models.

Energy is an essential resource for economic and social development, but its production, primarily based on fossil fuels, has significant environmental impacts. Transitioning to more sustainable energy sources is a global priority. In this context, peer-to-peer (P2P) energy trading emerges as a promising strategy.

The need for traceability in the sustainable origin of consumed energy has led to the creation of Renewable Energy Certificates (RECs), allowing the registration of energy producers operating with low environmental impact [15]. These certificates, known as RECs or Guarantees of Origin (GOs) depending on the country, are non-physical assets awarded after an audit, ensuring commitment to sustainable practices. Each REC or GO represents 1 MWh (one megawatt-hour) of energy generated from a renewable source and injected into the power grid [6]. Here, we will refer to these certificates collectively as RECs.

Faced with the urgency to reduce greenhouse gas emissions from electricity production, companies are seeking more accessible strategies to achieve their sustainable energy balance. RECs emerged as market instruments in the late 1990s [2] to promote renewable energy production, providing an additional revenue stream for renewable energy generators. For each megawatt-hour of renewable energy generated, the producer receives a REC. These can be traded in the market. Conversely, when a consumer purchases a REC, they acquire the right to consume the energy that was injected into the system, and the corresponding REC is retired from the market, ensuring no double counting. Each REC details the type of energy source (wind, solar, hydro, or biomass) and the month and year when the energy was generated [6].

Conventional processes for registering and trading RECs often rely on intermediaries and centralized systems, leading to potential trust and efficiency gaps. Blockchain technology aims to mitigate these limitations by introducing a decentralized approach that eliminates reliance on intermediaries, reduces operational costs, and promotes transparency throughout the value chain. Blockchain provides a distributed and immutable ledger where each REC transaction is recorded transparently and is accessible to all authorized participants. By decentralizing the management of these certificates, blockchain offers benefits such as reduced fraud risk, elimination of redundancies, and increased efficiency.

1.2 Objectives

The primary goal of this thesis is to develop and validate a blockchain-based energy trading platform that facilitates the secure and efficient transfer of energy certificates. The system aims to address key challenges in the current energy market by utilizing blockchain technology to create

a decentralized, transparent, and automated solution for energy trading. The specific goals of the project are outlined below:

- **Design and Implement a Blockchain-Based Energy Trading System**
  One of the main objectives of this thesis is to design and implement a robust energy trading platform powered by blockchain. This involves creating a system where energy certificates—representing renewable energy units—can be securely and transparently transferred between users. The blockchain serves as the underlying infrastructure, ensuring that each transaction is immutable, traceable, and verifiable.

- **Integrate Smart Contracts for Automating Energy Transactions**
  Another key objective is to integrate smart contracts into the system to automate energy certificate transactions. By leveraging smart contracts, the platform can automatically enforce transaction rules such as ownership transfer, price calculation, and consumption tracking, eliminating the need for intermediaries and reducing the risk of human error. This ensures the efficiency and integrity of each transaction while reducing operational overhead.

- **Develop an API for Easy Interaction with the Blockchain Platform**
  To enhance usability, this project aims to develop an API that allows external applications and users to interact with the blockchain-based platform. The API will provide endpoints for transferring energy certificates, calculating transfer prices, and retrieving consumption data, among others. This API is designed to be user-friendly and accessible, enabling developers and organizations to integrate the energy trading platform into their own systems.

- **Test and Validate the System's Performance and Reliability**
  A critical goal is to rigorously test and validate the energy trading platform to ensure its functionality, performance, and scalability under real-world conditions. This includes conducting end-to-end tests to simulate user interactions, load tests to validate the system's ability to handle high traffic, and validation of key features such as price calculation and transaction processing. The results of these tests will help assess whether the system meets the desired performance standards and can handle real-world usage scenarios.

- **Identify and Address Performance Bottlenecks and Scalability Issues**
  Throughout the development and testing phases, another key objective is to identify any performance bottlenecks or scalability issues in the system. For example, one challenge identified is the performance of the FabLo REST API, which is considerably slower than the Go-based API, potentially acting as a bottleneck. Addressing these issues through optimization, system design improvements, or integration of additional logic into smart contracts will ensure that the platform is capable of supporting a large number of users and transactions with minimal latency.

- **Provide a Clear Path for Future Enhancements and Scalability**
  Finally, this thesis aims to identify potential areas for future work and improvements, ensuring that the platform can evolve with the growing demand for renewable energy and energy trading. This may include expanding the scope of the platform to support additional energy types, integrating with external regulatory systems, or implementing advanced features like dynamic pricing models, energy consumption forecasting, or real-time energy trading.

1.3 Document Structure

This document is structured into five main chapters, each addressing a critical aspect of the research and development of the Energy P2P Blockchain Platform.

Chapter 1: Introduction

This chapter outlines the motivation behind the research, the problem statement, and the objectives of the study. It also highlights the contributions made by this research to the domain of energy trading platforms and provides an overview of the document's structure.

Chapter 2: Background

Chapter 2 presents the foundational concepts required to understand the Energy P2P Blockchain Platform. This includes a theoretical introduction to blockchain technology, smart contracts, renewable energy certificates, and related literature in the field of decentralized energy trading systems.

Chapter 3: Peer-to-Peer Energy Trading Platform

This chapter delves into the core of the research, detailing the architecture proposed for the Energy P2P Blockchain Platform. It includes system modeling, the design process, and a comprehensive description of the development and implementation phases, including the tools and technologies utilized.

Chapter 4: Implementation

This chapter details the implementation of the proposed solution, including the development of smart contracts, backend API, and the frontend interface. It explains the deployment environment using Docker and highlights challenges encountered during implementation. Key functionalities and workflows of the system are demonstrated.

Chapter 5: Validation and Results

In this chapter, the experimental validation of the platform is presented. It describes the test scenarios used to validate the platform, the results obtained, and an analysis of its performance in facilitating peer-to-peer energy trading and ensuring transparency and efficiency.

Chapter 6: Conclusions and Future Work

The final chapter summarizes the key findings and contributions of the thesis. It reflects on the outcomes of the research and discusses potential areas for future work to enhance the platform's capabilities or explore related applications.

This structure ensures a logical progression from identifying the problem to proposing, implementing, and validating a solution, culminating in a discussion of findings and future directions.

Chapter 2

Theoretical background

2.1 Introduction

The energy sector has been the stage for a significant revolution, driven by technological advancements and a growing awareness of the urgency for sustainable practices. The transition to renewable energy sources and the decentralization of energy generation have emerged as crucial responses to global environmental challenges.

In this context of profound changes, peer-to-peer (P2P) energy trading emerges as an innovative and promising approach. The ability for producers and consumers to trade energy directly with each other, without the need for intermediaries, represents a fundamental shift in the traditional dynamics of the energy market.

Simultaneously, blockchain technology and smart contracts have gained prominence as catalysts for this transformation. The application of blockchain, through its distributed and immutable ledger, aims to overcome challenges related to reliability, security, and efficiency in energy trading processes. The Hyperledger Fabric platform, with its focus on enterprise solutions, provides an ideal environment for implementing smart contracts that automate and securely enforce agreements between parties.

The development of a platform that integrates blockchain and smart contracts to enable P2P energy trading not only simplifies exchanges between producers and consumers but also redefines their relationship, establishing a more transparent and efficient market model.

2.2 Technologies and concepts related to renewable technology markets

2.2.1 Renewable Energy Certificates

Renewable Energy Certificates (RECs) play a crucial role in promoting sustainability in the energy sector, acting as fundamental instruments for tracking and certifying the origin of energy generated from renewable sources. The importance of these certificates lies in their ability to provide

transparency and ensure that consumers and businesses can make informed choices in favor of more sustainable practices.

The operation of RECs is based on a digital certification system that represents the specific amount of electricity generated by renewable sources [4]. When a renewable energy plant produces electricity, it is issued a corresponding REC, indicating that an equivalent amount of green electricity has been fed into the grid. The acquisition of this certificate by an end consumer serves as tangible proof that the electricity consumed is sourced from renewables.

RECs contribute to sustainability by creating a market for green energy [13]. Companies and consumers who wish to adopt sustainable practices can purchase RECs to offset their electricity consumption. This stimulates demand for renewable energy, driving investments and infrastructure development in this sector. Additionally, RECs play a vital role in compliance with environmental goals and regulations. Many countries establish renewable energy standards that companies must meet. RECs provide an effective way to meet these obligations, allowing companies to purchase certificates to offset their carbon emissions and promote a lower carbon footprint.

The transparency provided by RECs is fundamental to building consumer trust. By allowing consumers to track the origin of the energy they consume, RECs empower sustainable choices and encourage environmental responsibility.

Beyond their evident importance in promoting sustainability, RECs play a significant role in stimulating the development of more efficient technologies and infrastructures for renewable energy generation. The growing demand for RECs drives investments in clean energy sources, contributing to technological advancements and reducing financial barriers associated with sustainable electricity production.

Additionally, RECs can be seen as catalysts for global commitment to climate change mitigation. By providing a tangible tool to offset carbon emissions, RECs not only serve regulatory requirements but also encourage a proactive mindset in organizations towards achieving and surpassing emission reduction targets.

Another important aspect is the role of RECs in environmental education and awareness. By involving consumers and businesses in the process of tracking and purchasing these certificates, awareness of individual carbon footprints is promoted. This increased awareness of individual contributions to carbon emissions has the potential to create a more environmentally responsible society committed to sustainable practices.

2.2.2 Guarantees of origin (GO)

Guarantees of Origin (GoOs) play a vital role in verifying that a specific portion of energy consumed by customers was produced from renewable sources. In Europe, GoOs are issued to ensure transparency in the energy mix provided by suppliers, as required by Directive 2009/72/EC. Despite their significance, the current GoO system faces challenges, such as limited transparency and the risk of double counting.

The TRINITY project [10] addresses these challenges by implementing a blockchain-based platform for the issuance and management of GoOs. This platform leverages the Federated Byzantine Agreement (FBA) algorithm, which ensures fast transaction processing with minimal energy consumption. Blockchain technology significantly enhances the transparency, traceability, and security of the GoO system, making it more robust and efficient.

The TRINITY platform is built on three integrated components: the GO Platform for issuing GoOs, the RES Manager for managing requests from renewable energy producers, and a trading platform for GoO transactions. Together, these components ensure accurate issuance, efficient transfer, and reliable cancellation of GoOs, fully utilizing the strengths of blockchain to improve the process.

The primary objective of GoOs is to certify to end customers that a specified amount of energy comes from renewable sources while preventing issues like double counting and lack of disclosure. While the platform represents a significant improvement in transparency and efficiency, further development is required to achieve full functionality.

The document emphasizes the critical role of blockchain technology in addressing the current limitations of the GoO system. By using the FBA algorithm, the TRINITY platform [10] ensures fast, energy-efficient transaction processing, which is essential for maintaining the system's sustainability. Moreover, blockchain's decentralized nature guarantees transparency and traceability in all transactions, reducing the risk of fraud and enhancing trust in the GoO system.

Additionally, the TRINITY platform's comprehensive integration of its three components allows for efficient GoO management. The GO Platform ensures GoOs are issued in compliance with regulatory standards, while the RES Manager streamlines the request process for renewable energy producers. The trading platform provides a marketplace for buying and selling GoOs, ensuring liquidity and facilitating the overall market for renewable energy certificates.

The liberalization of European energy markets was designed to promote competition and transparency, paving the way for the rise of green energy tariffs. At the heart of these tariffs are Guarantees of Origin (GoOs), which serve as critical certifications verifying that a specific amount of energy has been generated from renewable sources. These electronic certificates allow energy suppliers to offer green energy at premium prices, encouraging the broader adoption of renewable energy.

This document explores the transformative potential of blockchain technology in reshaping the GoO system. By creating a decentralized marketplace [3] for trading tokenized GoOs, blockchain can significantly enhance transparency, security, and efficiency within the market. The study simulates such a marketplace using the Ethereum blockchain and smart contracts, enabling prosumers (producer-consumers) to sell GoOs directly to end consumers. This peer-to-peer approach removes the need for intermediaries, reduces transaction costs, and increases overall market efficiency.

Two distinct market strategies were simulated: a fixed price strategy based on the average GoO price from 2014, and a variable price strategy that adjusts with the fluctuating costs of fossil fuels and renewable energy. The findings suggest that the variable price strategy proves more profitable

for prosumers, showcasing the advantages of dynamic pricing in a decentralized energy market [3].

2.2.3 Blockchain Technology and Sustainability

Blockchain technology allows for the recording of transactions involving assets that may or may not be tangible. Blockchain offers an inherent trust mechanism, reducing the risks of manipulation and associated operational costs. Each transaction is recorded in a block [7] , which is linked to other blocks in a chronological chain structure. Once a block is added to the blockchain, all participants gain access to its information. Since each block contains information from the previous block, no data can be altered without detection. These blocks record transactions as an asset is transferred to a new owner. The irreversibility of transactions in the chain provides immediate, shared, and transparent information stored in an immutable ledger. No transaction can be added to a block without validation and consensus from participants [11], ensuring the data on the blockchain is authentic and tamper-proof [14]. The underlying infrastructure of the blockchain is a peer-to-peer (P2P) network where each participant must follow transaction processing rules and consensus protocols.

Several studies address the trade of Renewable Energy Certificates (RECs) on the blockchain. In the solution mentioned in , it was proposed that Public Service Commissions issue RECs on the Ethereum blockchain. According to the author, this involves creating a crypto-REC corresponding to the renewable energy injected into the power grid, which can be sold to qualified renewable energy producers or traded in an electronic market. This process can be repeated periodically, following the requirements of Renewable Portfolio Standards (RPS).

In , a proof-of-concept system was proposed for decentralized energy trading. Using blockchain, multi-signatures, and anonymous encrypted message flows, the proposed system facilitates secure energy transactions on a decentralized smart grid without an intermediary. The authors mention the use of the Elliptic Curve Digital Signature Algorithm (ECDSA) to validate transaction authenticity. ECDSA public keys are used to generate unique addresses, making ownership tracking difficult. The proposed system supports private messages between peers, protecting the identity of those involved through unique alphanumeric strings. The system reduces dependence on a central energy supplier, enabling token-based energy trading on a P2P network.

The blockchain-based market is approached with an architectural optimization design in , including two main modules: P2P energy trading forecasting and an intelligent energy prediction model to meet the demand of a distributed network. The first module uses blockchain for P2P transactions, while the second creates a prediction model to effectively meet energy demand. Energy consumption data from Distributed Energy Resources (DERs), supported by this architecture, are analyzed using supervised learning and data mining approaches to meet future energy demand. The pre-processed data are used for real-time and short-term scheduling of DERs. This dual focus involves using blockchain for P2P trading and creating a prediction model to effectively meet energy demand. The main focus in is to develop a trading strategy for prosumers (producer-consumers), aiming to reduce energy consumption, maximize cost savings, and realize a local

P2P trading network. The importance of optimized strategies for prosumers to plan entry and exit from trades, considering the availability of energy at the node, stored or locally generated, is highlighted. The optimized strategy could be implemented in a mobile application where the client shares information and uses environmental data to calculate optimized offers in the P2P trading system.

In a P2P network, transactions occur without the need for an intermediary, providing flexibility to add or remove peers as necessary, especially during network disruptions or system losses. The network is divided into three layers: the market layer, the physical layer, and the regulatory layer . The market layer facilitates the secure sharing of information and resources among users, handling financial settlements, offers, and requests. After financial transactions are completed, the actual energy transfer occurs in the physical layer. The regulatory layer ensures the presence of sufficient market users and support policies for P2P trading.

Decentralization in the market allows direct communication between peers, facilitating decentralized trading and data communication. This structure grants peers the freedom to choose their participation in energy trading, maintaining privacy and offering exceptional scalability. However, the absence of a central party presents security challenges that could affect user participation in the market.

2.3 Tecnologies

2.3.1 Blockchain

Blockchain is a decentralized and distributed digital ledger technology that securely records transactions across multiple computers, ensuring transparency, security, and immutability. Each transaction is grouped into blocks, which are linked together in a chronological chain, forming a continuous, unalterable record. This structure makes blockchain resistant to tampering, as any change to a single block would require altering all subsequent blocks, which is computationally impractical.

Blockchain operates through a consensus mechanism, which ensures that all participants (or nodes) in the network agree on the validity of transactions. These mechanisms, such as Proof of Work (PoW) or Federated Byzantine Agreement (FBA), enable trustless interactions between parties without needing intermediaries. Initially popularized by cryptocurrencies like Bitcoin, blockchain has since expanded into various industries, offering new solutions for enhancing transparency, traceability, and security in sectors ranging from finance to energy.

By utilizing a decentralized, transparent, and tamper-proof ledger, blockchain creates an infrastructure that can revolutionize how data and transactions are managed across many fields, including the energy sector.

2.3.2 Smart Contracts

Smart contracts are self-executing contracts with the terms of the agreement directly written into code. These digital contracts automatically enforce and execute predefined conditions when spe-

cific criteria are met, without the need for intermediaries. Smart contracts run on blockchain networks, ensuring transparency, security, and immutability.

Once deployed on a blockchain, smart contracts autonomously manage transactions between parties. The decentralized nature of blockchain ensures that the contract's code and execution are visible to all participants, reducing the risk of fraud or manipulation. Common use cases include automated financial transactions, supply chain tracking, and decentralized energy trading.

By eliminating the need for trusted third parties, smart contracts streamline processes, reduce costs, and improve efficiency across various industries, offering a reliable way to automate agreements in a secure and transparent environment.

2.3.3 Hyperledger Fabric

Hyperledger Fabric is an open-source blockchain platform designed for enterprise use cases, offering a modular architecture that provides flexibility, scalability, and confidentiality. Unlike public blockchains, Hyperledger Fabric is a permissioned blockchain, meaning that participants are known and authorized, making it suitable for business applications requiring high levels of privacy and security.

One of the key features of Hyperledger Fabric is its support for smart contracts (called "chaincode" in Fabric) and private channels, which allow organizations to conduct transactions in a confidential and efficient manner. It uses a unique consensus mechanism where transaction processing and validation are separated, enabling faster and more scalable transaction throughput.

This platform is widely adopted in industries such as supply chain management, healthcare, and finance, where transparency, data integrity, and controlled access are essential. Its modular approach allows customization of key components like consensus, membership services, and data privacy, making it highly adaptable to diverse business needs.

2.3.4 Fablo

Fablo is an automation tool designed specifically for setting up Hyperledger Fabric blockchain networks quickly and efficiently. It simplifies the process of configuring, deploying, and managing Hyperledger Fabric environments by automatically generating the necessary configuration files, network structure, and Docker-based containers for the various Fabric components.

Typically, setting up a Fabric network involves configuring numerous elements such as peer nodes, orderers, certificate authorities, and channel configurations, which can be time-consuming and prone to error. Fablo addresses this complexity by allowing developers to define the network architecture in a simple configuration file (usually YAML). Once the configuration is set, Fablo automates the creation of Docker containers for each component, along with smart contracts (chaincode) deployment and testing environments.

Fablo is particularly useful for those working with small to medium-sized Hyperledger Fabric networks in development and testing phases, as it accelerates the network setup process and reduces manual configuration. It also supports multi-organization networks, making it easier to simulate real-world blockchain deployments where different organizations participate in a distributed ledger.

By integrating with Docker, Fablo ensures that the entire network is containerized and portable, which simplifies testing and scaling as the network grows. Overall, Fablo streamlines Hyperledger Fabric deployments, making it a valuable tool for developers and blockchain engineers looking to experiment with and deploy Fabric-based blockchain solutions rapidly.

2.3.5 Go

Go, often referred to as Golang, is an open-source programming language developed by Google. It was designed to be simple, efficient, and scalable, making it ideal for building large, reliable software systems. Go combines the ease of use of dynamically typed languages like Python with the performance and safety of statically typed languages like C and C++.

One of Go's key features is its concurrency model, which allows developers to efficiently handle multiple tasks simultaneously using goroutines. This makes Go particularly well-suited for developing highly scalable applications, such as web servers, cloud services, and distributed systems.

Go's simplicity in syntax, combined with its performance, has led to its widespread adoption in industries requiring fast and reliable execution, including blockchain technology, cloud computing, and microservices. Its standard library is also extensive, providing built-in tools for tasks like networking, cryptography, and file handling, which makes Go a powerful and versatile language for modern development.

2.3.6 React

React is a popular open-source JavaScript library used for building user interfaces, particularly for single-page applications. Developed and maintained by Facebook, React allows developers to create reusable UI components that efficiently update and render as data changes, making it ideal for dynamic and interactive web applications.

One of the core features of React is its component-based architecture, where the UI is broken down into smaller, reusable pieces. Each component manages its own state and can be composed together to build complex interfaces. This modularity promotes code reusability and maintainability, reducing development time.

React uses a virtual DOM (Document Object Model) to improve performance. Instead of updating the entire real DOM with every change, React updates a lightweight virtual representation of the DOM, calculates the most efficient way to apply updates, and then applies only those changes. This results in faster rendering and a smoother user experience.

Additionally, React's declarative syntax makes it easier to reason about the code and predict how the interface will behave based on changes in state or data. React is widely used in modern web development and can be integrated with other frameworks or libraries like Redux for state management and Next.js for server-side rendering, making it highly flexible for various use cases.

2.3.7 Docker

Docker is an open-source platform that automates the deployment, scaling, and management of applications using containerization. Containers allow developers to package applications along with their dependencies, libraries, and configurations into a single, lightweight unit that can run consistently across different environments, such as development, testing, and production. This ensures that "it works on my machine" is no longer a problem.

Unlike traditional virtual machines (VMs), Docker containers share the host machine's operating system kernel, making them more resource-efficient, faster to start, and portable across different operating systems. Each container runs in isolation but uses only a fraction of the resources that a VM would need, leading to better performance and scalability.

At the core of Docker is the Docker Engine, which is responsible for building and running containers. Developers write a Dockerfile, which defines the environment setup, dependencies, and the commands needed to run an application. Using this file, Docker creates an image—a snapshot of the application—that can be distributed and deployed as containers.

Docker also provides Docker Hub, a centralized repository for sharing Docker images. This allows developers to reuse base images, such as operating systems or application servers, and layer their applications on top of them, speeding up development and reducing duplication.

In modern DevOps workflows, Docker is essential for creating consistent and reproducible environments, facilitating continuous integration and continuous deployment (CI/CD) pipelines. Docker containers can run on any infrastructure, whether it's a developer's local machine, cloud servers, or Kubernetes clusters, making it highly versatile for various use cases in modern application development and deployment.

2.4 Related work

The energy sector is undergoing significant transformation driven by technological advancements and the increasing urgency to adopt sustainable practices. The transition toward renewable energy sources and the decentralization of energy generation have emerged as vital strategies to address global environmental challenges. Within this evolving landscape, peer-to-peer (P2P) energy trading has gained prominence as an innovative and promising approach. This concept allows producers and consumers to exchange energy directly, without intermediaries, fundamentally reshaping the traditional dynamics of energy markets by fostering transparency, reducing costs, and increasing efficiency.

Blockchain technology plays a pivotal role in enabling these advancements. By leveraging a distributed and immutable ledger, blockchain ensures security, reliability, and transparency in energy transactions. The adoption of this technology addresses key challenges in the energy sector, such as the need for trust, efficient record-keeping, and automation of transactions. For instance, studies like the one in [8] propose the issuance of Renewable Energy Certificates (RECs) using blockchain platforms like Ethereum, representing the renewable energy injected into the grid. These certificates can be traded in an electronic marketplace, encouraging the adoption of renewable energy by facilitating transactions with verified credentials.

Other initiatives, such as those described in [1], present decentralized systems that employ blockchain, multi-signature algorithms, and encrypted anonymous messaging to facilitate secure energy trading in a smart grid environment. These systems aim to reduce dependency on centralized energy providers by introducing token-based transactions within a P2P network. Additionally, public-key cryptography mechanisms, such as the Elliptic Curve Digital Signature Algorithm (ECDSA), are used to validate transaction authenticity while safeguarding user identities.

Innovative models, such as the one proposed in [9], go further by combining P2P energy trading with predictive energy demand algorithms. This dual approach not only facilitates efficient trading through blockchain but also integrates forecasting methods using machine learning and data mining to optimize energy supply and demand. By leveraging Distributed Energy Resources (DERs) and real-time scheduling, this system reduces energy consumption, maximizes cost savings, and supports the establishment of local P2P energy markets. The model also emphasizes the importance of optimized strategies for prosumers (producers and consumers) to make informed decisions regarding energy storage and trade.

In Europe, renewable energy trading is often regulated through complementary voluntary and mandatory markets for green certificates. For example, countries such as Norway, Sweden, and the United Kingdom have established mandatory green certificate markets with quota obligations, while the European Union promotes a voluntary market for Guarantees of Origin (GOs). These mechanisms not only ensure compliance with renewable energy targets but also encourage the use of clean energy by offering incentives through programs like Feed-in Tariffs (FIT) and Feed-in Premiums (FIP) [12].

Renewable Energy Certificates (RECs) serve as critical instruments in advancing sustainability within the energy sector. Acting as digital proof of electricity generated from renewable sources, RECs provide transparency and enable consumers and businesses to make informed, sustainable choices [4]. When renewable energy is injected into the grid, a corresponding certificate is issued, representing the green energy's environmental attributes. These certificates are crucial for promoting renewable energy adoption, as they stimulate demand, attract investment, and facilitate compliance with environmental regulations.

Blockchain technology, particularly frameworks like Hyperledger Fabric, has emerged as a key enabler of P2P energy trading and REC management. The decentralized nature of blockchain eliminates single points of failure, enhances security, and ensures data integrity. Each transaction is recorded with cryptographic protection, timestamps, and unique identifiers, making the system

tamper-proof and transparent. Platforms like Hyperledger Fabric also support the implementation of smart contracts, which automate and secure agreements between parties, ensuring efficient and reliable execution of transactions [5].

The integration of blockchain into P2P energy trading presents numerous advantages. It simplifies energy exchanges, fosters trust among participants, and supports sustainable practices by reducing reliance on fossil fuels. Furthermore, the technology empowers prosumers by allowing them to manage energy trading autonomously while maintaining privacy and scalability. However, the absence of centralized oversight poses challenges, particularly in terms of security and market participation. Future implementations must address these limitations to fully realize the potential of blockchain in revolutionizing the energy sector.

By combining blockchain technology with innovative business models, this approach has the potential to redefine energy markets, making them more sustainable, transparent, and efficient. This thesis explores these transformative possibilities, laying the groundwork for a decentralized, renewable, and user-driven energy future.

Chapter 3

Peer-to-peer energy trading with a smart contract framework

The primary goal of this research is to develop a decentralized and transparent platform for peer-to-peer (P2P) energy trading, leveraging blockchain technology. This chapter introduces the proposed solution, which aims to address the challenges of trust, transparency, and efficiency in the renewable energy market. By combining the principles of blockchain with the concept of Guarantees of Origin (GoOs), the platform enables secure and traceable energy transactions between prosumers and consumers.

The proposed system is designed to provide a robust and scalable architecture that integrates smart contracts for automation, Hyperledger Fabric for a permissioned blockchain network, and a user-friendly interface powered by React. This solution supports the issuance, trading, and validation of GoOs in a decentralized market, ensuring compliance with energy certification standards and facilitating real-time energy transactions.

To achieve this, the platform incorporates the following key components:

- **Guarantees of Origin (GoOs):** A digital representation of renewable energy production, allowing prosumers to certify the origin of their generated energy.
- **Blockchain-based Architecture:** Utilizing Hyperledger Fabric to create a transparent and immutable ledger for recording all transactions, enhancing trust and accountability.
- **Smart Contracts:** Automating processes such as GoO issuance, transfer, and validation, reducing manual intervention and transaction costs.
- **P2P Trading Mechanism:** Enabling direct energy trading between prosumers and consumers through a decentralized marketplace, eliminating intermediaries and fostering efficiency.
- **Integration with User Interfaces:** A React-based front-end application to ensure seamless interaction with the platform for all users.

This chapter provides a detailed explanation of the system's architecture, including the design considerations and the technologies employed. It also discusses the implementation process, highlighting how the components work together to create a functional and reliable solution for P2P energy trading. Through this proposed solution, the research aims to contribute to the transition towards a more sustainable and decentralized energy ecosystem.

3.1 Use Case Diagram Description

The proposed platform for peer-to-peer (P2P) energy trading is modeled through a use case diagram, which identifies the main actors and their interactions with the system. This diagram (Figure 3.1) illustrates the functionalities offered by the platform to its users and how the system supports the various activities involved in energy certificate management and trading.

3.1.1 Actors Identified

Three primary actors are identified in the use case diagram:

- **Consumer:** Represents the end user of the platform who is interested in purchasing energy certificates, monitoring their energy consumption, and analyzing related financial and market data.
- **Energy Producer:** Represents renewable energy producers who generate energy and use the platform to register, manage, and trade energy certificates. They are also responsible for verifying energy sold and tracking their financial transactions.
- **Regulatory Authority:** Represents the entity responsible for verifying the origin of energy certificates to ensure compliance with regulatory standards and maintaining the integrity of the certification process.

3.1.2 Use Cases Description

The use cases in the diagram detail the functionalities provided by the platform. They include the following:

- **Buy Energy Certificates:** The *Consumer* can purchase energy certificates from the platform, enabling them to support renewable energy sources.
- **See Consumption Data:** The *Consumer* can access detailed information about their energy consumption patterns, allowing for informed decision-making.
- **See Amount Spent with Transactions:** The *Consumer* can review financial data, specifically the total amount spent on energy-related transactions.
- **See Market Energy Price Prediction:** The *Consumer* can view predictions regarding energy market prices, aiding in strategic decisions regarding certificate purchases.

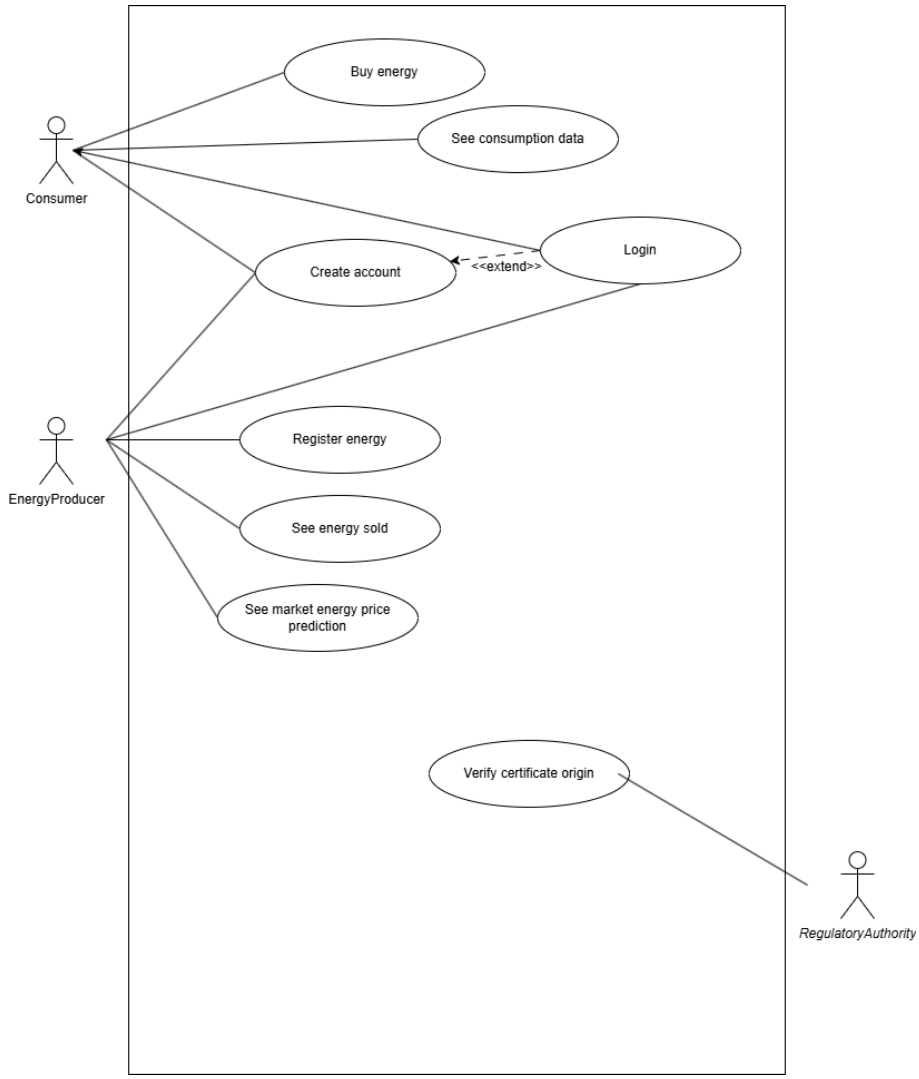

Figure 3.1: Use case model for the platform.

- **See Certificates Owned:** The *Consumer* and *Energy Producer* can track the energy certificates they currently hold.
- **Register Energy Certificates:** The *Energy Producer* can register energy certificates on the platform, certifying that the energy they produce is renewable.
- **See Energy Sold:** The *Energy Producer* can track the amount of energy they have successfully sold on the platform.
- **See Amount Earned with Transactions:** The *Energy Producer* can review financial details, specifically the revenue generated from energy certificate sales.
- **Verify Certificate Origin:** The *Regulatory Authority* ensures that the origin of energy certificates complies with regulatory standards, maintaining transparency and trust in the system.

3.1.3 System Overview

The use case diagram highlights how the platform serves as a centralized interface for consumers, producers, and regulatory authorities to interact. By integrating functionalities such as certificate registration, trading, and validation, the system fosters transparency, efficiency, and trust in the P2P energy market. Each actor's role and actions contribute to creating a dynamic and secure ecosystem for renewable energy trading.

3.2 Domain Model Description

The domain model (Figure 3.2) represents the key entities, their attributes, and the relationships that define the functionality of the proposed peer-to-peer (P2P) energy blockchain platform. It serves as the conceptual framework for the system, encapsulating the core elements, their interactions, and where the date will be stored:

- *On chain data:* Data that will be stored in the blockchain
- *Off chain data:* Represents data that will be stored in a relational database

3.2.1 Entities and Their Attributes

The domain model identifies the following primary entities and their respective attributes:

- **User:** Represents the system's primary actor. This entity encapsulates information about the user and their role within the platform.
  - *id (int):* Unique identifier for the user.
  - *email (string):* The user's email address.
  - *password (string):* Encrypted password for authentication.
  - *blockchainUser (string):* Identifier for the user's blockchain account.
  - *name (string):* The user's full name.
  - *address (string):* Physical address of the user.
  - *city (string):* The user's city.
  - *postalCode (string):* Postal code of the user's address.
  - *userType (int):* Indicates whether the user is a consumer or producer.
- **EnergyProducer (inherits from User):** Extends the *User* entity, representing energy producers.
  - *productionCapacity (double):* Maximum energy production capacity.
- **EnergyCertificate:** Represents certificates associated with renewable energy generation and trading.

  - *energyCertificateID (string):* Unique identifier for the certificate.
  - *ownerId (int):* Identifier of the current certificate owner.
  - *producerId (int):* Identifier of the producer who registered the certificate.
  - *emissionDate (Date):* Date when the certificate was issued.
  - *usableMonth (int):* Month for which the certificate is valid.
  - *usableYear (int):* Year for which the certificate is valid.
  - *energyTypeId (int):* Identifier for the type of energy the certificate represents.
  - *regulatoryAuthorityId (int):* Identifier for the authority that validated the certificate.
  - *availableToSell (boolean):* Indicates whether the certificate is available for trading.

- **Transaction:** Represents energy certificate transactions within the platform.
  - *transactionID (string):* Unique identifier for the transaction.
  - *certificateTokenRef (string):* Reference to the certificate token.
  - *fromUserID (string):* Identifier of the user initiating the transaction.
  - *toUserID (string):* Identifier of the recipient user.
  - *transactionDate (Date):* Date of the transaction.
  - *price (double):* Monetary value of the transaction.

- **Consumption:** Represents energy consumption data for users.
  - *userId (int):* Identifier for the consumer.
  - *consumptionYear (int):* Year of the recorded consumption.
  - *consumptionMonth (int):* Month of the recorded consumption.
  - *energyConsumed (double):* Total energy consumed by the user.
  - *energyTypeId (int):* Identifier for the type of consumed energy.

- **EnergyType:** Represents the classification of energy, such as solar or wind.
  - *energyTypeId (int):* Unique identifier for the energy type.
  - *energyType (string):* Description of the energy type.

- **RegulatoryAuthority:** Represents the entity responsible for verifying and validating energy certificates.
  - *name (string):* Name of the regulatory authority.
  - *id (int):* Unique identifier for the authority.
  - *address (string):* Physical address of the regulatory authority.
  - *apiURL (string):* API endpoint for certificate verification.

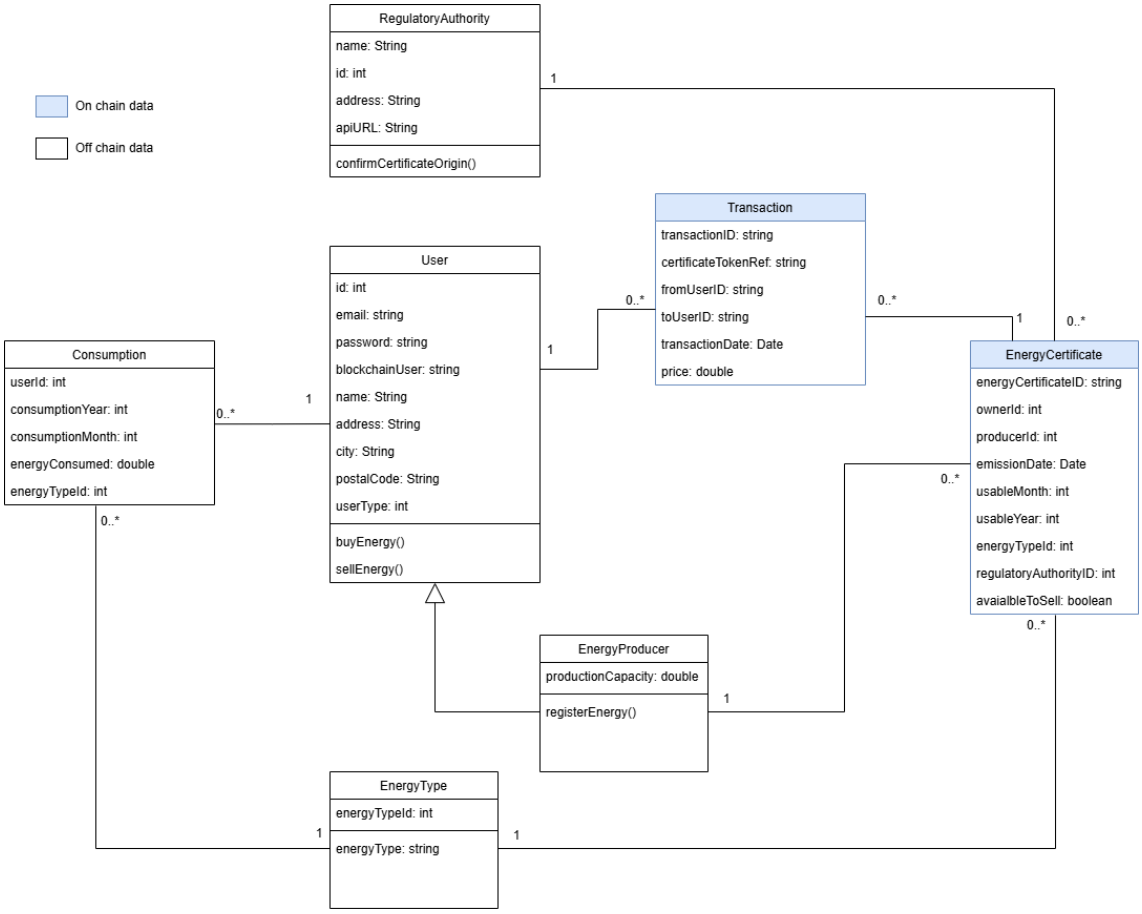

Figure 3.2: Proposed data model.

3.2.2 Relationships

The relationships between entities are as follows:

- **User and Consumption:** A *User* can have multiple *Consumption* records, capturing their energy usage over time.
- **User and EnergyCertificate:** A *User* can own multiple *EnergyCertificates*, representing their investments or production contributions.
- **EnergyProducer and EnergyCertificate:** An *EnergyProducer* registers certificates linked to their production.
- **Transaction and EnergyCertificate:** A *Transaction* references a specific *EnergyCertificate*, enabling its trading between users.
- **EnergyCertificate and EnergyType:** Each *EnergyCertificate* is associated with a specific *EnergyType*.
- **EnergyCertificate and RegulatoryAuthority:** A *RegulatoryAuthority* verifies the authenticity of *EnergyCertificates*.

3.2.3 System Overview

The domain model provides a structured representation of the P2P energy trading platform, emphasizing the flow of data and interactions between users, energy producers, certificates, transactions, and regulatory authorities. It highlights the core entities and relationships necessary to implement a secure and efficient blockchain-based solution for energy trading.

3.3 System Architecture

The architecture of the proposed (Figure 3.3) for P2P energy trading platform integrates multiple layers, ensuring secure, transparent, and efficient interactions among users, regulatory authorities, and energy producers. This architecture leverages modern technologies, including blockchain and RESTful APIs, to deliver a seamless experience for trading energy certificates. The key components of the architecture are described below.

3.3.1 Frontend Layer

The frontend layer represents the user interface through which participants interact with the platform. A web application built with React enables users to perform actions such as viewing certificates, initiating transactions, and analyzing consumption or production data. This layer acts as the main gateway for users, ensuring accessibility and user-friendliness while maintaining a robust connection to the backend.

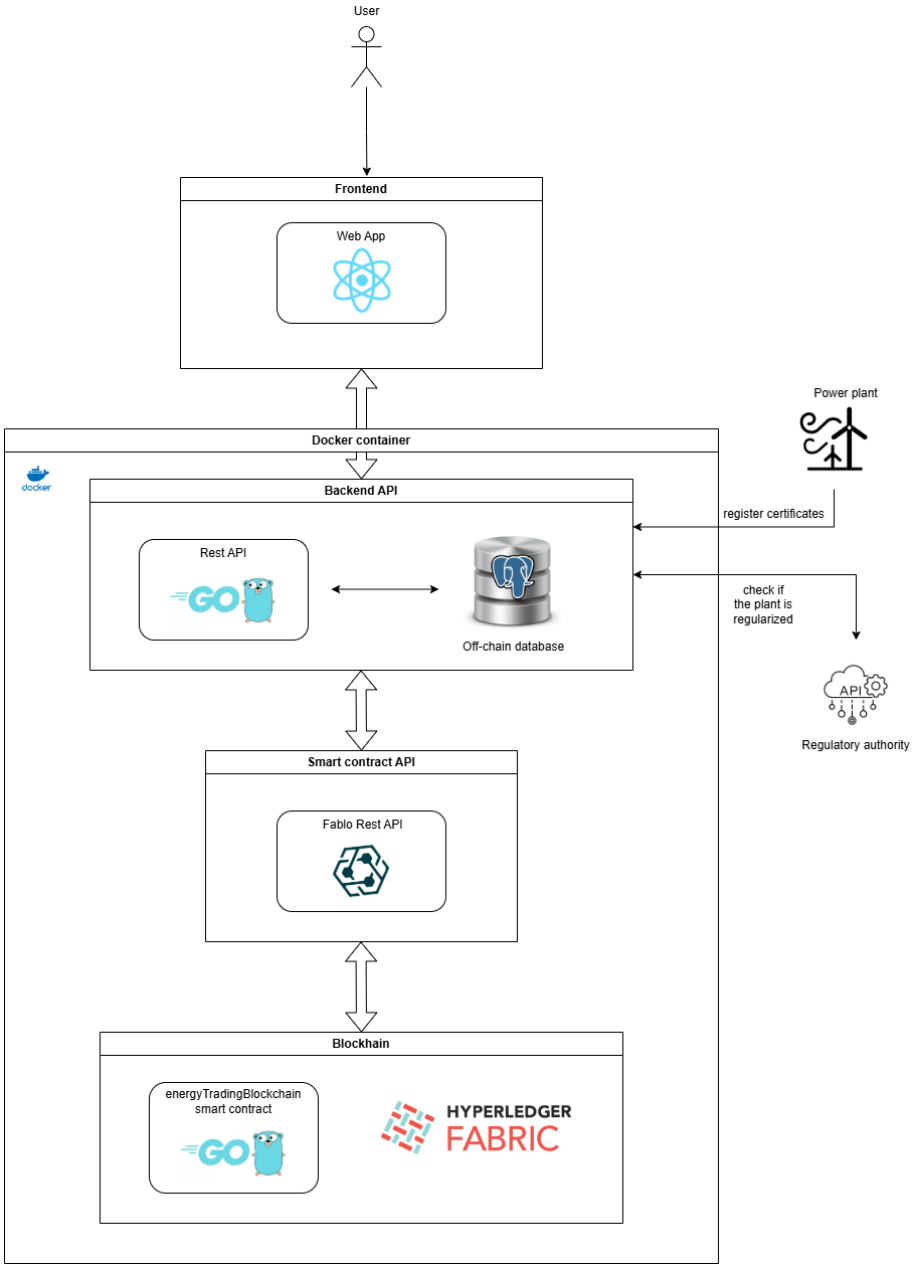

Figure 3.3: Architecture of the proposed system.

3.3.2 Backend API

The backend API is responsible for managing the platform's business logic and acts as a bridge between the frontend and the blockchain network. It is implemented using a RESTful API developed in Go, providing an efficient and scalable interface for handling user requests.

This layer connects with an off-chain PostgreSQL database to store auxiliary data that is not required on the blockchain, such as user details and operational metadata. Additionally, it integrates with external services, such as regulatory authority APIs, to validate power plants and ensure compliance with local energy regulations. The backend also facilitates the registration of energy certificates submitted by producers.

3.3.3 Smart Contract API

The smart contract API forms the core logic of the blockchain-based system. Built using Fablo REST APIs, it abstracts the interaction between the backend and the blockchain. This layer ensures that operations such as the creation, validation, and transfer of energy certificates are executed securely and immutably on the blockchain.

3.3.4 Blockchain Layer

The blockchain layer is powered by Hyperledger Fabric, a permissioned blockchain framework designed for enterprise applications. It hosts the *energyTradingBlockchain* smart contract, which defines the rules for certificate registration, ownership transfer, and transaction validation. By using a distributed ledger, this layer guarantees data integrity, transparency, and resistance to tampering.

The use of Hyperledger Fabric ensures that transactions are conducted between trusted parties, while the endorsement policies enforce consensus mechanisms for added security. Smart contracts, written in Go, govern all blockchain interactions, ensuring the correct application of business logic.

3.3.5 Integration with Power Plants and Regulatory Authorities

The architecture supports seamless integration with external entities such as power plants and regulatory authorities. Energy producers can register certificates for the energy they generate, and these certificates are validated against regulatory standards through APIs provided by the regulatory authority. This ensures that all traded energy originates from compliant and verified sources, promoting trust within the platform.

3.3.6 End-to-End Workflow

In the system's workflow, a user interacts with the frontend to perform actions such as buying or selling energy certificates. These requests are processed by the backend API, which communicates with the smart contract API for blockchain interactions or with the off-chain database for auxiliary

data management. For producers registering certificates, the backend verifies the producer's compliance via the regulatory authority's API before invoking the blockchain to register the certificate. The blockchain ensures immutability and traceability for all transactions.

This architecture ensures that the platform is robust, scalable, and secure, meeting the key requirements for a decentralized P2P energy trading system.

Chapter 4

Implementation

4.1 Implementation of the Decentralized Energy Trading Platform

This section presents the implementation details of the decentralized energy trading platform. The system leverages blockchain technology for secure and transparent transactions, while integrating off-chain storage for efficiency and scalability. The implementation is divided into several components, each described in detail.

4.1.1 Blockchain Smart Contract

The core of the platform resides in a smart contract implemented using Hyperledger Fabric. The smart contract manages energy certificates and their transactions. The primary functionalities include:

- **Energy Certificate Management:** Certificates are represented as structured data containing attributes such as `OwnerID`, `ProducerID`, `UsableMonth`, `UsableYear`, `EnergyTypeID`, `RegulatoryAuthorityID`, and `AvailableToSell`. Main methods provided by the smart contract:
  - `CreateEnergyCertificate`: Creates a new certificate on the blockchain.
  - `TransferEnergyCertificate`: Transfers ownership of a certificate while ensuring that the transaction details are recorded immutably.
  - `GetCertificatesByOwnerID`: Fetches all certificates belonging to a specific user.
  - `GetCertificatesByProducerID`: Retrieves certificates issued by a specific producer.
  - `GetCertificatesAvailableFromSpecificMonthAndEnergyType`: Retrieves certificates available to sell by month, year and energy type.
  - `GetCertificatesAvailableToSell`: Fetches certificates marked as available for sale.

  - `GetTransactionsByFromUserID`: Fetches transactions where the User was the seller.
  - `GetTransactionsByToUserID`: Fetches transactions where the User was the buyer.

- **Transaction Management:** Each transfer of a certificate generates a transaction record, which includes details such as `FromUserID`, `ToUserID`, `Price`, and `TransactionDate`.

The blockchain leverages Hyperledger Fabric's built-in features for key generation and ensures the integrity of stored data using unique transaction keys. The smart contract provides methods to:

4.1.2 Off-Chain Data Management

To improve scalability and facilitate complex queries, some data, such as user energy consumption, is stored in an off-chain PostgreSQL database. Key tables include:

- **Users:** Contains user information and roles (e.g., producer, consumer, regulatory authority).
- **Consumption:** Tracks monthly energy consumption by user, year, and month, grouped by energy type. The schema includes:

```
CREATE TABLE consumption (
    userId UUID REFERENCES users(id) ON DELETE CASCADE,
    consumptionYear INT NOT NULL,
    consumptionMonth INT NOT NULL,
    energyTypeId UUID NOT NULL,
    energyConsumed DOUBLE PRECISION NOT NULL,
    PRIMARY KEY (userId, consumptionYear,
    consumptionMonth, energyTypeId)
);
```

4.2 Backend Implementation

The backend implementation of the energy trading platform serves as the core of the system, connecting the frontend, external APIs, and blockchain network. It is responsible for managing user requests, validating inputs, interacting with the blockchain, and persisting auxiliary data in the off-chain database. The backend is primarily developed in Go and utilizes the Gin framework for HTTP API development.

4.2.1 Architecture Overview

The backend is structured around RESTful APIs that expose endpoints for creating and transferring energy certificates, validating users, and recording consumption. It interacts with the

blockchain through smart contract calls and external APIs for regulatory validations. Key operations include:

- **Create Energy Certificates:** Allows producers to create certificates for energy generated, ensuring compliance with regulatory authorities.
- **Transfer Energy Certificates:** Facilitates P2P energy trading by updating ownership of certificates on the blockchain.
- **Consumption Management:** Records user consumption details in an off-chain database to track utilization.

4.2.2 Important Smart Contract Functions

4.2.2.1 Create Energy Certificate

The `CreateEnergyCertificate` function registers a new energy certificate on the blockchain. This operation verifies the certificate details, associates them with a producer and regulatory authority, and stores them immutably on the blockchain. The following is a summary of the key steps:

- Generate a unique certificate ID using the blockchain transaction ID.
- Serialize the energy certificate as a JSON object.
- Use `PutState` to store the certificate on the blockchain ledger.

The implementation:

```
func (c *EnergyCertificateContract) CreateEnergyCertificate(
    ctx contractapi.TransactionContextInterface,
    ownerId string, producerId string, emissionDate string,
    usableMonth int, usableYear int, regulatoryAuthorityID string,
    energyType int) (string, error) {

    energyCertificate := EnergyCertificate{
        EnergyCertificateID:   ctx.GetStub().GetTxID(),
        OwnerID:               ownerId,
        ProducerID:            producerId,
        EmissionDate:          emissionDate,
        UsableMonth:           usableMonth,
        UsableYear:            usableYear,
        RegulatoryAuthorityID: regulatoryAuthorityID,
        AvailableToSell:       true,
        EnergyType:            energyType,
    }

    bytes, err := json.Marshal(energyCertificate)
```

```
    if err != nil {
        return "", err
    }

    err = ctx.GetStub().PutState(energyCertificate.EnergyCertificateID,
        bytes)
    if err != nil {
        return "",
           fmt.Errorf("failed to create energy certificate: %v", err)
    }

    return fmt.Sprintf("%s created successfully",
        energyCertificate.EnergyCertificateID), nil
}
```

Listing 4.1: Create energy certificate smart contract

4.2.2.2 Transfer Energy Certificate

The `TransferEnergyCertificate` function transfers ownership of a certificate. It validates the existence of the certificate, updates the owner, and creates a transaction record.

- Fetch the certificate by ID using `GetState`.
- Update the `OwnerID` field to the new owner.
- Record the transaction as a JSON object and store it using `PutState`.

The implementation:

```
func (c *EnergyCertificateContract) TransferEnergyCertificate(
    ctx contractapi.TransactionContextInterface,
    energyCertificateID string,
    newOwnerId string,
    price float64) (string, error) {

    certificateJSON, err := ctx.GetStub().GetState(energyCertificateID)
    if err != nil {
        return "", fmt.Errorf("failed to read certificate: %v", err)
    }
    if certificateJSON == nil {
        return "", fmt.Errorf("certificate does not exist: %s",
            energyCertificateID)
    }

    var energyCertificate EnergyCertificate
    err = json.Unmarshal(certificateJSON, &energyCertificate)
    if err != nil {
        return "", err
```

```
    }

    oldOwnerId := energyCertificate.OwnerID
    energyCertificate.OwnerID = newOwnerId
    energyCertificate.AvailableToSell = false

    updatedCertificateJSON, err := json.Marshal(energyCertificate)
    if err != nil {
        return "", err
    }

    err = ctx.GetStub().PutState(energyCertificateID,
        updatedCertificateJSON)
    if err != nil {
        return "", err
    }

    transaction := Transaction{
        TransactionID:       ctx.GetStub().GetTxID(),
        CertificateTokenRef: energyCertificate.EnergyCertificateID,
        FromUserID:          oldOwnerId,
        ToUserID:            newOwnerId,
        TransactionDate:     time.Now().Format(time.RFC3339),
        Price:               price,
    }

    transactionJSON, err := json.Marshal(transaction)
    if err != nil {
        return "", err
    }

    transactionKey := fmt.Sprintf("TRANSACTION_%s",
        transaction.TransactionID)
    err = ctx.GetStub().PutState(transactionKey, transactionJSON)
    if err != nil {
        return "", err
    }
    return fmt.Sprintf("%s transaction created successfully",
        transactionKey), nil
}
```

Listing 4.2: Transfer energy certificate smart contract

4.2.3 Backend API integration

The backend API provides REST endpoints using the Gin framework to integrate with the blockchain.

4.2.3.1 TransferEnergyCertificate

The `TransferEnergyCertificate` function serves as an API endpoint in a backend application. It facilitates the transfer of energy certificates between users in a blockchain-based energy trading system. The function's operations can be broken into the following steps:

First the function starts by retrieving the `claims` object from the request context, which contains user authentication details. If the claims are missing, an `HTTP 401 Unauthorized` response is returned. Then, the input JSON is deserialized into the `TransferCertificate` model:

```
claims, exists := c.Get("claims")
if !exists {
    c.JSON(http.StatusUnauthorized, gin.H{"error": "Unauthorized"})
    return
}

err := c.ShouldBindJSON(&transferCertificateBody)
if err != nil {
    c.JSON(400, gin.H{
        "error": "Cannot bind transferCertificate JSON: " + err.Error(),
    })
    return
}
```

Listing 4.3: Retrieve user claims

The function uses the helper `getEnergyCertificate` to fetch the certificate details by invoking the blockchain contract method `ReadEnergyCertificate`. This involves making a POST request with the certificate ID and parsing the response:

```
energyCertificate, err := getEnergyCertificate(
    transferCertificateBody.EnergyCertificateID,
    blockchainToken,
    transferEnergyCertificateUrl,
)
if err != nil {
    c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
    return
}
```

Listing 4.4: Retrieve energy certificate

The `getTransferPrice` function sends a POST request to the mock server with the `quantity` and `availability`, and parses the calculated price from the response.

Next, the actual transfer of the certificate is performed by calling the blockchain contract method `TransferEnergyCertificate`. If this step fails, an error is returned:

```
if err := transferEnergyCertificate(
    transferCertificateBody.EnergyCertificateID,
```

```
    userID,
    price.Price,
    blockchainToken,
    transferEnergyCertificateUrl,
); err != nil {
    c.JSON(http.StatusBadRequest, gin.H{
        "error": err.Error()
        })
    return
}
```

Listing 4.5: Call to transfer energy certificate

Finally, the function updates the energy consumption records for both the old and new owners in the database using the `createConsumption` function:

```
if err := createConsumption(
    userID,
    energyCertificate.UsableYear,
    energyCertificate.UsableMonth,
    energyCertificate.EnergyType,
); err != nil {
    c.JSON(http.StatusInternalServerError, gin.H{
        "error": "Failed to create consumption record: " + err.Error()
        })
    return
}

if err := createConsumption(
    oldOwnerId,
    energyCertificate.UsableYear,
    energyCertificate.UsableMonth,
    energyCertificate.EnergyType,
); err != nil {
    c.JSON(http.StatusInternalServerError, gin.H{
        "error": "Failed to create consumption record: " + err.Error()
        })
    return
}
```

Listing 4.6: Create consumptions records for seller and buyer

The price for the transfer is calculated based on the energy quantity and its availability by calling an helper function.

```
price, err := getTransferPrice(
    transferCertificateBody.Quantity,
    transferCertificateBody.Availability,
)
if err != nil {
    c.JSON((http.StatusInternalServerError), gin.H{"error": err.Error()})
```

```
    return
}
```

Listing 4.7: Get transferPrice call body

```
var PRICES_INTERVAL = struct {
        MinPrice float64
        MaxPrice float64
}{
        MinPrice: 4.0,
        MaxPrice: 8.0,
}

func getTransferPrice(quantity int, availability int) (float64, error) {
        if quantity <= 0 || availability <= 0 {
                return 0, errors.New("quantity and availability must be positive
                    ")
        }

        minPrice := PRICES_INTERVAL.MinPrice
        maxPrice := PRICES_INTERVAL.MaxPrice
        var price float64

        if quantity <= availability {
                price = minPrice + rand.Float64()*(maxPrice-minPrice)
        } else {
                scarcityFactor :=
           math.Min(1.0, float64(quantity)/float64(availability))
                basePrice := minPrice + scarcityFactor*(maxPrice-minPrice)
                variation := rand.Float64()*5 - 2.5
                price = basePrice + variation
        }

        price = math.Round(price*100) / 100

        return price, nil
}
```

Listing 4.8: Calculate transferPrice

- **Validation:** The function first ensures that both `quantity` and `availability` are greater than zero. If not, an error is returned.
- **Base Price Calculation:**
  - If `quantity` is less than or equal to `availability`, the price is a random value between `minPrice` and `maxPrice`.

  - If `quantity` exceeds `availability`, a scarcity factor is calculated as:

$$\text{scarcityFactor} = \min\left(1.0, \frac{\text{quantity}}{\text{availability}}\right)$$

    This factor is used to increase the base price, reflecting the higher demand relative to availability.

- **Random Variation:** Since in development phase we don't have many transactions to calculate a base price, a random variation in the range of -2.5 to $+2.5$ is added to introduce slight unpredictability in the price.

Summary of Helper Functions

The `TransferEnergyCertificate` function relies on several helper functions:

- `getEnergyCertificate`: Fetches certificate details from the blockchain.
- `getTransferPrice`: Calculates the price for the energy transfer.
- `transferEnergyCertificate`: Executes the transfer operation on the blockchain.
- `createConsumption`: Updates the consumption records in the database.

4.2.4 Authentication and Authorization

The platform incorporates a role-based access control (RBAC) system. During login, a JWT token is issued, embedding the user's ID and role. Middleware ensures that only authorized roles can perform specific actions, such as certificate creation or transfer.

The following function, `Auth`, implements middleware for userType-based authorization in a Gin web framework. The middleware validates JSON Web Tokens (JWTs) and optionally checks if the `userType` embedded in the token matches a required `userType`. The function dynamically allows specifying a required `userType` during its invocation.

```
1  func Auth(requiredUserType ...string) gin.HandlerFunc {
2      return func(c *gin.Context) {
3          const BearerSchema = "Bearer "
4          header := c.GetHeader("Authorization")
5          if header == "" || !strings.HasPrefix(header, BearerSchema) {
6              c.AbortWithStatus(401)
7              return
8          }
9
10         tokenString := header[len(BearerSchema):]
11
12         parsedToken, err :=
13              services.NewJWTService().ValidateToken(tokenString)
```

```
14          if err != nil || !parsedToken.Valid {
15              c.AbortWithStatus(401)
16              return
17          }
18
19          if claims, ok := parsedToken.Claims.(jwt.MapClaims); ok {
20              c.Set("claims", claims)
21
22              tokenUserType, ok := claims["userType"].(string)
23              if !ok {
24                  c.AbortWithStatusJSON(403, gin.H{
25                       "error": "userType missing or invalid in token"
26                       })
27                  return
28              }
29
30              if len(requiredUserType) > 0 {
31                  expectedUserType := requiredUserType[0]
32                  if tokenUserType != expectedUserType {
33                      c.AbortWithStatusJSON(403, gin.H{
34                         "error": "Access forbidden: incorrect userType"
35                         })
36                      return
37                  }
38              }
39          } else {
40              c.AbortWithStatus(401)
41              return
42          }
43
44          c.Next()
45      }
46  }
```

Listing 4.9: Authentication handling

4.2.5 Summary

The backend implementation ensures secure interactions with the blockchain and enforces the business logic of the energy trading system. By combining smart contract calls, RESTful APIs, and off-chain database integrations, it provides a scalable, efficient, and compliant foundation for the platform.

4.3 Frontend Integration

The project is designed as a modern web application leveraging a React.js front-end. The system follows a modular architecture, ensuring scalability, reusability, and maintainability. The primary

features implemented include:

- Dynamic visualization of consumption and transaction data through charts.
- User authentication and role-based access control.
- Interactive filters and real-time data updates.

The front-end integrates with third-party libraries such as Recharts for data visualization and Axios for HTTP requests.

4.3.1 Login and Dashboard

The application begins with a login screen (Figure 4.1), ensuring that only authenticated users can access the system.

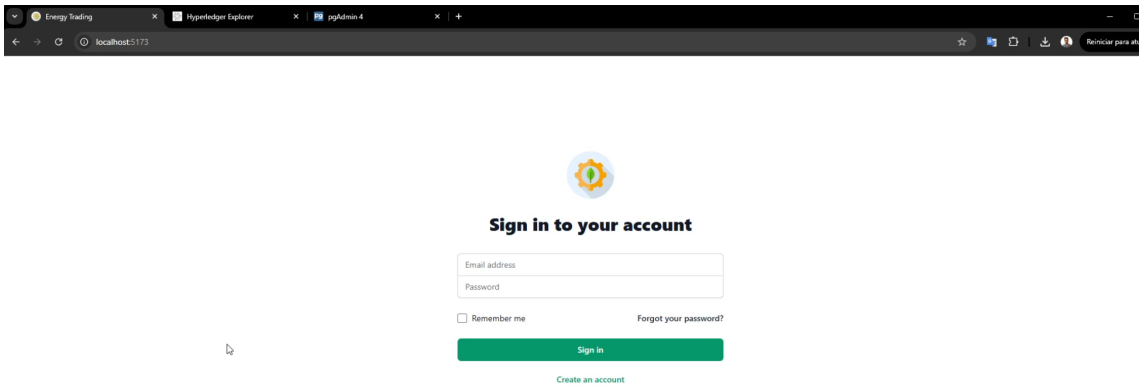

Figure 4.1: Login Screen web app.

4.3.2 Consumption Data Visualization

Upon successful login, the user is directed to the main dashboard (Figure 4.2), which provides an overview of energy consumption and transactions. Consumption data is aggregated and displayed in various formats such as bar charts and pie charts. The system handles the following cases:

- Energy consumption trends over time, broken down by energy type (e.g., Solar, Wind, Hydro).
- Aggregated transaction data, showing metrics such as total spending and average spending per day.

Dynamic Data Transformation: The consumption data is received in an array format, where each entry represents a unique combination of year, month, and energy type. This data is transformed into structures suitable for charting:

- Aggregated by year and month for bar charts.
- Summarized by energy type for pie charts.

The transformation logic ensures that data is grouped, sorted, and labeled appropriately, enabling clear and meaningful visualizations.

The charts are designed to be visually intuitive and dynamically update based on the user's selected filters, such as the year.

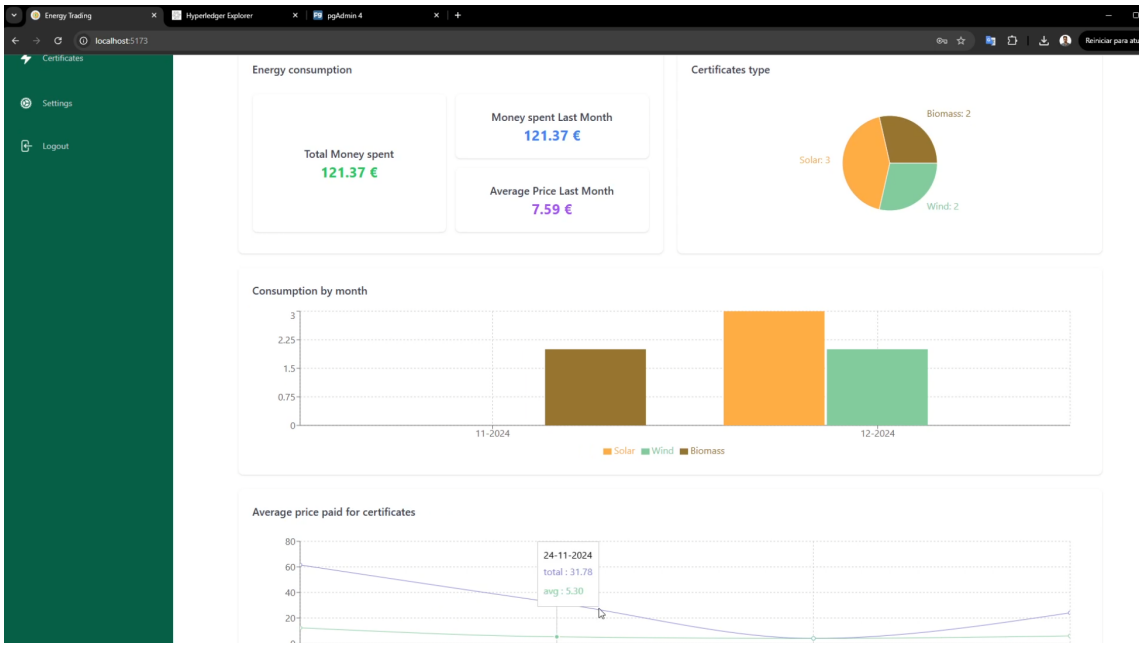

Figure 4.2: Dashboard web app.

Handling of Multiple Energy Types: Each energy type is assigned a unique color for consistent representation across all charts. A mapping is maintained between energy type labels and their corresponding colors.

4.3.3 Transaction Data Aggregation

The transaction data processing pipeline is designed to compute key metrics such as total amount spent and average spent per month. The following challenges and solutions were addressed:

Calculation of Metrics: For each month:

- The total expenditure is calculated by summing the `price` field of transactions.

- The average expenditure is derived by dividing the total expenditure by the number of transactions in the month.

These metrics are displayed in both textual and graphical form (Figure 4.3).

Filtering and Sorting: Users can filter consumption data by year through a dropdown menu. The implementation ensures that only the last 12 months of data are displayed, sorted from the most recent month to the oldest.

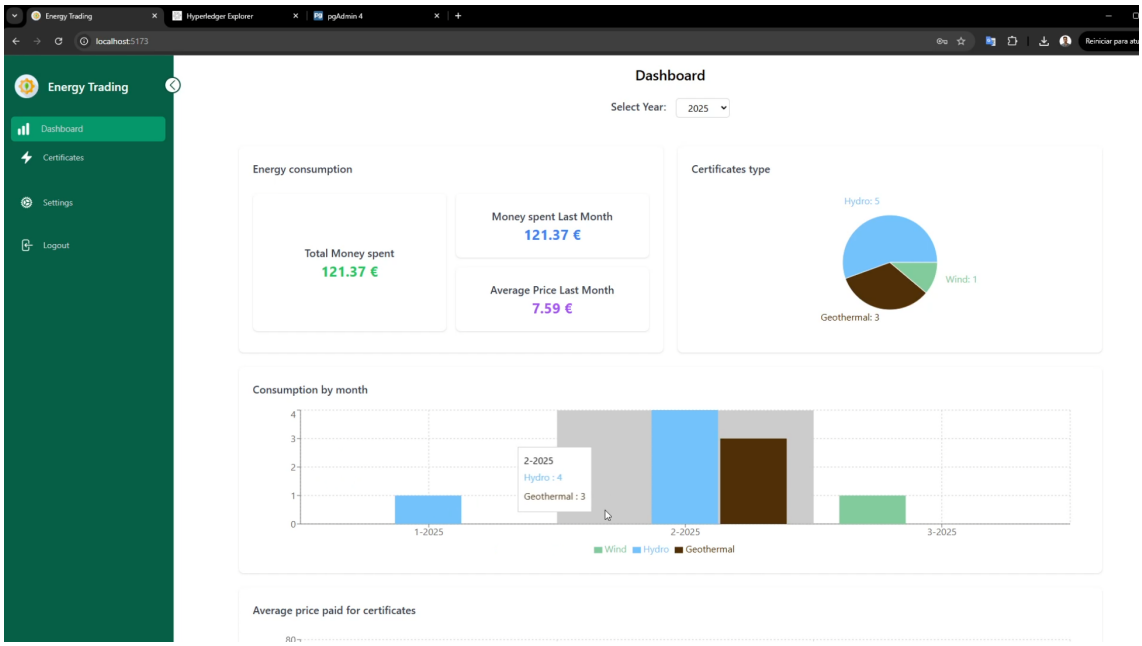

Figure 4.3: Dashboard filtered by year

4.3.4 Certificates

The certificates screen allows users to manage renewable energy certificates. This screen has two primary functionalities:

- **View Owned Certificates:** Users can see a detailed list of certificates they currently own, including relevant metadata such as certificate type and issuance date (Figure 4.4).
- **Request Certificates:** Users can explore available certificates on the marketplace and complete purchases directly through the web app (Figure 4.5).
- **Buy Certificates:** Users can buy multiple certificates and see an prediction of the price paid for certificate (Figure 4.6).
- **Create Certificates:** Energy Producers can create multiple certificates of the same energy type (Figure 4.7).

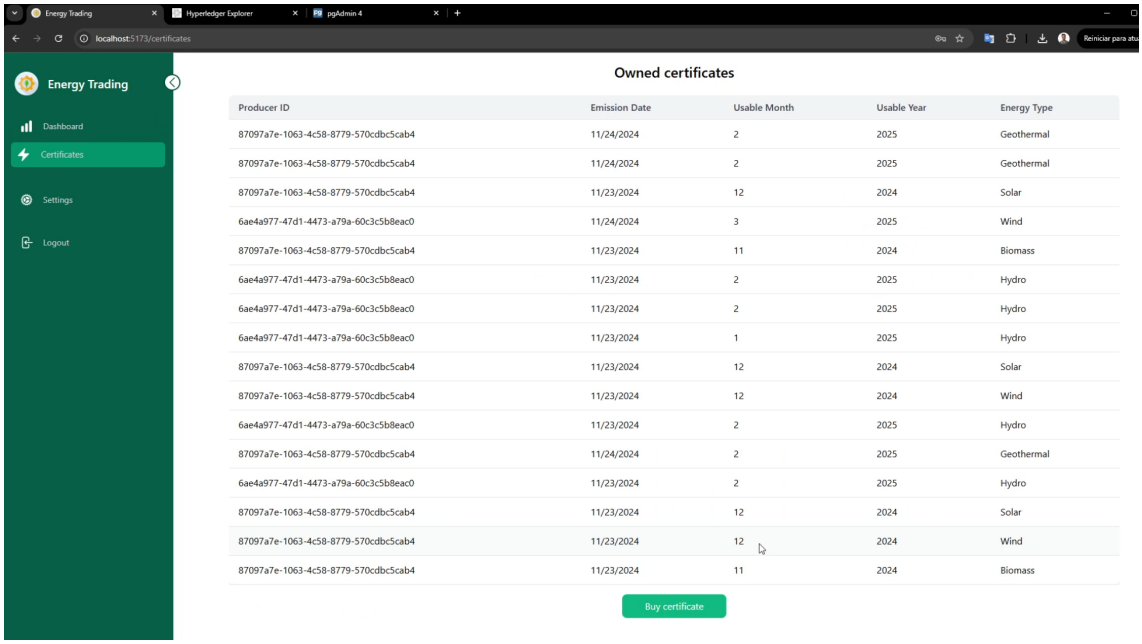

Figure 4.4: Owned certificates screen

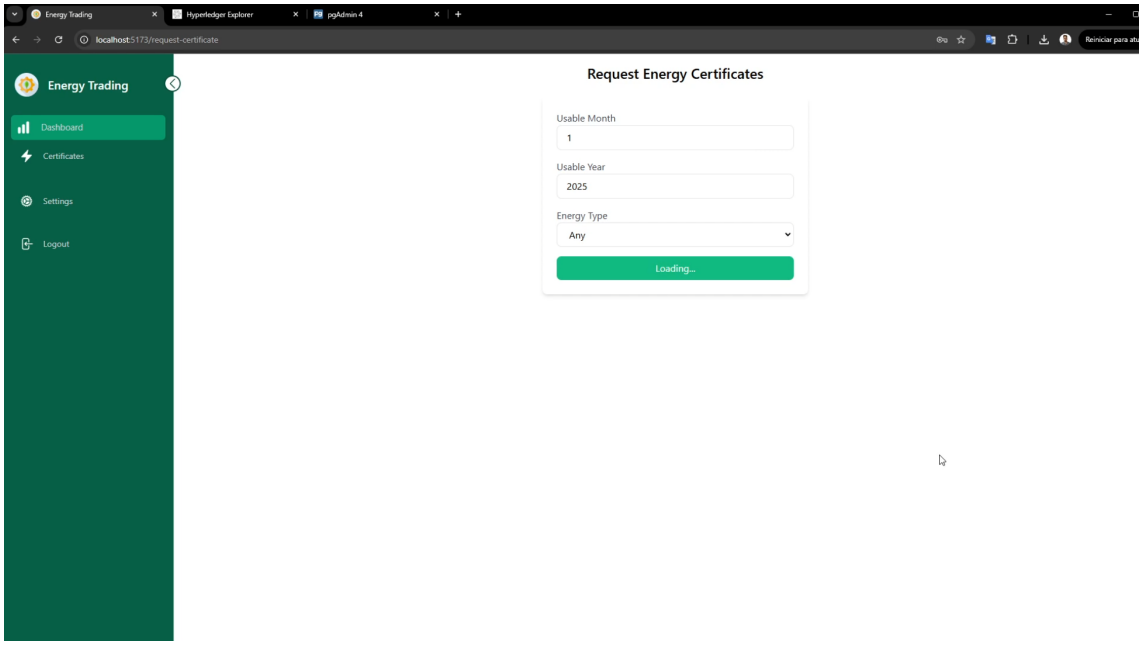

Figure 4.5: Request certificates by energy type, usable month and year

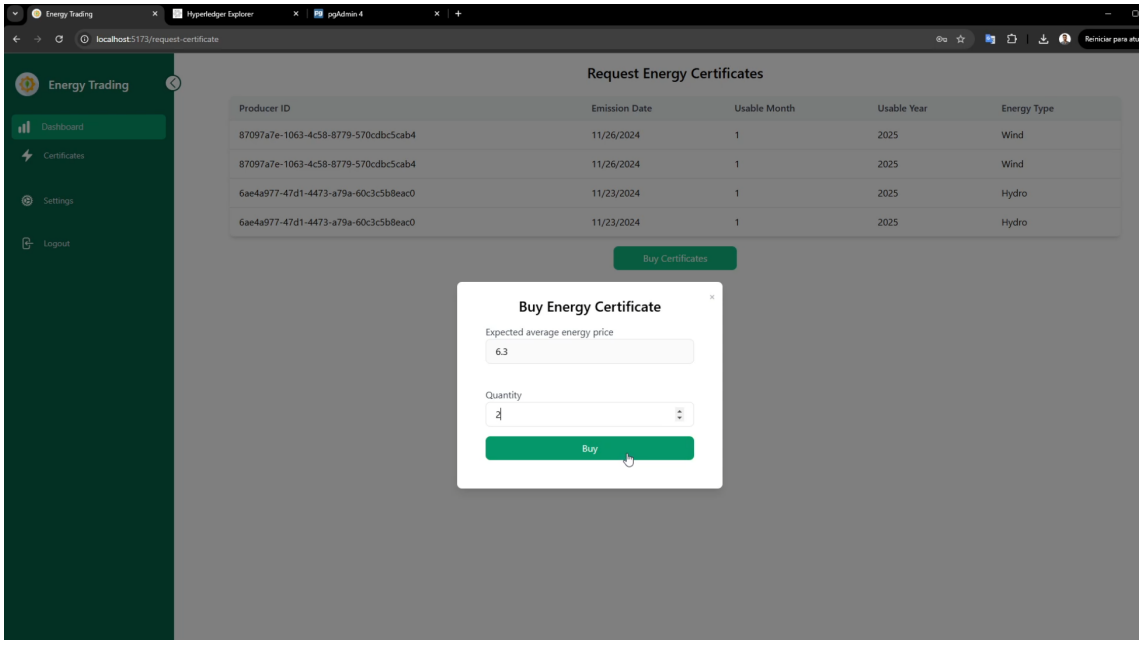

Figure 4.6: Buy certificate and energy price prediction

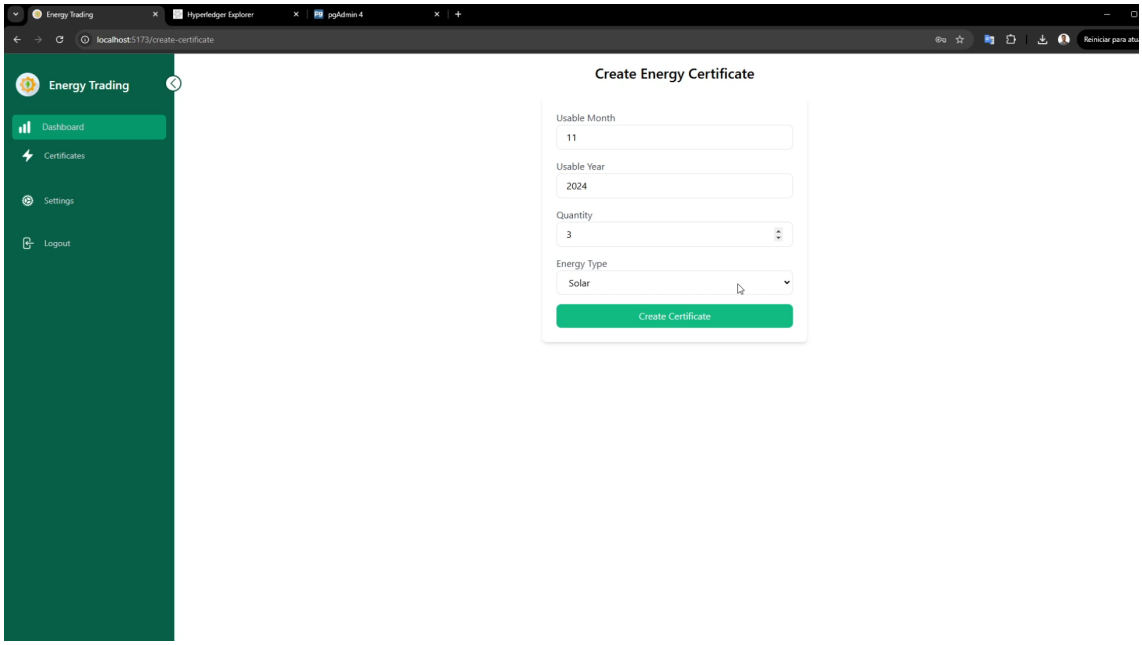

Figure 4.7: Create certificate

4.3.5 Error Handling and Edge Cases

The application is designed to handle edge cases gracefully, including:

- **Invalid Data:** When no data is available for visualization, placeholders and error messages are displayed.
- **API Failures:** Errors from the back-end, such as failed requests or invalid responses, are caught and displayed to the user.
- **User Interaction:** Input validation ensures that only valid quantities are submitted for predicted price calculations.

4.4 Docker Environment

To make sure the application is able to run in any machine, we used a Docker Compose file to orchestrate multiple services in the energy trading application. The file defines services for a PostgreSQL database, a pgAdmin interface, a backend API, and a mock server. Using Docker Compose ensures seamless integration and simplified management of the application's environment.

4.4.1 Key Components of the Docker Compose File

4.4.1.1 PostgreSQL Database (`db`)

- **Image**: Utilizes the `postgres` image for a reliable and production-ready database solution.
- **Environment Variables**:
  - Database credentials (`POSTGRES_PASSWORD`, `POSTGRES_USER`, and `POSTGRES_DB`) are sourced from an `.env` file, ensuring flexibility and security.
- **Port Mapping**: Maps the database's port `5432` to the host's port defined as `${DB_PORT}` in the `.env` file.
- **Volume**: Data is persisted in `./postgres-data`, preventing loss during container restarts.
- **Network**: Connected to `app-network` for secure inter-service communication.

4.4.1.2 pgAdmin (`pgadmin`)

- **Image**: Uses the `dpage/pgadmin4` image for a web-based PostgreSQL administration interface.
- **Environment Variables**:

  - Admin credentials (`PGADMIN_DEFAULT_EMAIL` and `PGADMIN_DEFAULT_PASSWORD`) are sourced from the `.env` file.

- **Port Mapping**: Maps port `80` in the container to `5050` on the host.
- **Volume**: Configuration data is persisted in `./pgadmin-data`.
- **Network**: Connected to `app-network` for direct interaction with the database.

4.4.1.3 Energy Trading Blockchain API (`energyTradingBlockchain-api`)

- **Image**: A custom image (`energy-trading-blockchain-api`) containing the backend logic for energy trading.
- **Environment Variables**:
  - Configures database access, blockchain settings, and mock server URL, sourced from the `.env` file.
- **Port Mapping**: Maps port `5000` in the container to `8080` on the host for external access.
- **Network**: Connected to `app-network` for communication with other services.

4.4.1.4 Mock Server (`mock-server`)

- **Image**: A custom image (`energy-trading-mock-server`) simulates external dependencies or APIs.
- **Port Mapping**: Maps port `3000` in the container to `13000` on the host.
- **Network**: Connected to `app-network` for integration with the other components.

The mock server is an integral part of the development and testing workflow for the energy trading application. Its primary purpose is to simulate the behavior of calls to the regulatory authority. By mimicking these external interactions, the mock server ensures that the application can be developed and tested in a controlled and predictable environment without relying on the actual regulatory authority's services.

4.4.1.5 Purpose of the Mock Server

- **Simulation of External APIs**: The mock server replicates the endpoints and responses of the regulatory authority, enabling the application to interact with realistic, predefined responses.
- **Controlled Environment**: By using the mock server, developers can test edge cases and unexpected scenarios that might not be feasible with the live regulatory system.
- **Decoupling Dependencies**: Development and testing can proceed independently of the availability or response times of the actual regulatory authority.

4.4.2 Why This Docker Compose File Was Used

1. **Simplified Development Environment**: Ensures all components (database, API, mock server, and pgAdmin) run with consistent configurations, reducing manual setup effort.
2. **Portability**: Enables seamless replication of the environment across different development or testing setups.
3. **Isolation**: Each service runs in its own container, avoiding dependency conflicts.
4. **Persistence**: Data and configurations are stored in volumes, preventing data loss during restarts.
5. **Scalability**: The setup can be easily scaled by modifying the service definitions (e.g., increasing API replicas).
6. **Ease of Networking**: The custom `app-network` ensures all services communicate securely without external exposure.
7. **Testing with Mock Servers**: A mock server is included to simulate external systems, enabling controlled testing.

4.4.3 Example Workflow

1. The `db` service provides a PostgreSQL instance for data storage.
2. The `pgadmin` service offers a graphical interface for database management.
3. The `energyTradingBlockchain-api` processes business logic and interacts with the `db` and `mock-server`.
4. The `mock-server` simulates external APIs, ensuring the API functions correctly in isolation.

4.4.4 Challenges and Future Work

Key challenges included ensuring synchronization between on-chain and off-chain data, optimizing query performance, and securing user authentication. Future improvements could involve:

- Implementing energy certificate expiration and automated notifications.
- Integrating with renewable energy providers for real-time certificate generation.
- Exploring cross-chain interoperability for broader energy trading.

Chapter 5

Validation and discussion

The validation process is a critical component of this project, ensuring that the energy trading application meets its functional and non-functional requirements. Through comprehensive validation strategies, the system is tested for correctness, robustness, and alignment with real-world scenarios. To ensure Functional Accuracy and Data Integrity, we utilized Postman (Appendix B.1) to validate API responses and Hyperledger Fabric Explorer (Appendix B.2 and B.3) to verify that objects were correctly stored on the blockchain.

5.1 Validation Objectives

The primary objectives of the validation phase are as follows:

- **Functional Accuracy**: Verify that all implemented functionalities, such as energy certificate transfers and price calculations, work as intended.
- **Performance and Scalability**: Ensure that the system can handle concurrent requests efficiently without performance degradation.
- **Regulatory Compliance**: Confirm that the application aligns with the requirements simulated by the mock server, representing regulatory authority behavior.
- **Data Integrity**: Validate that the system maintains the consistency and accuracy of data, especially in transactions and database operations.

5.2 Validation Techniques

End-to-end (E2E) testing ensures that the entire system functions as intended when all components are integrated. This type of testing simulates actual user scenarios and validates the system's behavior holistically. The following use cases were executed to validate real-world workflows:

- **A user initiates an energy certificate transfer**: This test simulates a user transferring an energy certificate to another entity. It verifies that the user interface, API, and backend systems interact correctly, ensuring that all necessary validations, such as token authentication and user role checks, are performed before the transfer is processed.
- **The system calculates the transfer price**: This test validates the logic within the function `getTransferPrice`. It ensures that the price calculation is accurate based on the availability of resources and requested quantity. Additionally, edge cases, such as extremely low availability or abnormally high quantities, were tested to confirm the robustness of the pricing logic.
- **The certificate's ownership is updated in the blockchain, and consumption records are adjusted accordingly**: This test ensures that the transfer operation triggers blockchain updates to change certificate ownership. Furthermore, the test validates that both the new and previous owners' consumption records are updated in the database accurately, reflecting the transaction's completion.

5.2.1 Load Testing

Load testing was conducted to assess the system's performance under varying levels of concurrent usage. This testing aimed to identify bottlenecks and measure the system's capacity to handle peak loads without failure.

- **Simulating high traffic scenarios**: Multiple concurrent requests were sent to the system, focusing on key endpoints such as energy certificate transfers and price calculations. This test determined the maximum number of simultaneous users the system could support while maintaining acceptable response times.
- **Monitoring system resources**: Metrics such as CPU usage, memory consumption, and database query performance were analyzed during peak loads. This evaluation ensured that resource utilization remained within manageable limits, avoiding overloading and system crashes.
- **Identifying potential bottlenecks**: By incrementally increasing the number of requests, specific areas of performance degradation were identified. This information was used to optimize database queries, reduce response times, and enhance overall scalability.

5.3 Identified Problems

During the validation and testing phase, certain performance-related issues were identified that could impact the overall efficiency of the system. One notable concern was the performance disparity between the Fabric REST API and the GoLang API. Specifically, the Fabric REST API demonstrated significantly slower response times compared to the GoLang API, which created

a potential bottleneck in the system. This discrepancy was particularly evident when retrieving information, where accessing data from our local database proved to be much faster and more efficient.

A contributing factor to this issue lies within our implementation code. Currently, some of the logic resides in the application layer, which results in repeated calls to the Fabric API for the same data. This redundancy not only increases the response time but also puts unnecessary load on the Fabric network.

To mitigate this issue, one potential solution is to move some of the business logic to the smart contracts on the blockchain. By embedding the logic directly into the smart contracts, we can reduce the number of API calls required and minimize redundancy. This approach would allow critical operations to be executed closer to the source of the data, thereby improving the overall performance and responsiveness of the system. Additionally, this would ensure that the blockchain network is utilized more efficiently while reducing latency for end-users.

Chapter 6

Conclusion

The development of this energy trading platform has been a comprehensive exploration of modern software engineering and blockchain integration, addressing the critical need for efficient, transparent, and secure energy certificate transactions. The project successfully leverages blockchain technology to ensure data integrity and traceability, while a robust API design connects various components seamlessly. Through this endeavor, we have demonstrated the feasibility and potential impact of blockchain-based energy trading systems in real-world applications.

Despite these achievements, several pain points were identified during the development and validation phases. The most significant challenge was the performance bottleneck introduced by the Hyperledger Fabric REST API. Its slower response times, compared to the locally hosted GoLang API, created inefficiencies in critical workflows such as retrieving and processing energy certificate data. This performance gap highlighted the limitations of relying on external APIs for operations that require high throughput and low latency.

Another pain point involved the redundancy in logic within the application layer. Multiple repeated calls to the Fabric REST API for identical operations increased overall latency and stressed the blockchain network unnecessarily. This inefficiency underscores the importance of careful design when integrating blockchain with traditional application architectures.

From a user perspective, the system fulfilled its functional requirements but still presents opportunities for improvement. The user experience could benefit from more intuitive interfaces for both buyers and sellers, as well as additional features such as predictive analytics for pricing and market trends.

6.1 Achievements

The primary objective of this project was to develop a blockchain-based energy trading platform capable of ensuring secure, transparent, and efficient transactions of energy certificates. This objective has been largely satisfied, as demonstrated through the functional implementation of core features, validation results, and overall system performance.

Key achievements include the successful integration of Hyperledger Fabric to maintain the immutability and transparency of energy certificate transactions. The use of smart contracts ensured the automated enforcement of business rules, reducing the risk of human error or tampering. The system's API design facilitated seamless interaction between components, enabling functionalities such as energy certificate transfers, price calculation, and consumption tracking.

While the project met its core objectives, it also provided valuable insights into areas requiring improvement. Performance bottlenecks, particularly in interactions with the Hyperledger Fabric REST API, were identified as a limitation. This highlights the need for optimization to meet the secondary objectives of peak performance and minimal latency.

Overall, the project successfully satisfied its primary objectives while providing a strong foundation for further enhancements. The implementation showcases the potential of blockchain in transforming nergy markets, paving the way for future advancements and broader adoption of similar systems.

6.2 Future Work

Future work on this platform can address the identified issues and enhance the system further. One immediate improvement involves optimizing the interaction with the blockchain network. By moving critical business logic from the application layer to smart contracts, the platform can reduce its dependency on repeated API calls. This approach would enhance performance, minimize latency, and provide a more streamlined and scalable architecture.

Another area for development is the exploration of alternative blockchain technologies or optimizations within Hyperledger Fabric itself. Implementing private data collections or more efficient consensus mechanisms could potentially mitigate some of the observed performance challenges.

Additionally, the platform could be expanded to include advanced analytics and forecasting tools. Machine learning models could analyze market trends and predict energy demand, allowing participants to make more informed trading decisions. Integration with IoT devices could further enhance the platform by enabling real-time monitoring of energy production and consumption.

Security remains a priority for future enhancements. Implementing advanced authentication mechanisms, such as multi-factor authentication and zero-knowledge proofs, could provide additional layers of protection for user data and transactions.

Lastly, fostering partnerships with regulatory authorities and energy providers could increase the platform's adoption and legitimacy. By aligning the system's design with industry standards and legal frameworks, the platform can better position itself for real-world deployment.

6.3 Final Remarks

This project has been an invaluable opportunity to bridge the gap between theoretical concepts and practical applications. While the challenges were significant, they provided crucial learning experiences that will inform future developments in blockchain-based solutions. With targeted

improvements and continued innovation, this platform has the potential to become a cornerstone in the evolving landscape of sustainable energy trading.

References

[1] Nurzhan Zhumabekuly Aitzhan and Davor Svetinovic. Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams. *IEEE Transactions on Dependable and Secure Computing*, 15(5):840–852, 2018.

[2] REC Energy Certificate Association. Understanding eac markets. Technical report, 12 2023.

[3] J. Alejandro F. Castellanos, Debora Coll-Mayor, and José Antonio Notholt. Cryptocurrency as guarantees of origin: Simulating a green certificate market with the ethereum blockchain. In *2017 IEEE International Conference on Smart Energy Grid Engineering (SEGE)*, pages 367–372, 2017.

[4] James Chen. Renewable energy certificate (rec). Technical report, 7 2024.

[5] António Miguel Rosado da Cruz and Estrela Ferreira Cruz. Blockchain-based traceability platforms as a tool for sustainability. In *International Conference on Enterprise Information Systems*, 2020.

[6] Gold energy. Certificados de energia renovável (rec). Technical report, 1 2024.

[7] IBM. What is blockchain technology? Technical report, 12 2023.

[8] R. Leonhard. Developing renewable energy credits as cryptocurrencies on ethereum's blockchain. *Social Science Research Network*, pages 3–13, 12 2016.

[9] Masoud Nazari and Antar Biswas. Decentralized p2p trading based on blockchain for retail electricity markets. 02 2024.

[10] Álvaro Nofuentes, Juan José Hernández, and Lucas Pons. Blockchain-based guarantees of origin issuing platform. In *2022 18th International Conference on the European Energy Market (EEM)*, pages 1–4, 2022.

[11] Morgen Peck and David 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. *IEEE Spectrum*, 54:56–61, 10 2017.

[12] Ion Plumb and Andreea-Ileana Zamfir. A comparative analysis of green certificates markets in the european union. *Management of Environmental Quality: An International Journal*, 20:684–695, 09 2009.

[13] U.S. Environmental Protection Agency. Renewable energy certificates (rec), 2024. Retrieved January 15, 2024, from https://www.epa.gov/green-power-markets/renewable-energy-certificates-recs.

[14] Shen Wang, Ahmad Taha, Jianhui Wang, Karla Kvaternik, and Adam Hahn. Energy crowdsourcing and peer-to-peer energy trading in blockchain-enabled smart grids. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, PP:1–12, 06 2019.

[15] Santos T. R. Carvalho A. P. Yamaguchi, J. A. R. Blockchain technology in renewable energy certificates in brazil. *BAR - Brazilian Administration Review*, PP:3–18, 11 2021.

Appendix A

Github

A.1 Smart contract

https://github.com/kevinDomingues/energyTradingBlockchain

A.2 Backend Golang API

https://github.com/kevinDomingues/energyTradingBlockchainAPI

A.3 Web application

https://github.com/kevinDomingues/react-energy-trading

Appendix B

Printscreens

B.1 Postman collection

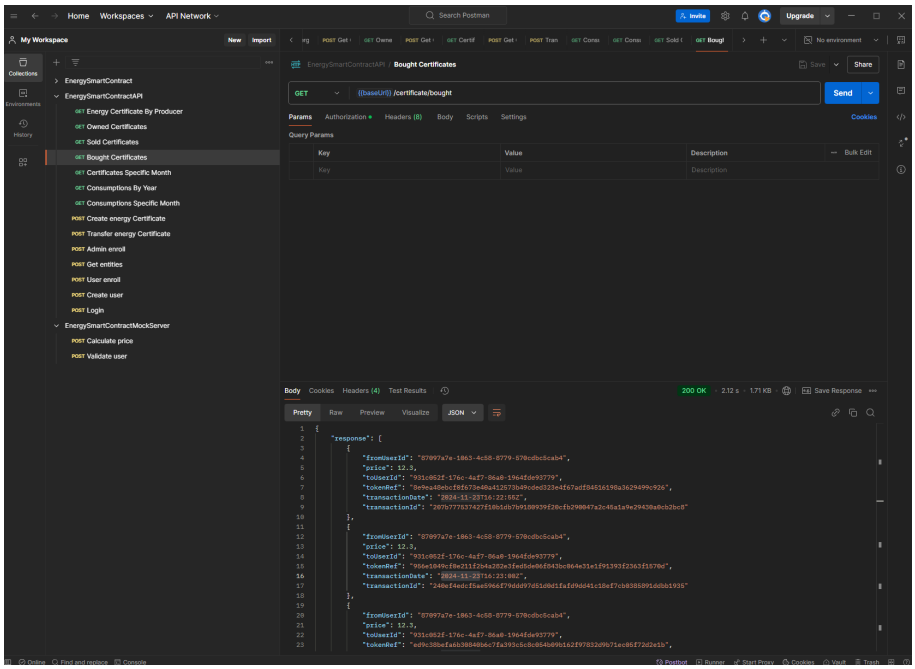

Figure B.1: Postman collection to test the API.

B.2 Hyperledger Explorer

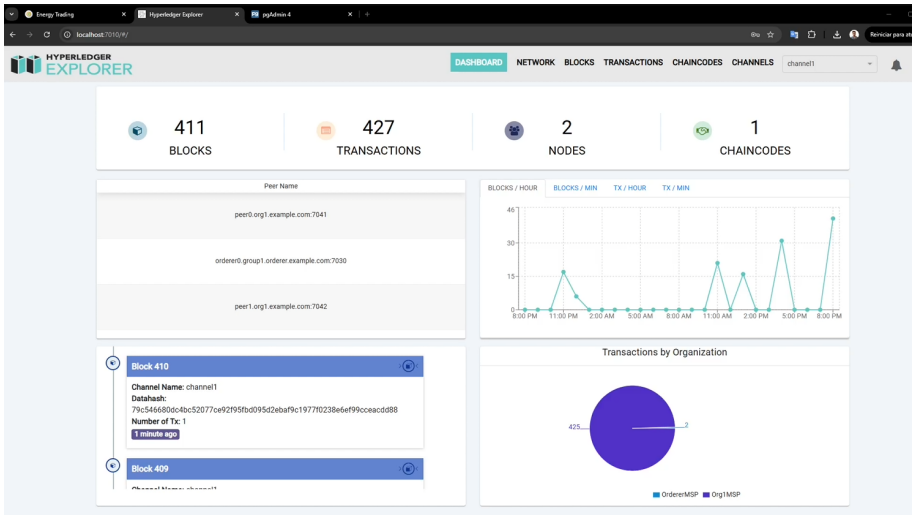

Figure B.2: Hyperledger Explorer dashboard.

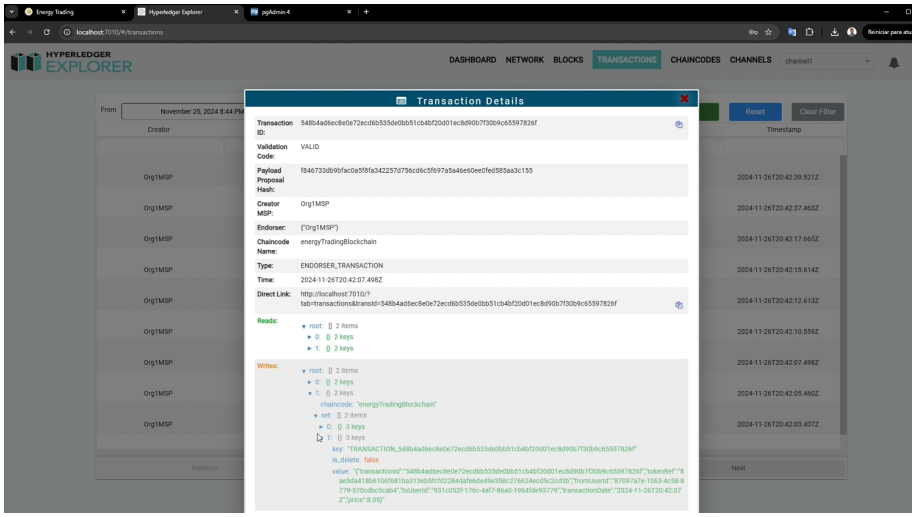

Figure B.3: Blockchain register of a transaction