Funded by
the European Union

HORIZON-CL5-2022-D4-02

EUROPEAN COMMISSION

European Climate, Infrastructure and Environment Executive Agency

Grant agreement no. 101123238

VELIXIA
BUILDINGS AS ACTIVE UTILITY NODES

Smart Grid-Efficient Interactive Buildings

Deliverable D3.5

BLOCKCHAIN Smart Flexibility Contracting

Project acronym	EVELIXIA
Full title	Smart Grid-Efficient Interactive Buildings
Grant agreement number	101123238
Topic identifier	HORIZON-CL5-2022-D4-02-04
Call	HORIZON-CL5-2022-D4-02
Funding scheme	HORIZON Innovation Actions
Project duration	48 months (1 October 2023 – 30 September 2027)
Coordinator	ETHNIKO KENTRO EREVNAS KAI TECHNOLOGIKIS ANAPTYXIS (CERTH)
Consortium partners	CERTH, RINA-C, CEA, CIRCE, UBE, HAEE, IESRD, UNIGE, SOLVUS, R2M, EI-JKU, FHB, EEE, EG, ÖE, PINK, TUCN, DEER, TN, ENTECH, SDEF, EGC, KB, AF, Sustain, NEOGRID, MPODOSAKEIO, DHCP, HEDNO, BER, MEISA, ITG, NTTDATA, TUAS, NEOY, HES-SO
Website	https://www.evelixia-project.eu/
Cordis	https://cordis.europa.eu/project/id/101123238

Disclaimer

Copyright Message

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon Europe Framework Programme for Research and Innovation under grant agreement no 101123238. ***Disclaimer:*** *The European Commission is not responsible for any use made of the information contained herein. The content does not necessarily reflect the opinion of the European Commission.*

Deliverable D3.5

BLOCKCHAIN Smart Flexibility Contracting

Deliverable number	D3.5
Deliverable name	BLOCKCHAIN Smart Flexibility Contracting
Lead beneficiary	CERTH
Description	This deliverable is directly linked to the activities foreseen in Task 3.4 and Task 3.5, consolidating all foreseen technical developments on EVELIXIA's smart-marketplace-contracting mechanisms and blockchain security mechanisms. This report is considered as the first version of D3.6.
WP	3
Related task(s)	T3.4, T3.5
Type	Document, Report
Dissemination level	Public
Delivery date	09.04.2025.
Main author	George Gogos (CERTH)
Contributors	Alberto Jose Moreno Perez (CIRCE)

Document history

Version	Date	Changes	Author
V1 – first draft	20.01.2025		CERTH
V1 – reviews	24.01.2025	Review and contribution	CIRCE
V1 – consolidated version	29.01.2025	Updated version with format changes and information enhancement	CERTH
2nd review	15.02.2025	Review from deliverable reviewers	UBE, SOLVUS
V2 – second draft	21.02.2025	Updated version based on reviewers' comments	CERTH
Final version	30.03.2025		CERTH
Final deliverable submission	09.04.2025		CERTH

EXECUTIVE SUMMARY

The EVELIXIA project aims to transform buildings into **Buildings as Active Utility Nodes (BAUNs)**. This transformation enables energy-efficient, connected, and flexible systems that promote sustainability, grid resilience, and active citizen engagement. This deliverable, titled " BLOCKCHAIN Smart Flexibility Contracting " and classified under Task 3.4, T3.5 of Work Package 3 (EVELIXIA's Middleware Layer and Blockchain Infrastructure), focuses on the development and deployment of a blockchain-based infrastructure to enable security, transparency, and interoperability across all energy services for pilot sites within EVELIXIA. The objective is to establish a robust framework that supports privacy-preserving bidirectional communication between stakeholders using EVELIXIA's blockchain-based identity, authorization, and governance mechanisms. By integrating advanced Blockchain technologies, each pilot site contributes to a cohesive and secure ecosystem for managing energy flexibility and transactions. Additionally, the framework includes innovative tokenization and origin verification services to ensure transparency and traceability of energy resources. These efforts align with EVELIXIA's overarching objectives to develop secure, efficient, and scalable systems that facilitate active stakeholder participation in energy marketplaces.

The deliverable focuses on leveraging innovative blockchain-based frameworks, cutting-edge technologies, and social engagement methodologies to achieve these goals.

VELIXIA
BUILDINGS AS ACTIVE UTILITY NODES

Funded by the European Union

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

Abbreviation	Name
ABAC	Attribute-Based Access Control
API	Application Programming Interface
Auth API	API for authentication and authorization services
BAUNs	Buildings as Active Utility Nodes
CA	Certificate Authority
CINEA	European Climate, Infrastructure and Environment Executive Agency
CMM	Community Management Microservice
D	Deliverable
DID	Decentralized Identifier
EM	Energy Marketplace
ERC721	Ethereum Request for Comments for NFTs
ETH	Ethereum cryptocurrency
EU	European Union
Fiat Transactions	Transactions conducted in traditional currencies
HLF	Hyperledger Fabric
IPFS	InterPlanetary File System
JWT	JSON Web Token
KWh	Kilowatt-hour, a unit of energy
Manage API	API for managing user and service subscriptions
NFT	Non-Fungible Token
NFT API	API for managing NFT-related transactions
OAuth 2.0	Open Authorization 2.0 protocol for authentication
OIDC	OpenID Connect protocol
REST API	Representational State Transfer Application Programming Interface
SC	Smart Contract

SSI	Self-Sovereign Identity
SSO	Single Sign-On
T	Task
VC	Verifiable Credential
WP	Work Package in a project structure

1 INTRODUCTION AND OBJECTIVES

This deliverable, D3.5 - Blockchain Smart Flexibility Contracting, consolidates the technical developments carried out in Tasks 3.4 and 3.5 of the EVELIXIA project. It focuses on integrating blockchain-based smart contracting mechanisms to support the governance and monetization of energy flexibility trading, aligned with the objectives of the project.

1.1 Scope and objectives

The scope of this deliverable is to extend and integrate existing smart-contract-based services for energy flexibility trading, governance, and value transfer through a dedicated marketplace. Key objectives include:

- Developing blockchain-enabled mechanisms for secure and transparent governance of energy flexibility transactions.
- Implementing privacy-preserving data management solutions to align with pilot requirements.
- Preparing for future integration of fiat currency mechanisms and marketplace connectivity.

1.2 Structure

This deliverable is organized as follows:

- Section 1: Provides an introduction to the deliverable's scope and objectives
- Section 2: Explores the technologies evaluated during the project, provides a comparison of blockchain with Decentralized Identities (DID), Verifiable Credentials (VC), and Self Sovereign Identity (SSI) technologies, and justifies the choice of blockchain for energy flexibility and authorization mechanisms.
- Section 3: Highlights key use cases and outlines future directions, including the integration of blockchain with the marketplace and fiat conversion mechanisms.
- Section 4: Details the developed smart contract mechanisms, including Non-Fungible Token (NFT) Smart Contracts for energy flexibility and Smart Contracts for authorization, data access, and service subscription.
- Section 5: Describes the system architecture and middleware integration, along with the prototype implementation and deployment process.

- Section 6: Concludes the deliverable by summarizing achievements and defining the next steps.
- Section 7: Lists references cited throughout the deliverable.
- Section 8: Includes annexes, such as technical specifications, API documentation, and a glossary of terms.

1.3 Relation to Other Task and Deliverables

This deliverable is directly linked to Tasks 3.4 and 3.5 and contributes to Work Package 3 (EVELIXIA's Middleware Layer and Blockchain Infrastructure). Specifically:

- Focus on the development of blockchain-based governance mechanisms and energy trading systems.
- Address the integration of middleware APIs and privacy-preserving techniques for seamless operation with the EVELIXIA platform.

2 EXPLORATION OF TECHNOLOGIES

The development of robust mechanisms for energy flexibility trading, authorization, and service subscription required a thorough evaluation of cutting-edge technologies. This section explores the key technologies considered during the project, focusing on their potential to meet the requirements of scalability, security, cost-effectiveness, maturity, and integration with existing systems. Blockchain technology, particularly Hyperledger Fabric, emerged as the most suitable solution due to its enterprise-grade features and ability to support complex transactional systems. Additionally, related technologies such as DIDs, VCs, and SSI were assessed to determine their applicability to the project's objectives. The following subsections detail the comparative analysis and provide a rationale for the selection of Hyperledger Fabric as the foundation for implementing energy flexibility trading and secure service subscription mechanisms.

2.1 Overview and comparison of Explored Technologies

In developing mechanisms for energy flexibility trading and authorization, the project team explored advanced technologies to meet the requirements of scalability, security, cost-effectiveness, and seamless integration. Key technologies evaluated included in Table 1.

Table 1. Decentralized technologies exploration

Technology	Description
Hyperledger Fabric (HLF)	Hyperledger Fabric is an open-source, permissioned blockchain framework designed for enterprise applications [1]. Its modular architecture enables features such as private channels, advanced access controls, and high scalability, making it well-suited for energy markets and transactional systems.
Decentralized Identifiers	DIDs, developed by the W3C (World Wide Web Consortium), are decentralized identifiers that provide a foundation for secure, verifiable identity management without centralized authorities [2].
Verifiable Credentials	VC, another W3C standard, are tamper-proof digital credentials that enable secure sharing of identity attributes in decentralized environments [2].
Self-Sovereign Identity	SSI frameworks leverage DID and VC to give users control over their digital identities [3]. While promising, these technologies were found less suitable for the immediate needs of energy flexibility trading due to their focus on identity management rather than transactional processes.

The evaluation criteria for technology selection included scalability, security, cost-effectiveness, maturity of technology, and integration potential. The findings are summarized in Table 2.

Table 2. Comparative Analysis of Hyperledger Fabric and SSI Technologies

Criteria	Hyperledger Fabric	DID/VC/SSI
Scalability	High transaction throughput; modular design	Limited scalability for transactional systems
Security	Advanced encryption, private channels	Secure for identity management but lacks transactional focus
Cost-effectiveness	Flexible deployment, lower operational costs	High costs due to complexity of SSI frameworks
Maturity of Technology	Established, enterprise-grade adoption	Emerging technology, limited use cases
Integration	Seamless integration with off-chain systems	Requires significant customization for transactional systems

Hyperledger Fabric was selected for its high transaction throughput, security, and seamless integration with energy markets. Its modular architecture ensures scalability and real-time processing, essential for energy flexibility trading. Fabric provides encrypted transactions, private channels, and role-based permissions, ensuring secure multi-party transactions. Unlike SSI technologies (DID, VC), which focus on identity management, Fabric supports high-frequency transactions while reducing operational costs and integrating seamlessly with existing energy systems. Its enterprise-grade adoption makes it a reliable choice for EVELIXIA.

2.2 Blockchain and Smart Contract Adoption

The decision to adopt Hyperledger Fabric was based on its alignment with the project's requirements:

- Readiness: Hyperledger Fabric has been widely adopted in enterprise environments, demonstrating proven scalability and reliability in use cases such as supply chain management and energy trading.

- Privacy and Access Control: Features such as private channels and Attribute-Based Access Control (ABAC) allow for fine-grained management of sensitive data, ensuring compliance with privacy regulations [4].
- Flexibility in Smart Contract Development: Hyperledger Fabric supports the development of customized smart contracts tailored to specific project needs.

Developed Solutions

- NFT Smart Contracts for Energy Flexibility:
  The project implemented ERC721-compliant (Ethereum Request for Comments) smart contracts on Hyperledger Fabric. These smart contracts enable:
  1. Minting of Renewable Energy Certificates: NFTs are minted to represent energy flexibility, ensuring transparent validation of renewable energy contributions.
  2. Ownership Transfer and Verification: Secure mechanisms facilitate the transfer of NFT ownership, supporting trust in energy markets.

- Smart Contracts for Authorization and Service Subscription:
  1. Smart contracts were developed to securely store critical data and implement ABAC for user permissions.
  2. These contracts enable automated service subscription mechanisms, simplifying user access to marketplace services.

- Future Integration Potential:
  The flexibility of Hyperledger Fabric supports further integration with additional systems, ensuring scalability and adaptability for evolving market needs.

The adoption of Hyperledger Fabric ensures a robust, scalable, and secure platform for energy flexibility markets. The developed NFT mechanisms and smart contracts set the foundation for reliable and transparent operations, while the modularity of Hyperledger Fabric allows for future enhancements.

3 USE CASES AND FUTURE DIRECTIONS

3.1 Energy Flexibility Certificates through NFTs

EVELIXIA Use Case

Energy flexibility certificates are tokenized as Non-Fungible Tokens (NFTs), representing 1 kWh of energy flexibility per certificate. These certificates are generated by building managers who possess the capability to adjust their energy usage during specific periods. The flexibility they generate is transferred to energy aggregators and energy providers, enabling these stakeholders to balance energy supply and demand effectively. This process is described thoroughly in Figure 1.

The lifecycle of these NFTs integrates multiple components:

- Blockchain Network: The NFTs are minted and recorded on the Hyperledger Fabric blockchain.
- InterPlanetary File System (IPFS): The actual energy certificate documents are securely stored on a private IPFS network. The hash generated from storing these documents is embedded in the NFT metadata.
- Marketplace: The NFTs are listed on the Marketplace platform as energy certificates available for purchase. Interested parties can place offers to buy these certificates, and once a transaction is finalized, ownership is transferred through the blockchain.

Stakeholders

The key stakeholders in the NFT-based energy flexibility market include:

- Building Managers: Generate energy flexibility and mint NFTs to sell their certificates.
- Energy Aggregators: Purchase energy flexibility to optimize energy distribution across their networks.
- Energy Providers: Use the purchased flexibility to meet demand fluctuations and enhance grid stability.

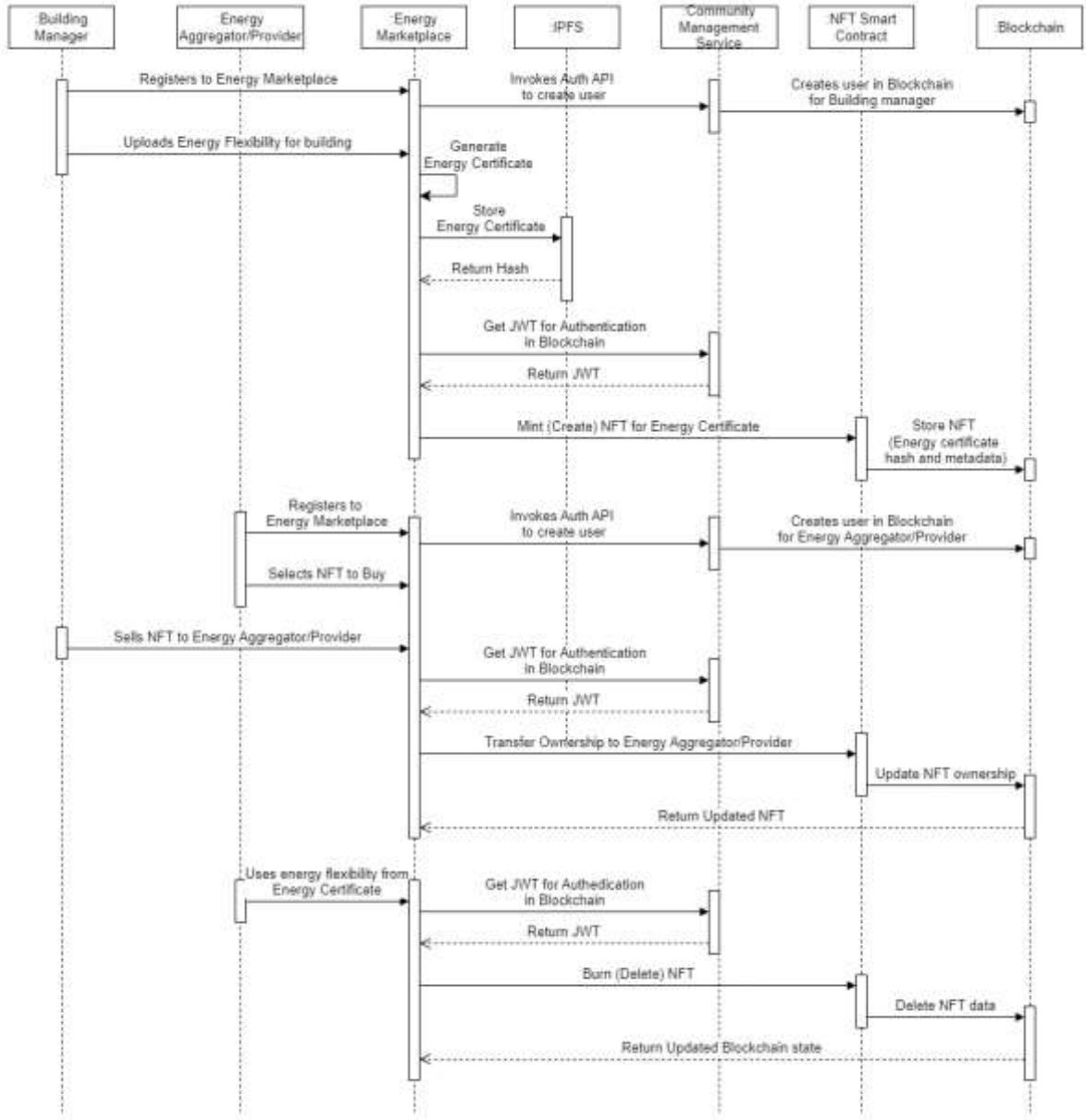

Figure 1. Energy Flexibility Certificate generation and distribution Sequence Diagram

Workflow Details

1. Minting:
   - Building managers mint NFTs through the Marketplace when energy flexibility is generated (e.g., 5 kWh generates 5 NFTs).
   - The NFT contains metadata such as energy amount, flexibility period, and the IPFS hash of the certificate document.
2. Listing:
   - The minted NFTs are listed on the Marketplace as available energy flexibility certificates.

3. Purchase Request:
    - Energy aggregators or providers submit purchase offers through the Marketplace platform.
4. Ownership Transfer:
    - Upon accepting an offer, the building manager initiates the ownership transfer using the NFT smart contract. The transaction is recorded immutably on the blockchain.
5. Validation:
    - Stakeholders can verify certificate authenticity by checking the IPFS hash stored in the NFT metadata.

Benefits for Stakeholders

- Building Managers: Gain an efficient platform to monetize energy flexibility.
- Energy Aggregators and Providers: Access verified and transferable energy flexibility assets to improve grid stability and optimize operations.
- Regulators: Benefit from a transparent system that ensures compliance and facilitates oversight.

3.2 Authorization and Service Subscription using Smart Contracts

EVELIXIA Use Case

Smart contracts simplify and automate the process of authorizing access to energy services and managing subscriptions. They add an extra layer of security and authorization control, additional to the ones implemented on EVELIXIA Marketplace. A typical scenario, presented in Figure 2, involves an end user requesting access to an energy service through the Marketplace:

1. Subscription Request: The end user submits a subscription request via the Marketplace. This request is logged on the blockchain using the Service Subscription API.
2. Service Provider Action: The service provider reviews the request and either approves or rejects it. The decision is recorded on the blockchain through the smart contract.

3. Lifecycle Management: If the subscription is updated or canceled, the changes are reflected in the blockchain by invoking the relevant smart contract function.

All interactions between the blockchain and the Marketplace occur seamlessly through the Service Subscription API. The key stakeholders in the authorization and subscription processes are the end users requesting and managing subscriptions to EVELIXIA services.

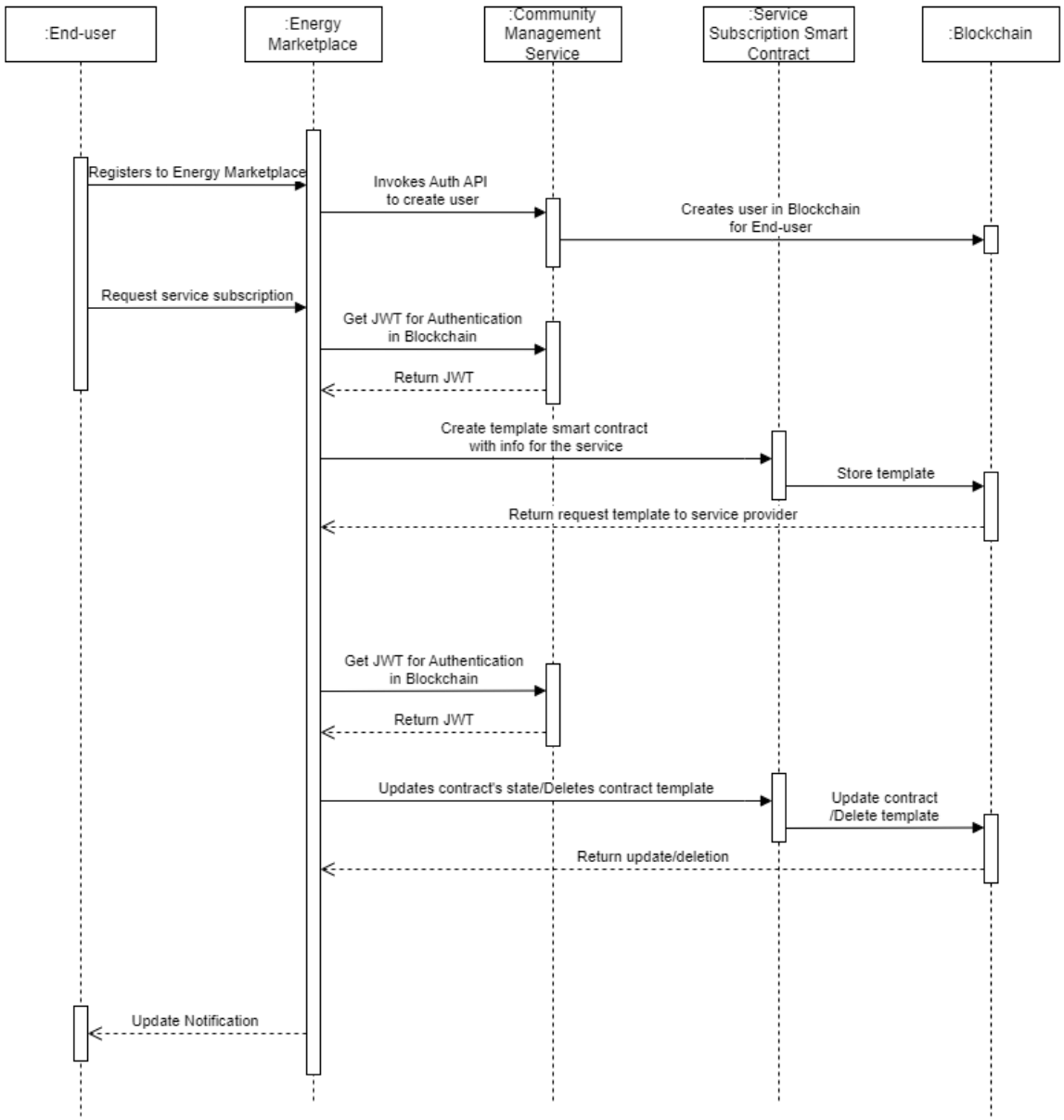

Figure 2. Service Subscription Sequence Diagram

Workflow Details

1. Request Submission:
   - The end user logs into the Marketplace and selects an energy service.
   - The subscription request is submitted and logged on the blockchain using the Service Subscription API.
2. Subscription Updates:
   - Any updates or terminations to the subscription (e.g., service modifications, cancellations) are handled via the smart contract, ensuring that all changes are logged transparently.
3. Access Management:
   - The smart contract enforces the access policies defined for the subscription. Only authorized users are granted access to the subscribed services.

Benefits

- Automation: Smart contracts eliminate the need for manual approval processes, reducing time and effort.
- Security: ABAC ensures that only authorized users can access services.
- Transparency: Blockchain provides a clear audit trail for all subscription actions, enhancing trust among stakeholders.
- Scalability: The system is capable of handling a large volume of subscription requests, making it suitable for growing markets.

3.3 Future Directions

The EVELIXIA project is poised to expand its capabilities through two key developments: the integration of the blockchain network with the Energy Marketplace and the implementation of fiat transactions for Energy Flexibility Certificates (NFTs). These advancements aim to improve the accessibility, efficiency, and usability of the system, broadening its appeal and functionality for all stakeholders.

3.3.1 Integration of Blockchain with the Marketplace

The integration of blockchain with the Energy Marketplace will be achieved through existing blockchain APIs, including the NFT and Service Subscription APIs.

This seamless connection between the Marketplace and the blockchain network ensures that critical blockchain functions are accessible and efficiently utilized within the Marketplace environment.

Planned Features and Capabilities

- Direct Interaction with Blockchain Functions:
  - The Marketplace will interact directly with blockchain operations, such as minting, transferring, and validating NFTs, and managing service subscriptions.
  - Users will execute these operations via a unified Marketplace interface without requiring direct knowledge of blockchain operations.
- Real-Time Updates of Marketplace Listings:
- Blockchain data will provide real-time updates for Marketplace listings. For example, NFT ownership transfers and subscription approvals will automatically reflect in the Marketplace, ensuring data consistency and reducing delays.

About the integration of Blockchain with the rest of the platform, these are the agreed considerations:

- It is envisioned for basically enable datasets authentication and provide control access rules to them.
- Every transaction to access the information will be stored in the blockchain infrastructure.
- Exchanged data will use JSON format (for standardization and interoperability purposes) after validating its ownership.

These operations do not require the usage of Blockchain:

- Read RAW data from pilot sites
- Store healed data
- Send commands to pilot sites assets

On the contrary, these are the expected functionalities of the IS18 Energy Services Marketplace that will use Blockchain:

- Test user registration *(when someone registers in the marketplace, needs to be also registered in the blockchain)*
    - A user is registered in marketplace
    - The same credentials are used for the user to register in the Blockchain
    - The Blockchain registration is performed internally in the backend of Marketplace
- Two different users will be used, for testing purposes:
    - Building owner (provides energy flexibility)
    - Aggregator (buys this flexibility)
    - A unique ID will be created for every user in the blockchain.

The building user will not need to register anything manually.

- Test user login: After authorizing the user with the token, the NFT is created with whichever data wanted, with no restrictions.
- List energy services available: By making any GET call to access any resource, a new JSON Web Token (JWT) will be obtained to interact with the NFT, which represents 1kW of energy flexibility for granularity purposes, containing both data and its metadata.
- Test user subscription to an energy service *(using an access control mechanism that has been implemented)*
- User publication of an energy service
    - Whenever some user registers a new service, the consumer will be registered in the blockchain.
- Test navigation through energy services

Benefits of Integration

- Streamlined Workflows: The integration eliminates redundant processes by connecting Marketplace activities directly to blockchain operations. This ensures a seamless user experience for minting, buying, and transferring NFTs, as well as managing service subscriptions.
- Enhanced Security: Blockchain's inherent immutability and transparency ensure the security of critical data, including NFT metadata, ownership records, and subscription details, reducing the risk of fraud or tampering.

3.3.2 Implementation of Fiat Transactions

The introduction of real money transactions (fiat currency) for NFTs aims to make the system accessible to a wider range of users, particularly those unfamiliar with cryptocurrency. By enabling payments in traditional currencies, the platform aligns with industry regulations and user preferences.

Motivation

- Accessibility: Fiat transaction support lowers barriers to entry for non-crypto users, allowing building owners, energy aggregators, and providers to participate without needing cryptocurrency knowledge or wallets.
- Regulatory Compliance: Aligning with local and international financial regulations ensures that the platform remains compliant and trustworthy.

Technical Implementation

Fiat transactions will be facilitated through a secure and trusted payment gateway. This integration will enable users to:

1. Pay for NFTs directly using fiat currency.
2. Generate receipts and maintain transaction histories tied to blockchain operations.
3. Reconcile fiat payments with blockchain-based ownership transfers in real time.

Stakeholders

- Building Owners: Can sell energy flexibility certificates and receive payments in fiat currency, eliminating the need for complex cryptocurrency conversions.
- Energy Aggregators and Providers: Can purchase NFTs using familiar payment methods, streamlining their operational processes.
- End Users: Benefit from a simpler payment experience, fostering trust and engagement with the platform.

Challenges

The integration of fiat transactions may introduce regulatory and technical challenges, such as:

- Payment Reconciliation: Ensuring that fiat payments are accurately reconciled with blockchain operations, such as ownership transfers.
- Operational Security: Protecting sensitive financial data from unauthorized access.
- System Scalability: Handling large transaction volumes while maintaining performance and reliability.

However, these challenges can be mitigated by using a robust payment gateway that supports secure and scalable fiat transactions.

4 SYSTEM ARCHITECTURE

4.1 Implementation and Deployment

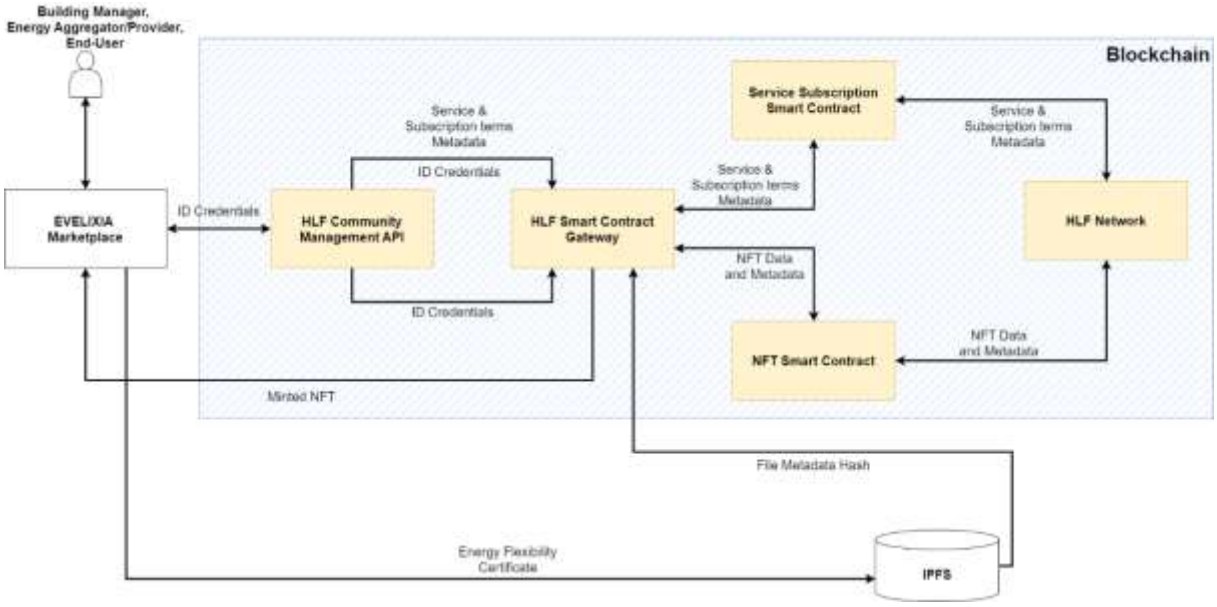

Figure 3. Holistic Blockchain framework Architecture

The Blockchain system architecture, presented in Figure 3, is designed to enable seamless and secure interactions between users, the Energy Marketplace, and the underlying Hyperledger Fabric (HLF) blockchain network. The architecture leverages key components such as the Community Management API for identity and access management, inside the blockchain network and the REST Smart Contract Gateway to expose smart contracts' functionalities via user-friendly RESTful (REST) endpoints. Users access the system through the Energy Marketplace, where they can perform operations related to energy flexibility certificates and service subscriptions.

The REST Smart Contract Gateway plays a pivotal role by bridging the Marketplace and the blockchain network, exposing key functionalities such as NFT creation, ownership transfer, and service subscription management via HTTP-based endpoints. These endpoints allow the Marketplace to submit authenticated requests for chaincode execution, making it easier for stakeholders to interact with the blockchain without requiring deep technical knowledge. The gateway retrieves HLF client credentials, stored securely within an HLF wallet, to sign and process transactions on behalf of users. This architecture enhances system scalability,

security, and user experience by abstracting blockchain complexities while maintaining high levels of data integrity and transparency.

4.1.1 System Architecture Overview

The system architecture for the blockchain-enabled platform integrates multiple cutting-edge components to ensure scalability, security, and efficiency. At its core, the system is built on Hyperledger Fabric, a permissioned blockchain framework, combined with smart contracts written in Golang. The architecture also incorporates the following components:

- Community Management Microservice (CMM): A decentralized Identity and Access Management (IAM) application that enables digital identity management and login services for users in the Hyperledger Fabric network.
- REST Smart Contract Gateway: The REST Smart Contract Gateway integrates the blockchain network with EVELIXIA Marketplace, leveraging the NFT and Service Subscription smart contracts and exposing their business logic via REST API endpoints.

4.1.2 Hyperledger Fabric Network

The Hyperledger Fabric network is deployed following industry best practices, ensuring a secure, scalable, and efficient system. The deployment details are presented in Table 3.

Table 3. Hyperledger deployment steps

Deployment practices	Description
Peers , Ordering Service and Channels	- The network consists of three peers, each representing a distinct node. These peers maintain copies of the ledger, endorse transactions, and execute smart contracts. - A secure ordering service with a Raft consensus mechanism is implemented to ensure fault-tolerance and high availability of transaction ordering. This design aligns with Hyperledger Fabric's best practices for permissioned networks. - The ordering service manages one dedicated channel, ensuring that all transactions related to energy flexibility and authorization are isolated for improved performance and confidentiality. - A private channel is set for the communication between the network members, for the purpose of conducting private and confidential transactions. The smart contracts are deployed on this channel.
Certificate Authority (CA)	- A single Hyperledger Fabric CA is deployed to issue digital certificates for user authentication and node identification. The CA enforces identity-based access policies, aligning with the Attribute-Based Access Control (ABAC) mechanisms.
Smart Contract Deployment	- Smart contracts, written in Golang, are deployed to the network using CLI peer commands. These contracts handle critical functions, including NFT creation, ownership transfers, and authorization mechanisms.
Blockchain Operations	- All blockchain operations, including peer-to-peer communication, endorsement policies, and consensus, are configured to meet the highest standards of security and reliability. The ledger data is stored securely on each peer node.

4.1.3 Integration with External Services

The blockchain network integrates with external systems to enhance functionality and usability, as detailed below:

- IPFS for Certificate Storage:
  - A private IPFS network is deployed to store energy certificates securely. Each certificate is uploaded to the IPFS network, and the resulting cryptographic hash is stored in the blockchain as part of the NFT metadata. This integration ensures that certificates are immutable and accessible while offloading large data storage from the blockchain.
  - A secure HTTPS API is exposed for interacting with the IPFS network. This API facilitates the upload and retrieval of certificates, ensuring secure communication between the blockchain network and the IPFS service.
- Energy Marketplace Platform:
  - The Energy Marketplace interacts with the blockchain via the NFT and Authorization & Service Subscription APIs. This integration allows users to mint, transfer, and validate NFTs representing energy flexibility certificates, as well as manage service subscriptions securely.

4.1.4 Deployment Process

The deployment of the system leverages containerization and automation tools to ensure consistency, scalability, and ease of maintenance. Key steps in the deployment process include:

- Containerized Environment:
  - All core components, including peers, ordering service, ledger, network, smart contracts, and the Community Management Microservice, are deployed using Docker containers. This approach ensures isolated environments for each component, simplifies scalability, and reduces deployment complexity.
- Orchestration:
  - Docker Compose is used to define and manage multi-container configurations, automating the setup of the blockchain network, smart contracts, and supporting services.
- Smart Contract Deployment:

- Smart contracts are deployed to the network using CLI peer commands, ensuring precise control over the deployment process.

4.1.5 Testing and Validation

Comprehensive testing and validation have been conducted to ensure the reliability and functionality of the system. The following methodologies were employed:

- Functional Testing for Smart Contracts:
  - Unit tests were created for each smart contract function to validate their behaviour under all expected scenarios. These tests cover core functionalities such as minting NFTs, transferring ownership, and managing access control lists.
- API Testing for Community Management Microservice (CMM):
  - The CMM APIs were tested using Postman to verify their functionality, including user registration, authentication, and access management.
- Validation Tools:
  - The testing process utilized the Fabric Test Network to simulate blockchain interactions and validate network performance under realistic conditions.

4.2 Community Management Microservice

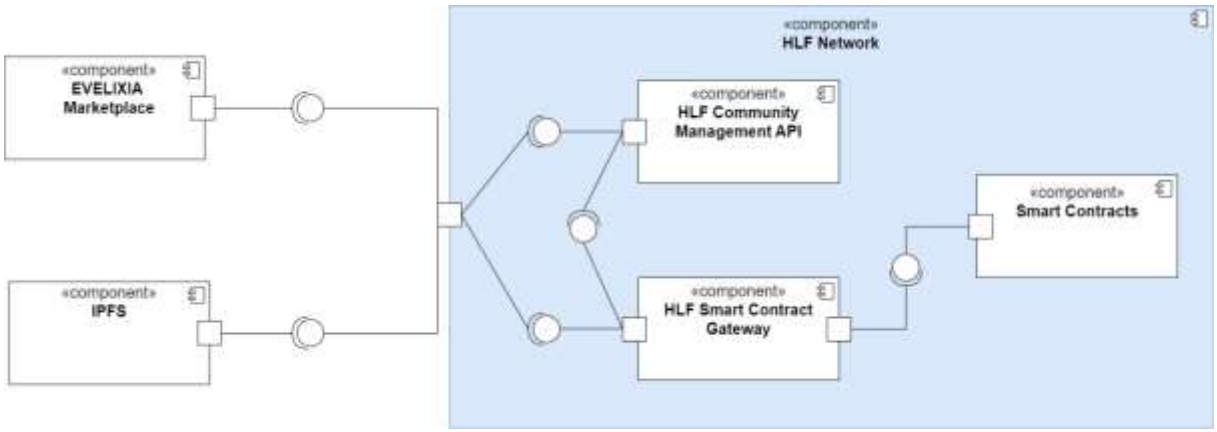

Figure 4. Blockchain system Component diagram

The Community Management Microservice (CMM), presented in Figure 4 ,is an Identity and Access Management system that allows other blockchain services to manage their digital identities and login services in a Hyperledger Fabric Network.

Briefly, CMM is an OAuth 2.0 and OpenID Connect Provider that interacts with Hyperledger Fabric Certificate Authorities and supports Single Sign On (SSO) operations for any blockchain service in the network. The CMM facilitates user registration, credential generation and storage, inside the HLF wallet, using the HLF Certificate Authority (CA) service, ensuring robust HLF identity verification and management.

CMM provides the following functionalities:

- Single-Sign On and Single-Sign Out for blockchain services.
- User Registration to the network.
- Password Manager for secure credential storage.
- Hyperledger Fabric X.509 Certificate support.
- OAuth 2.0 protocol support.
- OpenID Connect protocol support.

Additionally, CMM offers REST APIs with a set of HTTP endpoints. These APIs enable developers and administrators of each blockchain service to easily interact with CMM in a secure, fast, and user-friendly way.

The APIs exposed by the CMM are:

1. The Auth API: Contains endpoints to handle all SSO, OAuth 2.0, and OpenID Connect-related features.
2. The Manage API: Contains endpoints that provide developers and administrators of each blockchain service the ability to manage users and applications.

Thus, blockchain services use these APIs to interact with the CMM, allowing them to manage and identify their users effectively.

4.2.1 Auth API

The Auth API exposes the OAuth 2.0 and OpenID Connect-related endpoints of the CMM and performs all identity-related features. In the context of OAuth 2.0 and OpenID Connect, tokens allow applications (namely clients) to identify users and access protected resources on behalf of themselves or the identified users. The tokens used in this ecosystem include:

- ID Token: Identifies users.

- Access Token: Allows access to protected resources.
- Refresh Token: Retrieves new ID and Access Tokens when they have expired.

The Auth API:

- Acts as an identity and access management provider, enabling the management of users for blockchain services and offering SSO (e.g., login and logout procedures) based on OAuth 2.0 and OpenID Connect.
- Can operate solely as an OAuth 2.0 and OpenID Connect Provider, allowing blockchain services to request an access token for a specific resource without requiring a user-authenticated session.

Using this API, each blockchain service can obtain the appropriate tokens to enable functionalities such as:

- User login and logout using ID Tokens.
- Accessing other blockchain services on behalf of themselves or their users.

Blockchain services log in and log out users by using the appropriate ID Tokens. Additionally, blockchain services must interact with the Auth API to obtain Access Tokens to access other resources or blockchain services. These Access Tokens, issued by the CMM, contain all the necessary information for authorized access to each resource. Once the Access Token is obtained, it must be included in the headers of all HTTP requests.

When a blockchain service receives a request with an Access Token from another blockchain service, it can verify the contents of the token using the Auth API endpoints. This ensures secure and authenticated communication between services.

The technical specification of the Auth API is provided in Annex 8.2.1.

4.2.2 Manage API

The Manage API exposes endpoints to handle user management-related features. Using this API, administrators of various applications and blockchain services can manage their users effectively. Users register through the Manage API endpoints by creating a registration request, which is subsequently accepted or rejected by administrators.

The technical specification of this API is provided in Annex 8.2.2.

4.3 REST Smart Contract Gateway

The REST Smart Contract Gateway, as shown in Figure 5, is a critical microservice that facilitates communication between the Energy Marketplace and the Hyperledger Fabric blockchain network. This gateway abstracts the complexities of blockchain interactions by exposing Hyperledger Fabric's smart contract functionalities via RESTful APIs, allowing external applications to seamlessly access blockchain data and perform transactions without directly interacting with the blockchain network. Once registered, users can initiate blockchain transactions via the Marketplace, which interacts with the HLF Smart Contract Gateway to securely invoke smart contract functions.

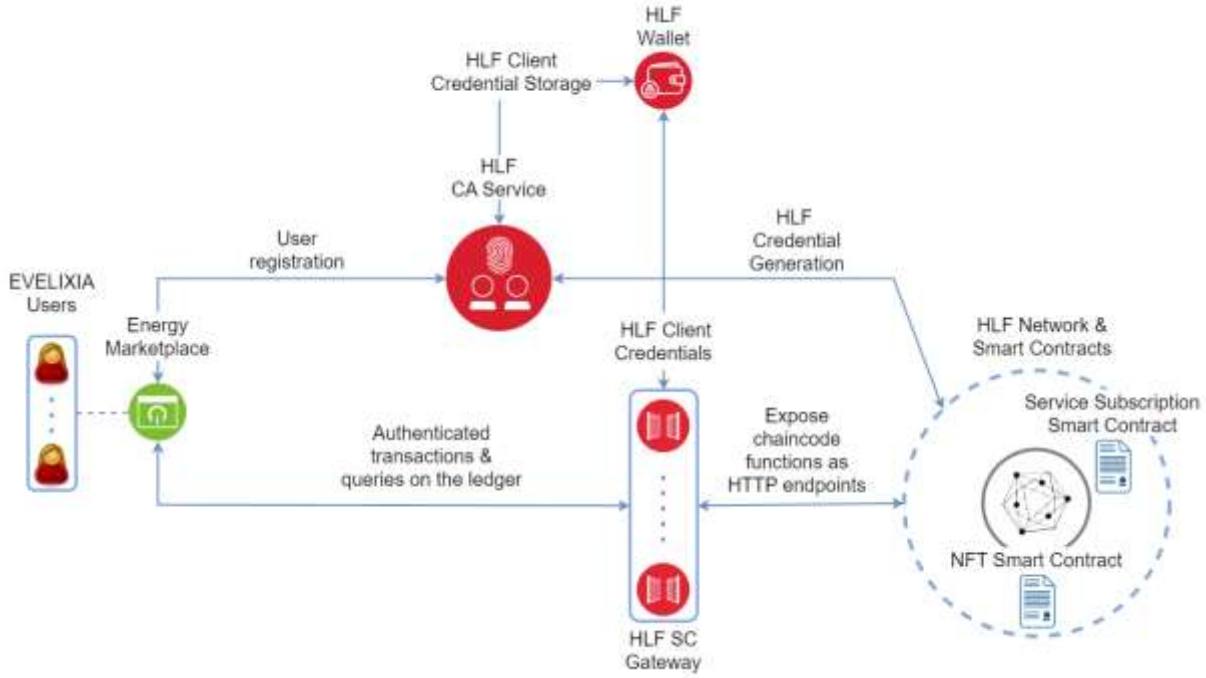

Figure 5. EVELIXIA Hyperledger Fabric (HLF) Smart Contract Gateway Architecture

Key Features and Architecture

The REST Smart Contract Gateway provides a range of functionalities that enable efficient and secure interaction with the blockchain. Its architecture is composed of the following core components:

- API Interface Layer
  - The gateway offers a RESTful API interface that maps directly to smart contract functions deployed on Hyperledger Fabric.
  - Supports standard HTTP methods such as:

    - POST for invoking blockchain transactions (e.g., NFT minting, EVELIXIA Service subscription management in Blockchain).
    - GET for querying ledger data.
    - DELETE for removing records where applicable.
  - The API is designed to follow RESTful principles, ensuring scalability and ease of use for developers.
- Hyperledger Fabric Client SDK Integration
  - Built using the Hyperledger Fabric Node.js SDK, the gateway interacts with the blockchain network by:
    - Submitting transactions to the appropriate peer nodes.
    - Querying the blockchain ledger.
    - Handling user identity through enrollment and credential management.
  - The SDK enables secure communication with the Fabric network by signing transactions with user-specific certificates.
- User Authentication and Authorization

  - The gateway enforces authentication using JWT, which store user identity information.
  - Authorization checks are implemented by verifying the invoking user's ID against the blockchain's certificate authority (CA).
  - User permissions are managed based on Attribute-Based Access Control (ABAC), ensuring granular access to smart contract functions.
- Chaincode Execution
  - Invoke chaincode functions (e.g., minting NFTs, transferring ownership).
  - Query chaincode state for real-time insights.
  - Handle endorsement policies and transaction approvals based on Hyperledger Fabric network configurations.

Deployment and Configuration

The REST Smart Contract Gateway is deployed as a containerized service using Docker, ensuring easy deployment, scaling, and management. Key configuration aspects include:

- Environment Variables:
  - Blockchain network parameters (e.g., peer addresses, channel name).
  - API security keys for authentication.
  - IPFS integration settings for certificate storage.
- Security Measures:
  - HTTPS is enforced for all communications between the gateway and external services.
  - Request payload validation and logging for audit purposes.

The technical specification of this API is provided in Annex 8.1.

5 SMART CONTRACT MECHANISMS

5.1 NFT Smart Contracts for Energy Flexibility

5.1.1 Design and Implementation

The NFT smart contracts developed in this project play a central role in representing Energy Flexibility Certificates (EFCs). Each NFT corresponds to 1 kWh of energy flexibility tied to a specific building, making them transferable and tradeable digital assets within the energy market. For instance, a building manager with 5 kWh of flexibility holds five separate NFTs, each encoded with critical energy-related metadata.

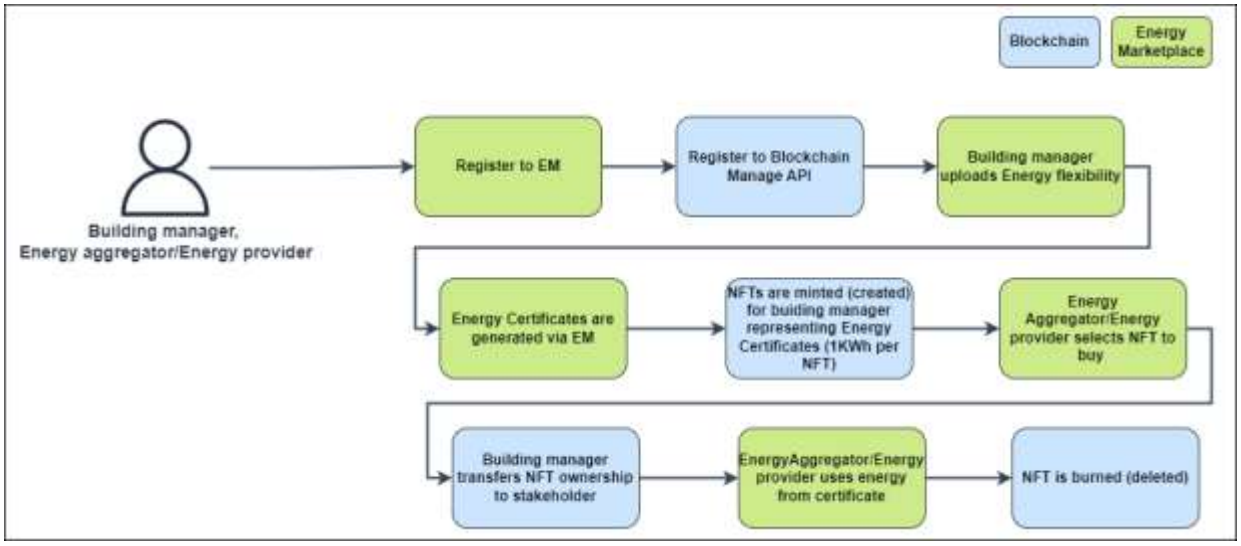

Figure 6. Energy Flexibility distribution workflow (NFT)

ERC721-Compliant NFTs

The ERC721 standard was chosen to ensure the NFTs comply with established global standards for non-fungible tokens. This compliance guarantees compatibility with existing tools and frameworks while maintaining the necessary security and integrity for the assets. Hyperledger Fabric [5], a permissioned blockchain designed for enterprise-grade applications, required custom implementation of ERC721-compliant smart contracts in Golang, aligning the flexibility of Ethereum standards with the privacy and scalability of Hyperledger Fabric.

Minting Workflow

The minting process begins with the building manager, who initiates the creation of NFTs through the Marketplace platform, developed in another task. The workflow includes:

1. Submission of energy-related details (e.g., energy amount, flexibility period) and metadata about the energy source.
2. The Marketplace validates the information and interacts with the smart contract to mint NFTs.
3. The resulting NFTs are stored securely on the Hyperledger Fabric ledger, ensuring immutability and transparency.

During the minting process, energy-related data (e.g., energy amount, flexibility period) and metadata about the energy source are uploaded to IPFS (InterPlanetary File System) [6]. The hash returned by IPFS is then embedded in the NFT smart contract, ensuring that the certificate data is stored in a decentralized and immutable way. This approach enhances transparency and prevents tampering of critical energy information.

Ownership Transfer

NFT ownership is transferred through a secure, multi-step process:

1. An energy aggregator or provider submits a purchase request through the Marketplace.
2. The building manager reviews the request and decides whether to accept or reject it.
3. If accepted, the building manager invokes the transfer function of the NFT smart contract, specifying the token ID and the recipient's ID. The transaction is recorded on the blockchain, ensuring traceability and trust.

5.1.2 Energy Flexibility

These NFTs tokenize energy flexibility, showed in Figure 6, enabling trading and verification of renewable energy contributions. Each NFT contains verifiable information about the energy source, flexibility period, and issuing entity, simplifying market operations and increasing transparency.

Data Stored on NFTs

The following critical information is embedded within each NFT:

- Energy amount: The kWh associated with the flexibility.
- Flexibility period: The time interval during which the flexibility is available.
- Energy source: Metadata about the origin of the energy.
- Issuer details: Information about the building manager or energy generator.

Validation and Verification

Validation occurs during the minting process, where the Marketplace ensures the data's accuracy before embedding it in the NFT. Users may supplement additional fields, such as descriptions, to provide context. However, the critical fields are immutable and verified by the system.

5.1.3 Security and Privacy

Ensuring Privacy and Security

Several security measures are implemented to ensure the integrity and privacy of NFT transactions:

- HTTPS Protocol: All API requests are transmitted over HTTPS, encrypting the data in transit.
- JWT Authentication: API endpoints require JWT tokens containing the NFT owner's unique ID for authentication and verification.

Fraud Prevention

The system prevents unauthorized access and tampering through:

- x509 Certificates: Each client's identity is encoded in their x509 certificate, ensuring only verified entities can interact with the blockchain network.
- Identity Validation: Before invoking smart contracts, the system verifies the client ID against the blockchain ledger to confirm authenticity.

5.2 Smart Contract for Authorization and Service Subscription

The smart contract for authorization and service subscription are designed to:

- Manage user access to energy data securely and efficiently.

- Enable service subscriptions between parties, ensuring all access and subscription policies are adhered to.

This smart contracts store critical information about shared services and enforce access policies. It leverages the same REST API-based interface used for NFTs, exposing their functions as endpoints. Authorization is implemented using JWT tokens, where each token embeds the invoking party's ID. The ID is compared with the one stored in the x509 certificate of the Hyperledger Fabric (HLF) client. If the IDs match, the user is granted access to invoke the endpoint, ensuring secure and controlled interactions.

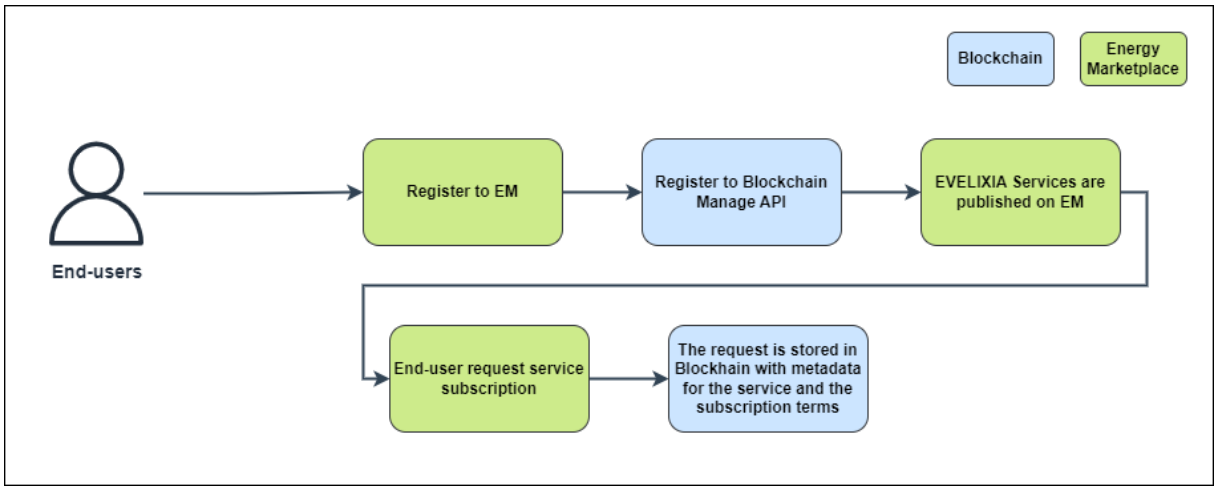

Figure 7. Authorization and Service Subscription Workflow

The smart contracts implement Attribute-Based Access Control (ABAC) to ensure granular access permissions based on user identities. Key features include:

- Attribute Storage: Each user's ID is embedded in their x509 certificate within the Hyperledger Fabric network. The HLF SDK extracts this ID for comparison during authorization.
- Dynamic Attributes: The attributes (e.g., user identities) are dynamic and can change over time to reflect updates in roles, permissions, or policies.
- Access Validation: During every interaction, the smart contract validates the invoking party's identity by comparing the ID in the JWT token with the x509 certificate ID. If the IDs do not match, access is denied.

5.2.1 Service Subscription Mechanisms

The smart contracts facilitate seamless service subscription processes between parties within the Energy Marketplace (EM), as illustrated in Figure 7. The subscription mechanism includes:

1. Request Submission: The party interested in an EVELIXIA service submits a request to the service provider through the EM.
2. Blockchain Logging: The action is recorded on the blockchain, ensuring transparency and traceability.
3. Policy Enforcement: Access policies for the subscribed service are updated and enforced dynamically.

The smart contract manages subscription data, including user ID, service details, subscription status, and policies, ensuring all interactions are governed and stored securely on the blockchain. It, also, follows the same security and privacy policies as the NFT Smart contract.

6 CONCLUSION

This deliverable has successfully outlined the development, deployment, and initial implementation of blockchain-based solutions for energy flexibility trading and service management within the EVELIXIA project. Focusing on smart contract mechanisms, energy flexibility certificates, and seamless integrations, the work completed demonstrates significant progress toward achieving the objectives outlined in Task 3.4 and Task 3.5.

The primary aim of this deliverable was to design and implement blockchain-enabled mechanisms that enhance governance, monetization, and trading of energy flexibility. By leveraging Hyperledger Fabric, the deliverable introduces innovative solutions, including:

- NFT Smart Contracts for energy flexibility certificates, ensuring transparency and trust in trading.
- Smart Contracts for Authorization and Service Subscription, automating and securing access management and subscription workflows.
- Integration with IPFS, providing decentralized and immutable storage for energy certificates.

These components form the foundation for a scalable, efficient, and user-friendly energy flexibility marketplace.

In addition, several significant milestones have been achieved:

- NFT Smart Contracts: The implementation of ERC721-compliant NFTs enables the tokenization of energy flexibility certificates. These NFTs, representing 1 kWh of energy flexibility each, are securely stored on the blockchain with associated certificate data managed in IPFS.
- Authorization and Service Subscription: Smart contracts automate user access and subscription management, streamlining workflows while ensuring secure data handling through Attribute-Based Access Control (ABAC).
- Integration of Key Components: The integration of Hyperledger Fabric, IPFS, and the Energy Marketplace enables seamless interactions through APIs, allowing stakeholders to mint, trade, and manage energy flexibility certificates.

- Robust Testing: Functional testing of smart contracts and API validation using industry-standard tools ensures the reliability and security of the implemented solutions.

The deliverable directly benefits key stakeholders:

- Building Owners: Gain a secure and transparent platform to monetize energy flexibility.
- Energy Aggregators and Providers: Access verifiable energy flexibility certificates to optimize grid operations and meet demand.
- End Users: Experience simplified workflows for accessing energy services, supported by automated subscription management.

By ensuring transparency, efficiency, and security, the developed solutions lay the groundwork for a more inclusive and accessible energy flexibility market.

Looking forward, two critical enhancements will shape the next phase of development:

- Integration of Blockchain with the Marketplace: This will streamline workflows and enhance real-time interactions between Marketplace features and blockchain functions, improving the user experience and operational efficiency.
- Implementation of Fiat Transactions: Adding support for fiat payments will broaden the platform's accessibility, enabling non-crypto users to participate in energy flexibility trading and ensuring compliance with financial regulations.

These developments will further enhance the platform's usability and adoption, fostering a robust and scalable ecosystem for energy flexibility management.

This deliverable demonstrates the EVELIXIA project's commitment to leveraging blockchain technology to address the evolving needs of the energy flexibility market. By developing innovative mechanisms for trading, governance, and service management, the project is creating a transparent, secure, and efficient marketplace for all stakeholders. With a clear roadmap for future enhancements, the platform is well-positioned to drive the adoption of decentralized energy solutions, contributing to a more sustainable and resilient energy system.

7 REFERENCES

1. Androulaki, E., et al. (2018). "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains." Proceedings of the 13th EuroSys Conference.
2. W3C (2021). "Decentralized Identifiers (DIDs) v1.0: Core Architecture, Data Model, and Representations." Retrieved from https://www.w3.org/TR/did-core/.
3. Allen, C., et al. (2016). "The Path to Self-Sovereign Identity." Coindesk.
4. Xu, X., et al. (2019). Architecture for Blockchain Applications. Springer.
5. Hyperledger Fabric Documentation (2022). "Developing Applications and Smart Contracts." Retrieved from https://hyperledger-fabric.readthedocs.io/.
6. Protocol Labs (2022). "InterPlanetary File System (IPFS)." Official Website. Retrieved from https://ipfs.io.

8 ANNEXES

8.1 Technical Specifications of Smart Contracts

8.1.1 NFT Smart Contract

Main Functionalities

The NFT smart contracts are designed with a range of functionalities to manage the lifecycle of energy flexibility certificates. These functionalities are exposed as REST API endpoints, simplifying integration with external systems. The key endpoints are presented in Table 4:

Table 4. NFT Smart Contract API

Endpoint	Purpose	Input	Output
POST http/nft/mint	Mints, or creates, a new NFT that is assigned to a specific owner.	JSON-encoded Mint payload	n/a
POST http/nft/transfer	Transfer the ownership of a non-fungible token to another user.	tokenID: String, ownerID: String	n/a
POST http/nft/create-proof	Create a proof-of-ownership of an NFT. The response contains a verification URL that can be communicated to third parties for verification.	tokenID: String	Base64URL-encoded string
DELETE http/nft/burn	Burns, or destroys the specified NFT. This function is restricted to NFT owners.	tokenID: String	n/a

GET http/nft/nfts	This function queries the contract's state for the concrete NFT objects that are owned by the invoker.	tokenID: String	JSON-encoded data object (depending on what was specified during the mint process)
GET http/nft/nfts/{id}	Retrieve the information associated with an NFT from the ledger.	tokenID: String	ContractArray: JSON-encoded array of State Objects
GET http/nft/verify-proof/{proof}	Verify the validity of a NFT's proof-of-ownership.	Base64URL-encoded string	n/a

These endpoints, built using the Hyperledger Fabric SDK for Golang, abstract blockchain complexities and allow seamless interaction between applications and the blockchain ledger.

8.1.2 Authorization and Service Subscription Smart Contract

The smart contracts include several key functions, exposed as REST API endpoints, to manage access and subscriptions. The primary functionalities are listed in Table 5:

Table 5. Service Subscription API

Endpoint	Purpose	Input	Output
POST http/acl/insert	Associates the input (key, value) in the smart contract's state.	key: An arbitrary string. value: An arbitrary byte array.	n/a
GET http/acl/get(key)	On input a string key, this function queries the state of the contract to retrieve the value (byte array) that	An arbitrary string that must exist in the contract's state.	The byte array associated with the key in the contract's state, assuming such an entry exists,

	is associated with it, assuming it exists.		or an error otherwise.
GET http/acl/exists	On input a string key, this function queries the state of the contract to infer if there is a state entry associated with this key.	An arbitrary string.	A string-encoded boolean, which reflects whether a state entry under the given key exists (true) or not (false), or an error otherwise.
DELETE http/acl/delete	Removes from the state of the smart contract the association between the (input) key and the value stored.	An arbitrary string that must exist in the contract's state.	n/a
GET http/acl/keys	This function can be used to retrieve all the keys that are maintained in the state of the smart contract.	n/a	A JSON array of all the keys that are stored in the contract's state (which may be empty if no state entries exist).
GET http/acl/values	This function can be used to retrieve all the values that are maintained in the state of the smart contract.	n/a	A JSON array of all the values that are stored in the contract's state (which may be empty if no state entries exist).
PATCH http/acl/update	The purpose of this function is to allow updates on a	key: An arbitrary string that must exist in	n/a

	preexisting key-value association in the smart contract's state.	the contract's state. value: An arbitrary byte array.	

These endpoints enable external systems, such as the EM, to interact seamlessly with the blockchain. The EM acts as the primary interface for all subscription-related activities.

8.2 API Documentation

8.2.1 Auth API

Table 6 presents the Auth API's endpoints which manage user authentication and authorization through the implementation of OAuth 2.0 and OpenID Connect (OIDC) protocols. It ensures that the blockchain services can securely handle login/logout operations, issue tokens, and manage access to protected resources.

Key Functionalities:

- Authentication Support:
  - Supports both Bearer Auth (using OAuth 2.0 Access Tokens) and Basic Auth (using Client ID and Client Secret).
  - Ensures that only authorized entities can access blockchain services.
- Token Management:
  - Issues ID tokens to identify users.
  - Issues Access tokens for resource access.
  - Issues Refresh tokens for renewing expired ID or Access tokens.
- SSO Support:
  - Facilitates Single Sign-On (SSO) for seamless interaction across various services.

Table 6. Auth API Endpoints

Endpoint	Method	Description
/auth/realms/evelixia/.well-known/openid-configuration	**GET**	Retrieves OAuth2.0/OIDC provider metadata.
/auth/realms/evelixia/protocol/openid-connect/token	**POST**	Issues tokens (Access, ID, Refresh) based on request type.
/auth/realms/evelixia/protocol/openid-connect/logout	**POST**	Logs out users and invalidates their tokens.
/auth/realms/evelixia/protocol/openid-connect/userinfo	**GET**	Fetches user information tied to an access token.

Implementation Details:

- Requests must include headers like:
  - Content-Type: application/x-www-form-urlencoded
  - Accept: application/json
- A robust validation mechanism verifies token authenticity using introspection endpoints

Example Use Case:

When a service needs to authenticate a user, it sends a request to /auth/realms/fever/protocol/openid-connect/token with the user's credentials. The API responds with an access token, which the service includes in subsequent requests to validate user identity.

8.2.2 Manage API

Table 7 lists the endpoints of the Manage API which facilitate user management operations, enabling administrators of blockchain services to register, approve,

retrieve, and delete user data. It ensures effective and secure management of digital identities.

Key Functionalities

- User Registration:
  - Allows new users to submit a registration request through the /users endpoint.
- User Approval:
  - Administrators can accept or reject registration requests using the PATCH /users/:email endpoint.
- User Data Retrieval:
  - Enables fetching all registered and pending users via the GET /users endpoint.
- User Deletion:
  - Administrators can delete user accounts using the DELETE /users/:email endpoint, ensuring user lifecycle management.

Table 7. Manage API Endpoints

Endpoint	Method	Description
/users	**POST**	Submits a new user registration request.
/users/:email	**PATCH**	Approves or rejects a registration request.
/users	**GET**	Retrieves a list of all users (registered and pending).
/users/:email	**DELETE**	Deletes a user account based on the provided email.

Data Handling

- JSON Format: All data is exchanged in JSON format to ensure compatibility and standardization.
- Authorization: All endpoints require a valid access token in the request headers to ensure secure interactions.

Example Workflow

- Registration:
  - A new user submits their registration details (e.g., username, email, and custom attributes) via the /users endpoint.
- Approval:
  - The administrator reviews and approves the request through the /users/:email endpoint.
- Retrieval:
  - Administrators query the /users endpoint to fetch a list of registered and pending users.
- Deletion:
  - If a user account is no longer needed or requires removal for security reasons, the administrator deletes it through the /users/:email endpoint. The operation ensures that all associated data is securely removed from the system.

Security Measures

- Integration with Auth API: The Manage API relies on the Auth API for authentication and token validation.
- Strict Authorization: All administrative operations, such as user deletion, require elevated privileges to prevent unauthorized access.