



Equity crowdfunding tool for Community-driven Investments v2

AUTHORS:

C. Patsonakis, G. Giannopoulos, D. Tzilopoulos, I. Gripiotis, A. Tsita, P. Tzallas, I. Moschos, D. Ioannidis [CERTH]



H2020-LC-SC3-2018-2019-2020 / H2020-LC-SC3-2020-EC-ES-SCC

EUROPEAN COMMISSION

Innovation and Networks Executive Agency

Grant agreement no. 957810

PROJECT CONTRACTUAL DETAILS

<i>Project title</i>	IntegrAted SolutioNs for the DecarbOnization and Smartification of Islands
<i>Project acronym</i>	IANOS
<i>Grant agreement no.</i>	957810
<i>Project start date</i>	01-10-2020
<i>Project end date</i>	30-09-2024
<i>Duration</i>	48 months
<i>Project Coordinator</i>	João Gonçalo Maciel (EDP) - JoaoGoncalo.Maciel@edp.com

DOCUMENT DETAILS

<i>Deliverable no.</i>	D3.4
<i>Dissemination level</i>	Public
<i>Work package</i>	WP3 - Transition and Investment Decision Support Framework
<i>Task</i>	T3.2 Equity crowdfunding tool for Community-driven Investments
<i>Due date</i>	31-04-2023
<i>Actual submission date</i>	31-04-2023
<i>Lead beneficiary</i>	CERTH

Version History

v	Date	Beneficiary	Changes
0.1	10/02/2023	CERTH	First Draft
0.2	11/03/2023	CERTH	Update Chapter 3: Related Work
0.3	13/03/2023	CERTH	Update Chapter 4: Platform analysis and Chapter 5: Technical study

0.4	24/03/2023	CERTH	Add Chapter 6: Platform's Architecture and Chapter 8: Smart Contracts
0.5	30/03/2023	CERTH	Add Chapter 7: Equity Crowdfunding Cloud Services and ANEX I, ANEX II, ANEX III
0.6	10/04/2023	CERTH	Update Chapter 9: Platform's structure and Chapter 2: Introduction
0.7	12/04/2023	CERTH	Update Chapter 1: Document Information and Chapter 10: Conclusions
1.0	20/04/2023	CERTH	Document finalization after quality review

This publication reflects the author's view only and the European Commission is not responsible for any use that may be made of the information it contains.

Table of contents

1	Document Information	9
1.1	Executive summary	9
1.2	List of acronyms and abbreviations	10
2	Introduction	12
2.1	Scope and objectives of the deliverable	12
2.2	Structure of the deliverable	12
2.3	Relation to other tasks and deliverables	13
2.4	Equity crowdfunding tool for Community-driven Investments second version changelog	13
3	Related work	15
3.1	Concept Introduction	15
3.2	Definition of crowdfunding and crowdfunding models	16
3.3	Motivation of crowdfunding in renewable energy projects	17
3.4	Existing European crowdfunding platforms for renewable energy projects	18
3.4.1	Peer-to-peer crowdfunding platforms	18
3.4.2	Equity based platforms	19
4	Platform analysis	21
4.1	General Description	21
4.2	Platform's features	21
4.2.1	Project management features	21
4.2.2	User management features	21
4.2.3	Investment management features	22
4.2.4	Performance monitoring features	22
4.3	Identification of actors	22
4.4	Platform requirements	25
4.4.1	Functional Requirements	25
4.4.2	Non-Functional Requirements	26
5	Technical study	27
5.1	Programing languages	27
5.1.1	Front-End	27
5.1.2	Back-End	27
5.2	Frameworks	27
5.2.1	Front-End	27



5.2.2	Back-End.....	28
6	Platform's architecture	29
6.1	Introduction.....	29
6.2	Notarizing Smart Contract Operations	33
6.3	Finite State Machines.....	36
6.3.1	Projects.....	37
6.3.2	Investments	40
6.4	Platform Core Flow	41
6.4.1	Project Creation.....	41
6.4.2	Investment.....	42
6.4.3	Funding Failed.....	46
6.4.4	Project Funded.....	50
6.4.5	Investment Tokenization.....	53
6.4.6	Equity Tokenization	54
7	Equity Crowdfunding Cloud Services.....	55
7.1	Identity & Access Management (IAM)	56
7.1.1	Investor Registration.....	58
7.1.2	Project Developer Registration	59
7.2	Payment Gateway	61
7.2.1	Minting Fungible Tokens	63
7.2.2	Burning Fungible Tokens	66
8	Smart Contracts	69
9	IANOS Equity Crowdfunding platform.....	71
9.1	Platform's structure.....	71
9.2	Login & Registration page.....	72
9.3	Account page	73
9.4	Project management page.....	74
9.5	Investment opportunities page	75
9.6	Investment page.....	76
10	Conclusions.....	78
	Annex I: Equity Crowdfunding Cloud Services Reference APIs	79
	Identity & Access Management (IAM) Reference API.....	79
	FundChain Backend Reference API.....	83
	Payment Gateway Reference API	95

Annex II: Smart Contracts Specifications.....	99
Fungible Token Management	99
Fungible Token Notarization.....	106
Project & Investment Handling	107
Project & Investment Notarization	112
Investment & Equity Tokenization	114
Investment Notarization.....	120
Equity Notarization	120
Annex III: Blockchain Cloud Services.....	122
Fungible Token Service	122
Project & Investment Handling Service	127
Investment Tokenization Service	132
Equity Tokenization Service	135
References	139

List of Figures

Figure 6.1: High-level overview of the blockchain-based equity crowdfunding framework.	30
Figure 6.2: Sequence diagram that depicts how the entities involved in the notary interaction pattern interact with each other.	35
Figure 6.3: Finite state machine diagram illustrating the state transitions that a crowdfunding project goes through, with various conditions and actions associated with each state.	38
Figure 6.4: Finite state machine diagram illustrating the state transitions that an investment goes through, with various conditions and actions associated with each state.	40
Figure 6.5: Sequence diagram illustrating the actors and the steps involved in creating a new crowdfunding project.	42
Figure 6.6: Sequence diagram illustrating the actors and the steps involved in the investment intent submission phase.	43
Figure 6.7: Sequence diagram illustrating the actors and the steps involved in the investment settlement phase.	44
Figure 6.8: Sequence diagram illustrating the actors and the steps involved in investing in a crowdfunding project via stable coin payments.	46
Figure 6.9: Sequence diagram illustrating the actors and the steps involved in the investment reimbursement submission phase.	47
Figure 6.10: Sequence diagram illustrating the actors and the steps involved in the investment reimbursement settlement phase.	48
Figure 6.11: Sequence diagram illustrating the actors and the steps involved in reimbursing stable coin investments of a crowdfunding project.	49
Figure 6.12: Sequence diagram illustrating the actors and the steps involved in the investment allocation submission phase.	50
Figure 6.13: Sequence diagram illustrating the actors and the steps involved in the investment allocation settlement phase.	51
Figure 6.14: Sequence diagram illustrating the actors and the steps involved in transferring stable coin investment funds to the project developer's stable coin account.	52
Figure 6.15: Sequence diagram illustrating the actors and the steps involved in the tokenization of investments.	53
Figure 6.16: Sequence diagram illustrating the actors and the steps involved in the tokenization of equity shares.	54

Figure 7.1: Architecture diagram illustrating the collection of services that compose the platform's Identity & Access Management (IAM) service.....	57
Figure 7.2: Sequence diagram illustrating the registration flow for an investor on the blockchain-based equity crowdfunding platform.....	59
Figure 7.3: Sequence diagram of project developer's registration request submission process to the platform.....	60
Figure 7.4: Sequence diagram illustrating the activation of a project developer's user account on the platform.....	61
Figure 7.5: Payment flow in an online store; a graphical representation of the steps involved in handling payments via a payment gateway.....	62
Figure 7.6: Sequence diagram illustrating the actors and the steps involved in the mint request submission phase.	65
Figure 7.7: Sequence diagram illustrating the actors and the steps involved in the mint request settlement phase.	66
Figure 7.8: Sequence diagram illustrating the actors and the steps involved in the burn request submission phase.	67
Figure 7.9: Sequence diagram illustrating the actors and the steps involved in the burn request settlement phase.	68
Figure 9.1: Platform's structure.....	71
Figure 9.2: IANOS FundChain platform's sidebar menus for: (a) Administrators, (b) Developers, (c) Investors, (d) Visitors.....	71
Figure 9.3: Platform's Registration Page for Investors.....	73
Figure 9.4: Platform's Registration Page for Developers.....	73
Figure 9.5: Platform's Login Page.....	73
Figure 9.6: Platform's Account Page	74
Figure 9.7: Platform's Project Management Page	75
Figure 9.8: Platform's Investment Opportunities Page.....	76
Figure 9.9: Platform's Investment Page.....	77

List of Tables

Table 1.1: List of acronyms and abbreviations.....	10
Table 4.1: Platform's actors - Investor	22
Table 4.2: Platform's actors - Project Developer	23
Table 4.3: Platform's actors - Platform Administrator	24
Table 4.4: Platform's actors - Visitor	25
Table 6.1: Mapping between the names of project states and their corresponding string encoding	39
Table 6.2: Mapping between the names of project states and their corresponding string encoding	41

1 Document Information

1.1 Executive summary

This document is part of the deliverables of the IANOS project, in the context of Work Package 3 (WP3) “Transition and Investment Decision Support Framework”. Specifically, it presents the second and last version of IANOS Equity Crowdfunding platform, the result of the effort for Task 3.2 “Equity crowdfunding tool for Community-driven Investments”. The platform will provide all stakeholders, namely project investors and islanders, with the opportunity to fund future initiatives in exchange for shares in the project or to register their future projects for funding through the FundChain Platform. The component will convert community members into shareholders in renewable energy assets, in accordance with IANOS' community and Islander-centered philosophy.

Initially, the introductory sections provide the necessary information about the current document. More specifically, the scope and the objectives of the deliverable are presented and, subsequently, the structure of the current deliverable alongside its relation to other tasks and deliverables.

Before the presentation of IANOS Equity Crowdfunding platform, a valuable initial review of the related works is held. Firstly, the focus is on defining and describing the crowdfunding model and the different types of crowdfunding that are available. In a similar way, the motivation behind crowdfunding initiatives for renewable energy projects is analysed. The existing crowdfunding platforms for renewable energy projects located in Europe are also presented.

The platform analysis section presents a detailed analysis of the platform's general description, features, actors, and user scenarios. The platform includes common features found in the most popular crowdfunding initiatives, containing project management features, user management features, investment management features, performance monitoring features and legal information.

The identification of the actors of the platform is a vital step in the development of the platform. Together with the latter, the initial user scenarios are also identified in this document. Lastly, the requirements concerning the cybersecurity of the platform, concerning user authentication, private data handling, and security of communications, are defined.

Furthermore, an initial technical study is held concerning the software pre-requirements of the platform, namely the programming languages and the frameworks utilized for the development of the platform's front-end and back-end.

The document also delves into the platform's architecture, including notarizing smart contract operations, finite state machines for projects and investments, and the platform's core flow for project creation, investment, funding status, and tokenization.

The equity crowdfunding cloud services section covers identity and access management, investor and project developer registration, and payment gateway for minting and burning fungible tokens.

Additionally, Hyperledger Fabric's chaincode-based smart contracts are presented, which enable secure and transparent equity crowdfunding on a blockchain-based platform, handling the entire project lifecycle and facilitating fungible and non-fungible token functionality.

The document introduces the IANOS FundChain platform, discussing its structure, login and registration page, account page, project management page, and investment opportunities page.

Finally, conclusions of the deliverable are presented, where the future steps that are required for the final development of the platform are identified and described in the final chapter of the deliverable.

1.2 List of acronyms and abbreviations

Table 1.1: List of acronyms and abbreviations

Abbreviations	Full Description
API	Application Programming Interface
CSS	Cascading Style Sheets
dApp	Decentralized Application
DLT	Distributed-Ledger based energy Transactions
ESB	Enterprise Service Bus
FAQ	Frequently Asked Questions
HTML	Hypertext Markup Language
IRR	Internal Rate of Return
MTV	Model-Template-View
p2p	Peer-to-peer
PV	Photovoltaic
SME	Small and Medium-sized Enterprises
UML	Unified Modeling Language
VSE	Very Small Enterprises
W3C	World Wide Web Consortium
AML	Anti Money Laundering
API	Application Programming Interface
CSS	Cascading Style Sheets
dApp	Decentralized Application
DLT	Distributed-Ledger based energy Transactions

ESB	Enterprise Service Bus
FAQ	Frequently Asked Questions
FSM	Finite State Machine
FT	Fungible Token
HLF	Hyperledger Fabric
HTML	Hypertext Markup Language
IAM	Identity & Access Management
IRR	Internal Rate of Return
KYC	Know Your Customer
MTV	Model-Template-View
NFT	Non-Fungible Token
p2p	Peer-to-peer
PV	Photovoltaic
SME	Small and Medium-sized Enterprises
SSO	Single-Sign-On
UC	Use Case
UI	User Interface
UML	Unified Modeling Language
UX	User Experience
VPP	Virtual Power Plant
VSE	Very Small Enterprises
W3C	World Wide Web Consortium

2 Introduction

2.1 Scope and objectives of the deliverable

Deliverable 3.4 is the second version of the Equity crowdfunding tool for Community-driven Investments. The document provides a detailed description of the FundChain platform developed within the IANOS project. All stakeholders, such as project investors and islanders, will be able to fund future projects in return for shares in the project or register their future projects to obtain funding through the FundChain Platform. The component will turn community members into shareholders in renewable energy assets, consistent with IANOS' community and Islander-centric strategy.

In this document, we introduce, in a progressive manner of complexity and technical depth, the architecture of an end-to-end secure, auditable, verifiable, transparent and decentralized framework for equity crowdfunding that incorporates all main actors, or stakeholders, of such platforms, i.e., project developers, investors, platform administrators, as well as casual, pass by visitors. The proposed framework digitizes the entire lifecycle of crowdfunding projects, starting from their draft phase and all the way to their (real-world) completion. Whilst investors are able to fund projects via traditional payment schemes based on FIAT currencies and prominent online payment platforms (e.g., PayPal), the proposed framework puts forth a novel payment gateway component that is built on top of blockchain-based fungible token (FT) schemes that are employed to deliver a stable coin construction, in which digital tokens that exist in the blockchain correspond to real-world FIAT currencies. Moreover, the framework harnesses blockchain-based, non-fungible token (NFT) technologies to issue verifiable, digital certificates to investors regarding, first, their various investments and, second, their equity in completed projects. In a few words, these certificates serve as verifiable, tamper-evident and trustworthy means that allow investors to prove their investments and project equities, respectively, to other real-world actors. These certificates are of independent, not only research, but also practical interest, as they provide a solid base layer for additional opportunities and business scenarios, such as (blockchain-based) equity trading marketplaces, tax deductions for green investments and others.

2.2 Structure of the deliverable

The deliverable is structured as follows

- **Chapter 1** presents information about the current document, namely the executive summary of the deliverable and a list of acronyms and abbreviations.
- **Chapter 2** presents the scope, objectives and structure of the deliverable, as well as its relation to the other tasks and deliverables of the IANOS project.
- **Chapter 3** provides a comprehensive introduction of the platform as well as the initial review of the related works. More specifically, the definition of crowdfunding and the crowdfunding models are presented together with the motivation for

12



crowdfunding in renewable energy projects. Finally, the existing crowdfunding platforms for renewable energy investments are presented.

- **Chapter 4** provides an analysis of IANOS FundChain platform. The features of the platform are presented, together with the actors and the functional and non-functional requirements of the platform.
- **Chapter 5** is the technical study of the platform, where the frameworks and the programming languages used in the development of the platform are presented.
- **Chapter 6** provides a comprehensive analysis of the platform's architecture, which includes key aspects such as notarizing smart contract operations, finite state machines for managing projects and investments, and the core flow for project creation, investment, funding status, and tokenization. Additionally, the equity crowdfunding cloud services section covers critical functionalities such as identity and access management, investor and project developer registration, and a payment gateway for minting and burning fungible tokens.
- **Chapter 7** presents the equity crowdfunding cloud services section covers critical functionalities such as identity and access management, investor and project developer registration, and a payment gateway for minting and burning fungible tokens.
- **Chapter 8** describes the smart contracts that the platform utilizes
- **Chapter 9** provides an overview of the IANOS FundChain platform, including its structure and key pages such as login and registration, account management, project management, and investment opportunities.
- **Chapter 10** provides conclusions about the overall work implemented in the current deliverable.

2.3 Relation to other tasks and deliverables

The deliverable D3.4 is based on the Use Cases and requirements described in D2.3 “Report on Islands requirements engineering and UCs definitions”. T2.5 “System Architecture”, which describes the system's architecture, also includes an early description of the functionality of the component detailed in this deliverable, as well as its linkages and dependencies. The deliverable is also related to “iVPP P2P transactive energy framework” described in D4.9.

2.4 Equity crowdfunding tool for Community-driven Investments second version changelog

This deliverable constitutes the second and final version of Equity crowdfunding tool for Community-driven Investments. The changes from the first version deliverable (D3.3) are presented in this section. Minor changes denote the changes to grammar or syntax error

or the substitution of words without any changes in meaning. The D3.4 changes in the respective chapters are as follows:

- Chapter 1: Revision to reflect the content of D3.4.
- Chapter 2: Minor changes.
- Chapter 3:
 - Section 3.1: Addition of the section to better introduce the concept of the platform within the IANOS project and the blockchain-based crowdfunding tool for renewable energy projects.
 - Section 3.2: Minor changes.
 - Section 3.3: Minor changes.
 - Section 3.4: Minor changes.
- Chapter 4:
 - Section 4.1: Minor changes.
 - Section 4.2: Minor changes.
 - Section 4.3: Detailed description of the actors of the platform that better reflect their roles and requirements.
 - Section 4.4: New section providing a detailed description of the functional and non-functional requirements of the platform.
- Chapter 5: Inclusion of all the technologies that the platform utilizes
- Chapter 6: New chapter that describes in detail the architecture of the platform.
- Chapter 7: New chapter that describes in detail the equity crowdfunding cloud services.
- Chapter 8: New chapter that describes in detail the platform's smart contracts.
- Chapter 9: A detailed up-to-date presentation of the most basic pages of the platform (Chapter 6 in the previous version).
- Chapter 10: Revision to reflect the conclusions of D3.4.
- Annex I: New chapter introducing equity crowdfunding cloud services reference APIs.
- Annex II: New chapter introducing smart contracts specifications.
- Annex III: New chapter introducing blockchain cloud services.

3 Related work

3.1 Concept Introduction

Equity crowdfunding is a relatively new concept in the world of investment, where instead of the traditional model of investment where only accredited investors or venture capital firms can invest in a company, anyone with an Internet connection can invest in a startup or small business and receive equity in return. Equity crowdfunding platforms act as intermediaries, connecting startups in need of funding with a wider pool of investors, who can contribute small amounts of money in exchange for a stake in the company. The concept of equity crowdfunding has been growing in popularity in recent years as a way for startups to access alternative sources of funding and for everyday investors to have the opportunity to invest in the companies they believe in and potentially receive financial returns.

Focusing on the domain of renewable energy projects, a significant increase in the number of crowdfunding platforms has been observed since the beginning of the 21st century. Peer-to-peer and equity models constitute the basis of the majority of these platforms, which involve initiatives created within the borders of the nation in which they are created. Given that the majority of nations have various legal frameworks for crowdfunding, this is to be expected. Undoubtedly, all of these platforms put forth a new paradigm in the pursuit of achieving a green, clean and sustainable future.

While all these centralized equity crowdfunding platforms have helped to democratize the investment process, they also have their own set of limitations and drawbacks, which stem from their inherent centralized architecture. One major issue is the lack of transparency and trust, as all transactions are managed by a central authority, who can potentially manipulate the data or engage in fraudulent activities. This can lead to a lack of confidence in the investment process, both for startups and investors.

Another downside is the high costs associated with centralized equity crowdfunding platforms, as they require intermediaries to manage the investment process, which increases the overall cost of investment. This can limit the amount of funding that startups, or developers, receive, and reduce the potential returns for investors.

Furthermore, centralized equity crowdfunding platforms can also limit accessibility, as they are often subject to strict regulations and restrictions, making it difficult for startups, developers and investors from certain countries or regions to participate. This can limit the potential for developers to reach a wider pool of investors, and for investors to access a diverse range of investment opportunities.

Overall, the limitations of centralized equity crowdfunding platforms highlight the need for a more secure, transparent, and accessible investment process, which is where blockchain-based equity crowdfunding platforms come in.

The use of blockchain technology in equity crowdfunding brings several benefits to the table. Firstly, it provides a secure and transparent way of tracking investments and the

distribution of equity, as all transactions are recorded on a public ledger that is tamper-proof and resistant to fraud. This helps to build trust between startups, investors, and regulators, as all parties have access to a transparent and verifiable history of transactions.

Blockchain-based equity crowdfunding platforms also provide more accessibility to investment opportunities, as they remove the traditional barriers to entry in the investment world such as geographical location and financial status. Startups can reach a wider pool of potential investors, and investors can easily invest in startups they believe in, without having to go through complex and time-consuming processes.

In addition, blockchain technology enables a more efficient and cost-effective way of managing investments and distributing dividends, as all transactions can be automated through smart contracts. This eliminates the need for intermediaries, reducing the costs associated with the investment process and allowing for a more streamlined experience for all parties involved.

Overall, the use of blockchain technology in equity crowdfunding offers a more secure, transparent, accessible, and efficient investment process, providing opportunities for startups to receive funding from a wider pool of investors, and for investors to own a piece of the companies they believe in.

In this work, we present a comprehensive, end-to-end architecture for a secure, transparent, auditable, verifiable, and decentralized framework for equity crowdfunding. This framework incorporates all of the key stakeholders involved in equity crowdfunding platforms, including project developers, investors, platform administrators, as well as casual visitors.

The framework digitizes the entire lifecycle of crowdfunding projects, from the draft phase through to their physical completion, providing a seamless and secure experience for all parties involved. While investors have the option to fund projects through traditional payment methods, such as FIAT currencies and online payment platforms (e.g., PayPal), the proposed framework introduces a novel payment gateway that utilizes blockchain-based fungible token (FT) technology to provide a stable coin construction. This allows for the creation of digital tokens that are directly tied to real-world FIAT currencies, providing a secure and transparent means of payment for equity crowdfunding.

Additionally, the framework utilizes blockchain-based non-fungible token (NFT) technology to issue verifiable, digital certificates to investors, which serve as proof of their investments and equity in completed projects. These certificates are tamper-evident and trustworthy, providing a solid foundation for additional business opportunities, such as equity trading marketplaces and tax deductions for green investments, whilst providing a seamless and secure experience for all parties involved.

3.2 Definition of crowdfunding and crowdfunding models

Crowdfunding is an alternative way of financing, which is based on the financial contribution of the public to support a new or existing business idea. It offers benefits to

both parties involved, as a project developer can receive financing for their project, while an investor receives certain benefits, depending on the model. Crowdfunding is possible thanks to specialized online platforms that provide the necessary transaction security and the necessary tools to manage the campaign. Each crowdfunding platform's user can contribute by giving from just one euro to several hundred, depending on the funding packages offered by the creator of the campaign.

There are mainly five crowdfunding models, depending on the gains that each investor has. Those models are:

- **Donation-based crowdfunding**, where contributors gain nothing, and it mainly focusses on charitable projects.
- **Reward-based crowdfunding**, where, in return for their funding, contributors are rewarded with a token. They receive no interest in the earnings, or shares.
- **Pre-purchase crowdfunding**, which is similar to the reward model, but the contributor, is provided with the final product that they finance, instead of any other token. In renewable energy projects, the investor's reward for investment comes in the form of electricity supplied or a discount on electricity rates.
- **Peer-to-peer (p2p) crowdfunding**, which is similar to acquiring a loan from a bank, meaning that contributors expect return of the capital they invested, either with an interest bearing, or alternatively not. It is also known as debt crowdfunding or lending crowdfunding. It is one of the most common models.
- **Equity crowdfunding**, where the investor is offered shares of the projects, they finance.

Some of the most popular crowdfunding platforms for several kinds of project ideas are Kickstarter [1], Indiegogo [2], Patreon [3], GoFundMe [4], Chuffed [5], ArtistShare [6], and more.

3.3 Motivation of crowdfunding in renewable energy projects

Crowdfunding models have been very beneficial in the development of renewable energy projects. They can be addressed not only to traditional investors that are motivated mainly by the prospect of future financial gains, but also to individuals with charitable goals motivated by the prospect of reducing emissions by developing sustainable energy solutions. Thus, the participation in a crowdfunding platform for renewable energy projects can bring financial reward, social return, as well as material gain in the form of energy, in pre-purchase crowdfunding model. Most crowdfunding platforms already have a large and dedicated investor community, as well as high traffic. Their users are people who are actively looking for projects to financially support and invest in. This fact not only makes the marketing of each crowdfunding campaign easier, but also constitutes the best return on investment.

As far as project development goes, crowdfunding gives a rare opportunity to acquire funds for their ambitious projects, without an intermediate, not including the potential platform fees. Additionally, such a platform provides the project developers with the opportunity to build a close and dedicated relationship with the project supporters. The

public that has invested in it financially and emotionally will follow it faithfully. Moreover, through crowdfunding platforms, they can communicate with a wide audience and collect data and feedback, which may be extremely useful in the future. In IANOS and similar projects, the equity crowdfunding can boost the sense of community that the islanders have, enabling the factorization of the renewable energy assets. Even without the social features that the FundChain platform incorporates, the common ownership of the future assets can psychologically enhance the collaboration and communication of the community members.

3.4 Existing European crowdfunding platforms for renewable energy projects

Since the end of the first decade of 2000s, there has been a great growth in crowdfunding platforms for renewable energy projects, based in Europe. Most of the platforms are based on the peer-to-peer and equity models. Most of the platforms involve projects developed within the boundaries of the country in which they are developed. This is expected, as most of the countries have different legal regulations for crowdfunding.

3.4.1 Peer-to-peer crowdfunding platforms

Bettervest [7] was founded in 2012 in Germany as an online platform that gives actors the opportunity to invest in renewable energy projects across the globe. It is based on the p2p crowdfunding model, where project owners borrow money from investors in exchange of interest. There is a plethora of renewable energy projects available for investment varying from renewable energy farms (e.g PV and wind farms) to public transport. The minimum investment is set to €250 and each project that is financed using the platform mechanics must meet certain criteria, such as minimum loan requirement of €100,000, reduction of harmful emissions, payback period of one to ten years, 10% of the loan volume as equity capital and additional collateral. There is no platform fee or payment involved in the transactions. The platform provides secure login and registration via username and password, data protection, legal information via term and condition contacts, analytics about the project financing development and FAQs section for user assistance.

GreenVesting [8] is another debt-based crowdfunding platform based in Germany since 2012. It is presented as a competent service partner for green project development, energy efficient real estate and general financing of energy projects. Together with financing, it is responsible for developing and operating various PV systems on behalf of different investors and project providers. The minimum investment is set to €100. Although the only language available is German, it provides helpful guidance for each step of the investment process, from selecting a project to financing it and latter receive payment. Each available project is equipped with an overview of all important information about the owner, the development, the purpose etc. Additionally, all the required legal information and risk are provided.

Abundance [9] was founded in 2012 in United Kingdom as an easy way to access investments for sustainable energy projects. It is based on the debt-based crowdfunding. It provides actors with opportunities to invest in different kinds of projects with a minimum investment of £5. More specifically, each actor can invest in a green energy company directly choosing the risk that they want to take or invest in council led projects across the UK in return of a lower risk, or even buy existing investments from other customers on the platform's marketplace. The platform provides a plethora of information about the investment procedures and the available projects. Moreover, the platform assists in the assessment of the risks in each project. It provides secure login and registration via username and password, data protection via disclaimers and term and condition contracts and a blog providing news and other articles.

3.4.2 Equity based platforms

Windcentrale [10] is one of the oldest and largest crowdfunding platforms, founded in the Netherlands in 2010. It is based on the equity crowdfunding model. More specifically, it offers the opportunity to become owner of wind turbines by splitting them in wind shares available for purchasing with a simple mouse click. Each wind share has an expected annual capacity of 500kWh, and it costs €210 for 18 years. Additional costs include the statutory energy tax, the storage of sustainable energy, as well as a little less than €2 per month for maintenance and management. The power produced by the wind shares usually is deducted from the annual electricity bill and it usually corresponds to the 90% of the required electricity for each actor. When more electricity is recurred, a regular electricity tariff must be paid. A blog and a FAQ section are provided by the platform to help the investment process. Moreover, the platform offers an app that allows the user to monitor the performance of the windmills that they invested in.

Econeurs [11] is a German equity crowdfunding platform on the market since 2013. It aims to support climate and environmental protection by investing in renewable energies and in their efficient use. The platform does not require a registration or agency fee and the minimum investment is set to €250. The investors are offered projects varying from solar and wind farms to organic food products. Each project registered must provide an overview of the project's information and the company involved. The platform provides secure login and registration via username and password, information for the investors, the companies, a blog with various news articles, and articles about green initiatives.

Lendopolis [12] is a French based equity crowdfunding platform founded in 2014. It allows projects in renewable energy sector developed by very small enterprises (VSEs) and small and medium-sized enterprises (SMEs) to acquire funds from individuals. In Lendopolis, an actor invests in a project of their choice via an intermediary company, created especially for this occasion, grouping together all the investors within a single structure. The intermediary company, managed by Lendopolis, owns 40% of the renewable energy projects, while the developer owns the remaining 60%. In the end of investment, the developer is required to buy back the shares that the intermediary company owns, becoming the only owner of the asset, while intermediary company pays the investors a

price for their shares, fixed in advance. This price is set to ensure a profit based on the original investment. The platform offers detailed information about the investment process, together with a guide to understand crowdfunding.

1miljoenwatt [13] is a foundation based in the Netherlands. It launched a crowdfunding platform in 2013 in order to fund solar energy projects, PV panels installed in roofs that supply electricity to football stadiums in the city of Groningen. The minimum investment is set to €550 and corresponds to one individual solar panel including maintenance. Each investor receives money each year, depending on the electricity that their panels produced. During the course of the project, 1miljoenwatt is involved as owner or manager of the project, taking care of administration and maintenance. The platform provides an online 'MijnStroom' portal in order to inform the participants about the development of their investment.

Citizenergy [14] is a crowdfunding platform founded in 2012 based in Portugal. Citizenergy offers every model of crowdfunding available, although most of the projects' funding are based on the equity and the lending model. It is the first platform that allows cross-border investment in renewable energy projects. Citizenergy achieves the latter by bringing different cooperatives and crowdfunding platforms together on an international level. Citizenergy is a form of intermediate between any investor and a plethora of crowdfunding platforms, mainly in Europe.

4 Platform analysis

4.1 General Description

IANOS FundChain platform is a unique equity crowdfunding platform developed within IANOS project. The platform is designed to fund and thus promote the use of renewable energy related assets in the two pilot islands of the project, Terceira in Portugal and Ameland in the Netherlands. The platform would eventually allow shared ownership of renewable energy infrastructures, as well as financially support islanders' initiatives. The final goal of crowdfunding platform is to promote the communication and collaboration between the islanders.

In the crowdfunding platform each user can have two roles, either creator or investor. By creating a new entry, a user will be able to set a fundraising goal to support his/her project, while, by investing in a project a user will acquire equities of the investment. The investment can be held either via normal FIAT currency or by specifically defined energy tokens.

4.2 Platform's features

IANOS FundChain platform will acquire the majority of features that the most popular equity crowdfunding platforms have, with the addition of projects specific features. These features can be categorized depending on their functionality and utility.

4.2.1 Project management features

The projects management features include several features concerning the projects that are in need of funding. Each project must include a clear Title, a detailed Description, and a Category based on the renewable energy asset that is going to be developed. Moreover, in terms of the investment procedure, an Investment Target, Minimum/Maximum investment fee, the available Shares and current Number of Investors will be included. Additional information about the project will include the Project's prospectus, a Time Overview about the project's development stage and information about the project owner. These features will provide the potential investor with all the needed information about their investment.

4.2.2 User management features

In order to use the services of the platforms, each user will be given the opportunity to make a profile in the platform. To make said profile Secure Username/Password Registration and Secure Username/Password login is needed, as well as User verification every time they log in. The user will have the ability to edit their profile by adding personal and professional information, a profile picture and interests. Additionally, each profile will be equipped with secure decentralized dApp dashboards (wallets) that will enable the access to the user's energy tokens.

4.2.3 Investment management features

Another very important aspect of a crowdfunding platform is the management of project's funding, and the actual payments. IANOS FundChain platform will be equipped with the most common features for payment management, which include Escrow accounts, where funds are held in trust while the parties involved complete a transaction, Subscription payments, which are automatic payments based on a schedule, while different payment APIs are going to be provided. Additionally, the users will be able to choose the currency of the transaction, varying from real FIAT current to tokenized energy based on the Distributed-Ledger based energy Transactions (DLT). Platform developers' main priority will be the security of the transactions using the latest blockchain technologies.

4.2.4 Performance monitoring features

IANOS FundChain platform will be equipped with visual analytics in order to assist the users to monitor their investments. More specifically, summary reports are going to be available on a scheduled basis, accompanied by statistical analysis of the investment and development report. Internal rate of return (IRR) analysis is going to be provided for each project as an additional advice to the investors. A key goal of the platform is the energy sufficiency of the islands, using renewable energy assets, thus, it is important to present each future investment's energy yield report.

4.3 Identification of actors

Equity crowdfunding platforms bring together various actors such as investors, project developers, and platform administrators, each playing a crucial role in the success of a project. Investors provide the funding necessary for project development, while project developers present their ideas and utilize the funds to bring their projects to life. Platform administrators act as intermediaries, managing the platform and ensuring compliance with legal and regulatory requirements. To ensure a successful outcome for all involved, it is important to understand the functional and non-functional requirements of each actor in the equity crowdfunding process, which we analyze in tabular format in the remainder of this section.

Table 4.1: Platform's actors - Investor

Actor	Investor
Description	Investors play a crucial role in equity crowdfunding platforms by providing funding for projects or businesses in exchange for ownership equity. They evaluate investment opportunities, make decisions to invest, and become part owners of the project or company. The goal for investors is to see a return on their investment through the growth and success of the funded entity.
Functional Requirements	<ul style="list-style-type: none"> • User-friendly interface for browsing and investing in projects • Secure payment gateway for transactions

	<ul style="list-style-type: none"> • Clear and transparent information on projects and their financial status • Easy access to investment portfolio management • Regular updates on project progress and financial returns • Integration with popular payment methods • Option to communicate with project managers and other investors • Ability to withdraw investments and receive returns • Access to detailed financial reports and risk assessments • Compliance with local and international financial regulations
Non-functional Requirements	<ul style="list-style-type: none"> • High availability and reliability of the platform • Fast and responsive user interface • Strong security measures to protect sensitive information and financial transactions • Compliance with data privacy regulations • Scalability to accommodate growth in the number of projects and investors • Support for mobile devices and desktop browsers • Integration with financial tools and services • Ease of use for people with different levels of investment knowledge • Accessibility for people with disabilities • Good customer support and efficient dispute resolution mechanisms.

Table 4.2: Platform's actors - Project Developer

Actor	Project Developer
Description	Project developers on equity crowdfunding platforms are responsible for presenting their projects and securing funding from a large number of investors through the platform. They are responsible for overseeing the implementation and execution of the project and ensuring that the investment from the crowd is used to achieve the desired outcome.
Functional Requirements	<ul style="list-style-type: none"> • Project Presentation: The platform must allow the project developers to present their project in an engaging and detailed manner to potential investors. • Investment Tracking: The platform should provide project developers with the ability to track investment progress, including the number of investments received and the amount of funding raised. • Communication Tools: The platform must provide project developers with tools to communicate with investors, such as a messaging system or forum. • Financial Management: The platform should provide project developers with tools to manage their finances, such as tracking expenses, generating financial reports, and setting up bank transfers.

	<ul style="list-style-type: none"> Project Update: The platform must provide a mechanism for project developers to update investors on the progress of their project.
Non-functional Requirements	<ul style="list-style-type: none"> User Experience: The platform must have a user-friendly interface that is easy to navigate and use. Data Security: The platform must provide secure storage of sensitive information, such as financial data and personal information. Scalability: The platform must be able to accommodate an increasing number of projects and investors over time. Reliability: The platform must be reliable and available to users at all times, with minimal downtime. Support: The platform must provide comprehensive support to project developers, including technical support and assistance with using the platform.

Table 4.3: Platform's actors - Platform Administrator

Actor	Platform Administrator
Description	The platform administrator is responsible for managing and maintaining the equity crowdfunding platform. They ensure that the platform operates smoothly, securely, and transparently for all parties involved. The administrator is responsible for handling the technical aspects of the platform, verifying project proposals, and overseeing the distribution of funds. They also monitor the compliance of all parties with the terms and conditions of the platform, as well as provide support and assistance to project developers and investors.
Functional Requirements	<ul style="list-style-type: none"> User Management: The platform administrator should be able to manage user accounts and access levels, approve or reject new user registrations, and assign appropriate permissions to users. Project Management: The platform administrator should be able to manage and monitor the entire project life cycle, from creation to completion. They should also be able to review project submissions, approve or reject projects. Payment Processing: The platform administrator should be able to process payments from investors and disburse funds to project developers, manage escrow accounts, and ensure compliance with financial regulations. Data Management: The platform administrator should be able to manage and store large amounts of data securely, including user data, project data, and financial data. They should also be able to provide reports and analytics on platform activity. Customer Support: The platform administrator should be able to provide support to investors and project developers, respond to inquiries and resolve any issues.
Non-functional Requirements	<ul style="list-style-type: none"> Security: The platform should be secure and protected against unauthorized access, hacking, and data breaches. Scalability: The platform should be scalable to accommodate growth in user numbers and projects.

- **Performance:** The platform should be fast, responsive, and reliable, with minimal downtime and technical issues.
- **User Experience:** The platform should be user-friendly, intuitive, and easy to navigate for all users.
- **Compliance:** The platform should comply with all legal and regulatory requirements, including data protection laws and financial regulations.

Table 4.4: Platform's actors - Visitor

Actor	Visitor
Description	This role encompasses all casual, pass by visitors that are browsing the content of the platform and have, generally, very limited access to the platform's features, i.e., restricted to reading what is considered as publicly available content.
Functional Requirements	<ul style="list-style-type: none"> • User-friendly interface for browsing projects
Non-functional Requirements	<ul style="list-style-type: none"> • High availability and reliability of the platform • Fast and responsive user interface • Support for mobile devices and desktop browsers • Accessibility for people with disabilities

4.4 Platform requirements

Equity crowdfunding is a revolutionary method of funding small and medium-sized enterprises (SMEs) that allows individuals to invest in start-ups, projects and other businesses they believe in. With the advent of blockchain technology, equity crowdfunding has been further streamlined and made more secure. A blockchain-based equity crowdfunding platform not only offers a decentralized and transparent system for investment, but also enables the creation of a shared ownership structure, where each investor is recognized as a stakeholder. In this section, we will outline the functional and non-functional requirements that a blockchain-based equity crowdfunding platform must meet in order to provide a seamless and secure investment experience for its stakeholders.

4.4.1 Functional Requirements

In the following, we outline a concise list of the platform's functional requirements:

- **Decentralized platform:** The platform must be built on a decentralized blockchain network to ensure transparency, security and immutability of data.
- **Smart Contract functionality:** The platform should be able to deploy and execute smart contracts for transactions, funding, and distribution of equity.
- **Asset Tokenization:** The platform must be able to tokenize the assets being crowdfunded and enable the secure transfer of ownership.

- **KYC/AML Compliance:** The platform must have a mechanism to verify the identity of investors and comply with Know-Your-Customer (KYC) and Anti-Money Laundering (AML) regulations.
- **User Management:** The platform must have a user management system that allows project developers to access the platform and manage their projects.
- **Investment tracking:** The platform must provide real-time tracking of investments, including contributions, transfers, returns and various kinds of certificates.
- **Secure Wallet:** The platform must have a secure wallet for each investor to store their assets and manage their investments.
- **Multi-currency Support:** The platform must support multiple currencies to make it accessible to a wide range of investors.

4.4.2 Non-Functional Requirements

In the following, we outline a concise list of the platform's non-functional requirements:

- **Scalability:** The platform must be able to handle a high volume of transactions and user traffic.
- **High Availability:** The platform must have a high availability rate and must be accessible to users at all times.
- **Security:** The platform must have robust security measures to prevent data breaches and unauthorized access to user information.
- **User Experience:** The platform must have a user-friendly interface that is easy to navigate and understand.
- **Performance:** The platform must have fast transaction processing times and low latency to ensure a seamless user experience.
- **Interoperability:** The platform must be compatible with other blockchain networks and ecosystems to enable cross-border investment opportunities.
- **Compliance:** The platform must adhere to all relevant legal and regulatory requirements for equity crowdfunding.

Now that we have documented all the functional and non-functional requirements for the blockchain-based equity crowdfunding platform, we can proceed with the introduction of the platform's architecture. The requirements documented thus far provide us with a solid foundation for designing the architecture, which aims to accommodate the needs of all the actors involved in the platform.

5 Technical study

In this section, information is going to be provided regarding the technologies that are utilized by the developers of IANOS FundChain platform, in order to efficiently facilitate its requested functionalities and features.

5.1 Programing languages

5.1.1 Front-End

For the development of the front-end of the platform, HTML and CSS are utilized.

HTML [15], or Hypertext Markup Language, is the standard markup language for documents that will be viewed on a web browser. Technologies such as Cascading Style Sheets (CSS) and programming languages like JavaScript can support HTML. HTML documents are sent to web browser usually via a web server or locally stored files and afterwards are converted to multimedia web pages. HTML originally offered cues to the document's layout and described the structure of a web page semantically.

CSS [16] (Cascading Style Sheets) is a style language for describing how a document generated in a markup language like HTML appears. CSS specifies how elements, defined by HTML, are showed on a screen, in speech, on paper, or in other forms of media. According to W3C guidelines, CSS is one of the core languages of the open web and is standardized across Web browsers.

5.1.2 Back-End

The reference language of the back-end IANOS FundChain platform, in order to facilitate its functionalities, will be **Python** [17]. Python is a general-purpose programming language with a high level of abstraction, ideal for fast and easy prototyping. Its design philosophy prioritizes code readability. Python elements and object-oriented approach are intended to assist programmers in writing clear, logical code for both small and large-scale projects. Python supports a wide range of development methodologies, including structured, object-oriented, and functional programming.

5.2 Frameworks

5.2.1 Front-End

The framework utilized for the development of the front-end of the platform is **Angular** [18]. Angular is a Typescript-based programming environment. Angular is a component-based framework for developing scalable web applications, as well as a collection of libraries that cover a wide range of features such as routing, forms management, and client-server communication, as well as a set of developer tools for developing, building, testing, and updating the code.

5.2.2 Back-End

The three frameworks utilized for the development of the back-end of the platform are Django, Flask and PostgreSQL for the database.

Node.js [19] is a JavaScript runtime built on Chrome's V8 JavaScript engine that allows developers to run JavaScript code on the server-side. It provides an event-driven, non-blocking I/O model that makes it highly scalable and efficient for building real-time applications, web servers, APIs, and other networked applications. Node.js has a large and active community of developers and offers a wide range of modules and libraries through its built-in package manager, npm, which makes it easy to build complex applications using JavaScript on the server-side.

Flask [20] is a web application framework based on Python programming. Flask allows the developer to add application features as though they were built into the framework. Extensions are available for form validation, upload handling, object-relational mappers, several open authentication protocols, and other framework-related features.

PostgreSQL [21] is a feature-rich open-source relational database. It is built on an architecture that has earned its customers' trust in terms of dependability, data integrity, and appropriate operation. PostgreSQL is available for Linux, UNIX (AIX, BSD, HP-UX, SGI, IRIX, MAC OS X, Solaris, Tru64), and Windows. It is ACID compliant and supports the majority of SQL92 and SQL99 data types, such as INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, and TIMESTAMP. It also allows for the storage of big binary items such as photos, music, or films. It also provides programming environments for C, C++, Java, Perl, Ruby, Tcl, and finally Python, which is used in this project.

MongoDB [22] is a scalable, NoSQL database with a flexible document-oriented data model. It supports ACID transactions, offers advanced indexing, and has official drivers for popular programming languages. MongoDB is trusted for handling unstructured or semi-structured data, making it ideal for modern applications. Its distributed architecture ensures high availability and fault tolerance. MongoDB is widely used and supported by a large community of users and contributors.

6 Platform's architecture

6.1 Introduction

A blockchain-based equity crowdfunding platform is a novel way of raising funds for small businesses and startups. The use of blockchain technology in this platform brings several benefits such as security, transparency, and decentralization. A technical architecture that is well-designed and implemented is critical for the smooth operation of the platform and the satisfaction of its users. This section will begin by providing a birds-eye view of the various components that make up the technical architecture of the proposed blockchain-based equity crowdfunding platform, including the underlying blockchain technology, the smart contracts that govern the investment process and the off-chain components that contribute to the realization of the platform's overall functionalities. The components introduced here will be further analyzed in future sections of this document in order to provide a more in-depth view of the platform's internals.

Our objective is to provide a system architecture that is both practical and user-friendly for both technical and non-technical users of a blockchain-based equity crowdfunding platform. To cater to the diverse audience, the architecture must consider off-chain components that complement the functionalities provided by the DLT infrastructure. For instance, an identity and access management infrastructure for the actors and the services they consume, integration and deployment-related components such as API gateways, and user interfaces for the involved actors. However, this document will primarily focus on the flows and functionalities supported by the DLT infrastructure. While important, aspects related to UI/UX design, visualization, and similar topics are beyond the scope of this document.

Figure 6.1 provides a high-level architecture diagram that illustrates the overall structure and components involved in realizing the proposed, blockchain-based equity crowdfunding platform. In this diagram, the main building blocks of the system are represented, as well as the relationships and interactions between them. This allows stakeholders to better understand the platform's technical infrastructure and the role of each component in the overall functioning of the system.

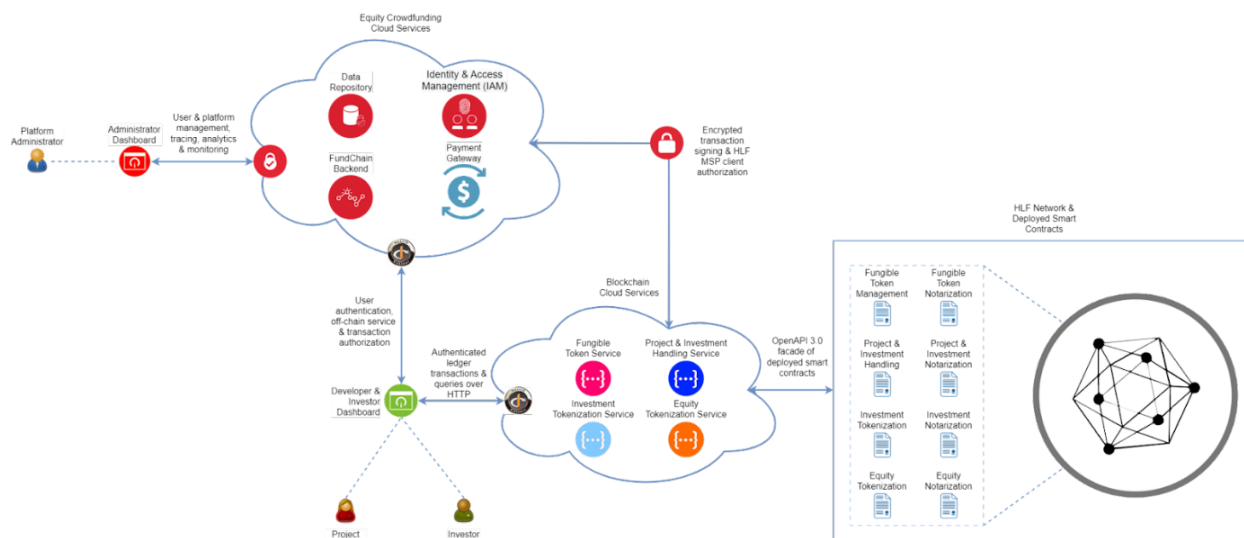


Figure 6.1: High-level overview of the blockchain-based equity crowdfunding framework.

We dedicate the remainder of this introductory section to, on the one hand, introduce the reader to the various components depicted in Figure 6.1 and, on the other hand, discuss technology-related choices. We begin with the latter of the two.

The right-hand side of Figure 6.1 depicts the blockchain network and the smart contracts that are deployed on it. Our rationale for choosing Hyperledger Fabric (HLF) as the underlying blockchain technology can be summarized as follows. Hyperledger Fabric is a permissioned blockchain platform that provides a high degree of privacy, scalability, and strong identity management features. These benefits make it an ideal choice for a blockchain-based equity crowdfunding platform. The permissioned nature of the network enables only authorized parties to participate in the network, ensuring the privacy and security of sensitive data. The scalable nature of the platform allows for efficient handling of high transaction volumes, ensuring smooth operation of the platform. Furthermore, the modularity and flexibility of Hyperledger Fabric allow for the creation of smart contracts, enabling the automation of certain processes, reducing the need for intermediaries and increasing the efficiency of the crowdfunding process. Finally, the strong identity management features of Hyperledger Fabric allow for a robust and trusted network where all actors are uniquely identified and their actions on the platform are fully auditable, increasing the transparency and trustworthiness of the system. These benefits make Hyperledger Fabric an ideal choice for a secure, efficient, and trustworthy equity crowdfunding platform.

As illustrated in Figure 6.1, the platform employs a collection of smart contracts to manage various aspects of its operations. In the following we provide a short summary of their functionalities. The fungible token smart contract forms the basis of the platform's stable coin construction, which allows payments (e.g., investments) and their respective settlements to be completely automated without requiring the involvement of external payment processors (and the incurred fees thereof). The investment & equity tokenization smart contracts model, respectively, investments and equity shares as non-fungible tokens (NFTs) on the ledger, which provide for verifiable ownership by the respective entities. The latter tokenization contracts are associated with their respective

notarization contracts that define, in short, the policies and conditions for minting the respective NFTs. Finally, the project and investment handling contract manages the lifecycle of crowdfunding projects and their associated investments, while the project and investment notarization contract evaluates policies and conditions that govern the state transitions of these objects.

We now shift our attention to all the off-chain components that are depicted in Figure 6.1, i.e., the components that compose the blockchain and equity crowdfunding cloud services and the user and administrator dashboards that are connected with them. In the context of the following paragraph, we discuss the security frameworks employed under which communication between the depicted cloud services takes place.

OAuth 2.0 and OpenID Connect are widely used open standards for securing web APIs and applications. OAuth 2.0 provides a secure and standardized way for users to grant access to their resources (such as photos, videos, or emails) to third-party applications or services, without giving away their login credentials. OpenID Connect builds on top of OAuth 2.0 and provides a standardized way to authenticate users and obtain identity information about them. Both protocols have become the de-facto standard for web security because they offer a high level of security and are widely supported by major software vendors and service providers. They provide a convenient way to enable users to sign in to a variety of applications and services using their existing credentials from a trusted identity provider, while also enabling developers to easily integrate security and identity management features into their applications and have interoperability guaranteed. OAuth 2.0 and OpenID Connect are widely recognized as industry-leading security protocols that provide robust authentication and authorization mechanisms for web applications. They are particularly useful for applications that manage sensitive data, as is the case for our domain of interest. Additionally, these protocols enable the use of modern multi-factor authentication mechanisms, such as biometric or one-time password (OTP) verification, to further enhance security. As a result, they posit as essential components of any modern web application and thus, constitute, our protocols of choice regarding web security.

The Equity Crowdfunding Cloud Services of Figure 6.1 is composed of a collection of services that are essential, on the one hand, delivering the platform's functionalities, as identified in the respective requirement sections and, on the other hand, providing all the necessary data points that are consumed by the dashboards of the platform's users, i.e., administrator, investors, project developers and pass by visitors, the latter of which for the sake of simplicity of presentation are not depicted in the figure.

The Identity and Access Management (IAM) component is a service that is compliant with OAuth 2.0 and OpenID Connect protocols. It is responsible for managing registration requests, as well as, the authentication and authorization of platform users, allowing them to securely access the platform's resources. In addition to managing user identity, the IAM component generates and manages X.509 digital certificates for users, which are used to sign transactions on the blockchain. These certificates provide an additional layer of security to the platform by ensuring that only authorized users are able to interact with the blockchain. By combining the IAM component with the platform's use of X.509

certificates, the platform is able to provide a highly secure environment for users to invest in projects with confidence.

The Payment Gateway component is a crucial aspect of the equity crowdfunding platform's funding phase. It enables investors to fund projects through an integrated payment system that supports popular online payment services such as GooglePay and PayPal. Additionally, this component plays a focal role in the creation and management of stable coins, leveraging the fungible token functionality of the smart contracts deployed on the Hyperledger Fabric network. This integration ensures a secure, efficient, and transparent flow of funds throughout the platform while providing a seamless and user-friendly experience for all participants involved.

Lastly, the "FundChain Backend" is a vital component of the blockchain-based equity crowdfunding platform that provides the necessary infrastructure for storing and managing off-chain data. It is paired with a data repository that contains essential information for the platform's operations, such as project details, and it is designed to be scalable and flexible in querying to meet the growing demands of the platform's users. NoSQL databases like MongoDB are preferred for handling large volumes of data, ensuring high scalability and availability. The backend's architecture also allows for real-time data processing and analytics using Apache Kafka and Apache Spark, providing valuable insights into platform operations. To enable efficient querying and modification of the data, the backend is complemented by a set of expressive web APIs that adhere to the security and privacy requirements of the platform. In addition, this component manages various events related to, e.g., the lifecycle of projects and investments, such as timers evaluating the funding status of projects and ensuring they meet the necessary requirements to progress to their implementation phase, orchestrating processes for refunding investments in case projects fail to meet their funding goal, and others.

The Blockchain Cloud Services of Figure 6.1 is composed of a collection of services that are designed to expose an OpenAPI 3.0 compliant specification of HTTP endpoints. In simple terms, these endpoints are mapped to functions of smart contracts that are deployed on top of a Hyperledger Fabric network. The main aim of these services is to provide a simple and easy-to-use HTTP interface that can be consumed by off-chain web applications. This set of services offers a range of functionalities such as handling fungible tokens, managing investments and projects, as well as creating investment and equity certificates using NFT technologies. Apart from some internal caching that provides for increased performance, these services are completely stateless, i.e., they are not coupled with any sort of database. As illustrated in the figure, the Blockchain Cloud Services communicate over encrypted authenticated channels with the Equity Crowdfunding Cloud Services, and more specifically, with the platform's IAM component, for transaction signing and client authorization purposes when attempting to interface with the HLF network. This provides a robust, secure and scalable solution that is well-suited for handling sensitive and complex blockchain transactions and queries.

In conclusion, this introductory section has provided a high-level overview of the architecture of the blockchain-based equity crowdfunding platform. It has touched on the use of Hyperledger Fabric as a permissioned blockchain platform, as well as the

OAuth 2.0 and OpenID Connect protocols that ensure secure identity and access management for the platform's users. Additionally, we have introduced key components such as the Blockchain Cloud Services and the Equity Crowdfunding Cloud Services that form the foundation of the platform's functionality. The following subsections are dedicated to providing a more detailed look into concepts and flows involved in the architecture that warrant additional elaboration, prior to advancing the description to the internal architecture of the presented components that will additionally highlight their functionalities and interactions.

6.2 Notarizing Smart Contract Operations

In the overwhelming majority of cases, software development is a process that begins with a business need or requirement. It involves a systematic approach to building, designing, testing, and maintaining software that fulfills that need. The process is iterative and involves several stages, each with its unique set of challenges and considerations. A thorough understanding of the business need is critical for successful software development, as it forms the foundation of any project and guides the design and development process. The development team must work closely with stakeholders and subject matter experts to ensure that the software solution meets the requirements and provides value to the organization. That being said, there is an important, yet very subtle, decomposition of the meaning of the term *business logic* that warrants discussion.

Consider the case of a simple transfer of funds between bank accounts. The operation could be separated into two distinct processes. The first process would focus on verifying that the transfer can take place, by checking factors such as whether the account has sufficient funds, whether the account is active, whether the account is subject to restrictions, and so on. The second process would be responsible for actually executing the transfer of funds, which could involve steps such as debiting the funds from the source account and crediting them to the target account.

More broadly, we argue that separating the business logic of any operation into two distinct processes, one that is solely concerned with the decision-making process about whether the operation should be executed, or not, and one that is solely concerned with the (algorithmic) steps that are involved in executing said operation, provides substantial benefits, which can be summarized as follows. It enables greater modularity, maintainability, and flexibility, as the processes can be modified and tested independently of each other. Among others, this can help prevent errors, malicious attacks, and other security risks.

The aforementioned rationale forms the basis of what we refer to as the *notary interaction pattern*, a novel mechanism that was initially devised as part of the efforts of the RENAISSANCE H2020. The objective was twofold. First, provisioning a generic mechanism for minting coins to fungible token accounts by abstracting out of the fungible token smart contract the conditions that govern whether a mint operation should be executed, or not. Second, to address the disparity regarding the rules governing financial settlements across the different smart contract-based marketplaces developed in the context of the project. The notary interaction pattern was, subsequently,

33



utilized in the context of the FEVER H2020 project for similar purposes, as well as, in the context of Pioneer Use Case (PUC) 2 (entitled “Transparent Management of Public Accounts”) of the TOKEN H2020 project, which delivered a blockchain-based framework that allows municipalities to digitize the entire lifecycle of arbitrary procurements.

In the context of this effort, the notary interaction pattern will be employed as a base framework to manage the lifecycle, or more specifically, the rules governing:

1. The state transitions of the project and investment lifecycles and the respective emitted events.
2. The minting and burning of fungible tokens, i.e., the maintenance of the stable coin construction.
3. The minting of non-fungible tokens, i.e., the issuance of equity and investment certificates.

To this end, we will describe the pattern in detail in the remainder of this section. In the following, we provide a descriptive overview of the entities that are involved in the notary interaction pattern, along with an explanation of their role:

1. **Client:** The entity, human or non-human, that wishes to execute an operation of the service contract.
2. **Service Contract:** A smart contract that can be invoked by clients and contains the business logic for the operation.
3. **Notary Contract:** A smart contract that governs the policies for executing operations invoked by clients. This includes access control, authorization, and other policies.

The process begins with the client issuing a request to the service contract's notarized operation by invoking the function "requestOperation()" (Step 1). The service contract assigns a unique identifier to the request and returns it to the client (Step 2). At a later time, the client invokes the service contract to execute the operation by supplying the identifier of the operation's instance (Step 3) and the service contract invokes the notary contract for this operation (Step 4). The notary contract responds with a status code (Step 5), which the service contract uses to determine whether to proceed with the operation's instance (Step 6). Finally, the service contract provides the client with the result (Step 7).

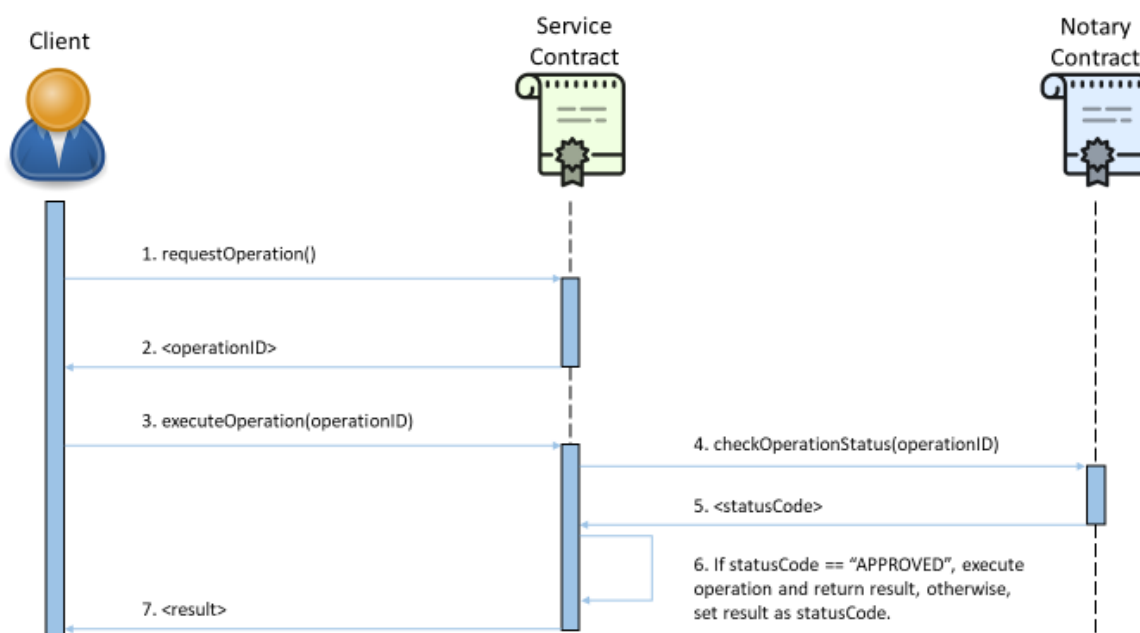


Figure 6.2: Sequence diagram that depicts how the entities involved in the notary interaction pattern interact with each other

Several key points should be noted about the notary interaction pattern. Firstly, successful completion of the pattern requires two transactions in a happy-path scenario, with Steps 1-2 occurring in the first transaction and Steps 3-7 in the second. Secondly, the pattern is asynchronous, with no inherent timing assumptions. Thirdly, each notarized operation of a service contract is associated with a specific notary, and a service contract can use different notaries for different operations. The implementation of notaries can be shared across multiple service contracts and updated dynamically. Fourthly, any peculiarities in the implementation of the service and notary contracts are completely abstracted from the protocol. Lastly, from the perspective of the service contract, the notary is simply a function call, and configuration is required to ensure proper interaction between the service and notary contracts in Hyperledger Fabric. Specifically, the service contract must be configured with the channel name, notary contract name, and notarization function name, and the notary contract must be instantiated on the same ledger as the service contract in the same channel to allow for contract state modifications.

The following list provides a description of the status codes that may be returned by a notary contract in response to its notarization function:

- **NOT_FOUND:** The notary is not aware of an operation instance that corresponds to the input identifier.
- **PENDING:** The notary is aware of the operation's instance, but an affidavit regarding its execution is not yet available.
- **APPROVED:** The policies governing the execution of this operation's instance have been met, and the service contract may proceed with executing the business logic.

- **REJECTED:** The evaluation of the policies governing the execution of this operation's instance instructs the service contract not to proceed with executing the business logic.

These status codes have been found to be sufficient for all the aforementioned use cases. However, designers and developers have the flexibility to extend the set of supported status codes to suit the specific requirements of their use case. Furthermore, we emphasize that the "contract" conveyed by these status codes and their particular meaning is specific to the use case. For instance, in response to a "REJECTED" status code, the service contract may perform other business logic before responding to the client.

6.3 Finite State Machines

A finite state machine (FSM) is a mathematical model used to represent and control systems with multiple states and transitions. It consists of a set of states, a set of inputs or events that can trigger state transitions, and a set of rules that govern the transitions between states based on the inputs or events received. FSMs are commonly used in computer science to model systems with discrete behavior, such as software protocols, digital circuits, and user interfaces. One prominent example of FSMs in use is in the development of compilers, which use FSMs to transform high-level programming languages into machine code. Another example is in the design of digital electronics, where FSMs are used to specify the behavior of electronic components and circuits.

Crowdfunding projects go through various state transitions as they progress from their inception to their completion. Initially, they are in the planning stage, during which they are being evaluated and refined for suitability and feasibility. Once they are approved, they move to the fundraising stage, during which the actual fundraising process takes place. If the fundraising is successful, the project moves to the execution stage, where the project is implemented as per the plan. Finally, the project moves to the completion stage, where it is evaluated for its success and the results are communicated to the backers.

Correspondingly, an investment typically goes through various state transitions. At first, the investment opportunity is open, and investors can make pledges to invest in the project. Once the investment target is met, the investment opportunity moves to a funded state. In this state, the project developers can access the funds, and the investment contracts between the investors and the project developers are initiated. If the investment target is not met within the specified timeframe, the investment opportunity, for instance, may move to a failed state, and investors can withdraw their pledges. After the investment is funded, the project goes through different stages of development, such as planning, execution, and delivery, and the investors can track the progress. Finally, when the project is completed, the investment opportunity moves to a closed state, and the investors receive their returns.

Based on the aforementioned, it follows that modeling crowdfunding projects and investments as finite state machines provides several benefits in the context of the proposed blockchain-based equity crowdfunding platform. First, it allows for a clear

representation of the various stages and transitions that a project or investment can go through, providing a structured and consistent framework for tracking and managing them. This facilitates better decision-making and risk management by enabling platform administrators and investors to easily understand the current state of a project or investment and the available options for moving it forward. Additionally, finite state machines can enable the automation of certain processes, such as triggering the disbursement of funds to a project upon reaching a certain funding threshold, making the platform more efficient and less prone to errors. Finally, it enables the platform to leverage the full power of smart contracts and blockchain technology, allowing for the secure and transparent execution of state transitions and ensuring that all parties involved have a tamper-proof record of the history of the project or investment.

Now that we have introduced the benefits of modeling crowdfunding projects and investments as finite state machines, we will now present and discuss the finite state machines for projects and investments, respectively, in the context of the proposed blockchain-based equity crowdfunding platform. By modeling the various states and transitions that projects and investments go through, we can more easily track and manage them on the platform, providing a more streamlined and transparent experience for all users involved.

6.3.1 Projects

In order to have a clear understanding of the state transitions that crowdfunding projects go through, a finite state machine can be used to model the different phases of the project's life cycle. This approach allows for a systematic analysis of the various stages of a project and the transitions between them, providing a visual representation of the different states and the conditions that trigger their transitions. In this section, we will present and discuss the finite state machine of crowdfunding projects in the context of the proposed blockchain-based equity crowdfunding platform.

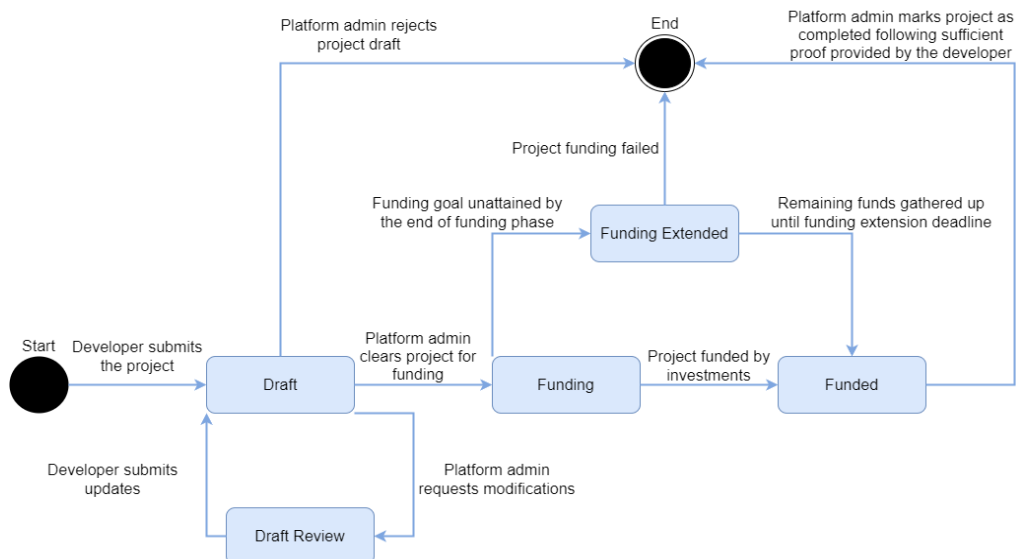


Figure 6.3: Finite state machine diagram illustrating the state transitions that a crowdfunding project goes through, with various conditions and actions associated with each state.

Figure 6.3 illustrates a finite state machine that depicts the state transitions for a crowdfunding project. The machine begins in the "Draft" state, where the project is being created and edited by the project developer. From there, it can transition to the "Draft Review" state where a platform administrator reviews the project and decides whether additional information, modifications, or other relevant updates are warranted. As illustrated in the figure, this is a cyclical, or back and forth, process that can be repeated multiple times up until a platform administrator decides to either approve the project to transition to the "Funding" state, or reject it in its entirety. If approved, the project transitions to the "Funding" state, where it goes live and investors can contribute towards its funding goal. After an investor commits funds to a crowdfunding project, an investment certificate is minted and distributed to her wallet.

If the project does not reach its funding goal before the deadline, it transitions to the "Funding Extended" state, where the project is given, put simply, a second chance to accumulate all remaining funds up until a predefined deadline. Including this additional state in the crowdfunding project's finite state machine allows project developers and other interested parties to compensate (as investors) for any remaining funding needed to reach the project's goal. If this process fails, the project transitions to a "Funding Failed" state, where it is essentially considered to be closed and all accumulated funds (if any) are returned to the respective investors. If the project reaches its funding goal by the end of its designated funding period, it transitions to the "Funded" state, where all necessary funds have been collected. From there, it can move to the "Completed" state once the project developer provides convincing evidence to the platform's administrators.

After a crowdfunding project comes to a successful completion, the investors who supported the project are granted equity certificates that represent their ownership in the venture. These certificates are digitally minted and distributed to the investors' wallets, providing them with a tangible and secure means of holding and proving their shares. This process ensures that the investors are able to reap the benefits of their contributions to the project.

Table 6.1: Mapping between the names of project states and their corresponding string encoding

State Name	String Encoding
Draft	draft
Draft Review	draft_review
Funding	funding
Funding Extended	funding_ext
Draft Rejected	draft_rej
Funding Failed	funding_fail
Funded	funded
Completed	completed

The above table provides a mapping between the names of project states as they appear in the previously presented finite state machine diagram and their corresponding string encoding. When designing a finite state machine, it is often helpful to use descriptive names for the states to make the diagram more readable and understandable. However,

when implementing the machine in code, it is more efficient to represent the states as strings. This table serves as a reference for translating between the two representations.

6.3.2 Investments

In the context of an equity crowdfunding platform, investments are subject to various state transitions that reflect the progress of the funding campaign and the fulfillment of obligations by both the project developers and investors. Therefore, modeling investments as finite state machines can be particularly useful to provide a clear understanding of the investment process and ensure that all parties involved are aware of their responsibilities at each stage. This section presents and discusses the finite state machine of investments in the proposed blockchain-based equity crowdfunding platform, highlighting the benefits of this approach and detailing the different states and their associated actions.

The following figure illustrates a finite state machine that depicts the various states an investment can go through in the proposed blockchain-based equity crowdfunding platform. The initial state is "Intent" which represents the investor's initial intention of investing to a crowdfunding project. This is an intermediary state for investment's whose payment method depends on an external payment processor (e.g., PayPal). Once the investment's settlement takes place, the investment will transition to the "Funds Reserved" state. For investments that employ the platform's stable coin construction, assuming their successful submission, they immediately transition to a "Funds Reserved" state. If the project is designated as "Funding Failed", any investments made towards it will be automatically refunded and recorded as such.

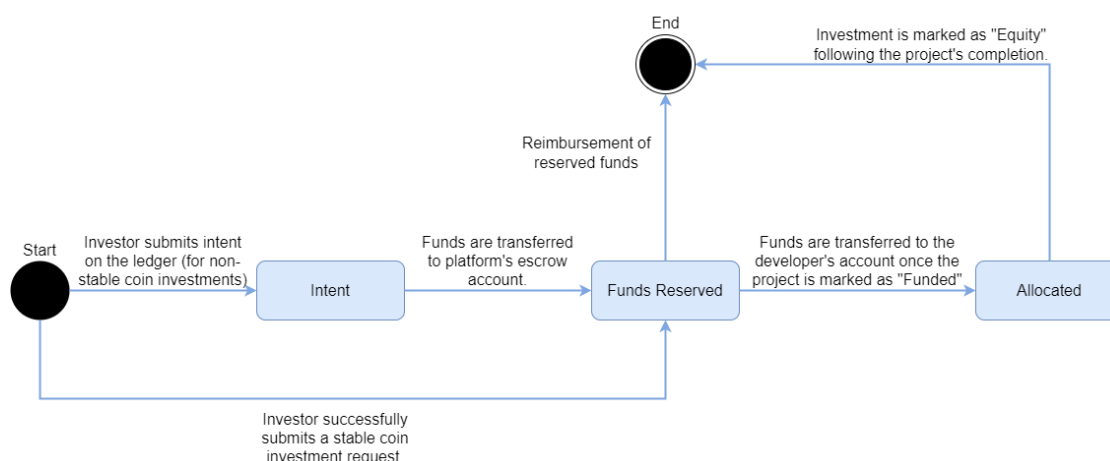


Figure 6.4: Finite state machine diagram illustrating the state transitions that an investment goes through, with various conditions and actions associated with each state.

Upon the successful funding of the project, all investments undergo an automated transition to the "Allocated" state. In this state, the invested funds are promptly transferred to the designated account of the developer, thereby providing them with the necessary resources to commence the project development phase. Finally, following the successful completion of the project, each investment reaches the pinnacle of its journey and enters the esteemed "Equity" state. This entails the creation of an exclusive equity

certificate, symbolizing the investor's ownership in the project and their stake in its future success.

Table 6.2: Mapping between the names of project states and their corresponding string encoding

State Name	String Encoding
Intent	intent
Funds Reserved	reserved
Reimbursed	reimbursed
Allocated	allocated
Equity	equity

The above table provides a mapping between the names of investment states as they appear in the respective finite state machine diagram and their corresponding string encoding. In the context of investment state tracking, it is useful to have a mapping between the state names as they appear in the finite state machine diagram and their corresponding string encoding to facilitate communication and data transfer between different parts of the system.

6.4 Platform Core Flow

To facilitate reader understanding on how the platform works, we have prepared a series of sequence diagrams that illustrate the key processes and interactions between actors and components. These diagrams demonstrate the platform's core flows, including the creation of new crowdfunding projects, investing in projects, managing failed and successful projects, tokenizing investments and equity shares, and more. By examining these sequences in detail, stakeholders can gain a comprehensive understanding of the platform's inner workings, enabling them to make informed decisions when using or interacting with the platform. In what follows, we dedicate a separate subsection for each core flow of the platform.

6.4.1 Project Creation

Creating a new crowdfunding project is a straightforward process that involves a series of steps to ensure the smooth initiation of the project's lifecycle. An overview of these steps is presented in the following figure and are further analyzed in the context of this section.

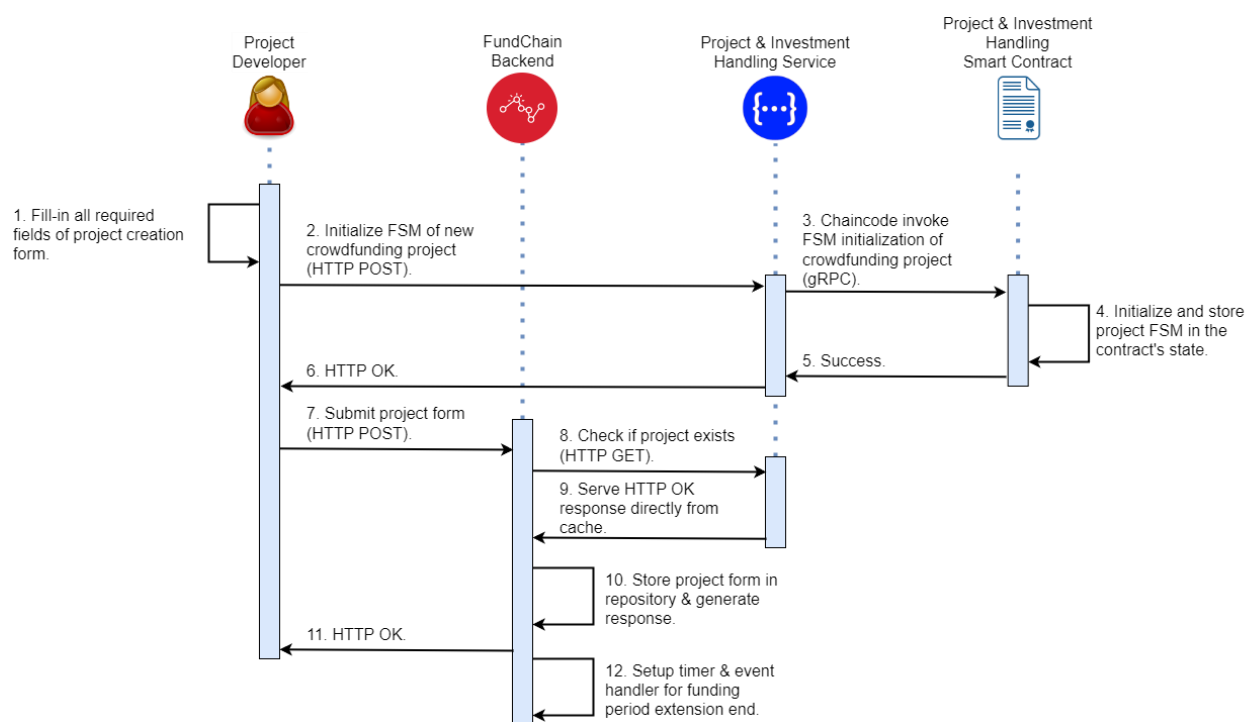


Figure 6.5: Sequence diagram illustrating the actors and the steps involved in creating a new crowdfunding project.

To create a new project, the project developer must fill in all required fields of the project creation form, including the project's name, funding goal, and other essential details. Once the form is complete, the project developer requests the FSM initialization of a new crowdfunding project by issuing an HTTP POST request to the Project & Investment Handling service. The Project & Investment Handling service then invokes the respective function of the Project & Investment Handling smart contract and propagates the successful response all the way back to the project developer. Once the project developer receives the confirmation, she submits the project's filled-in form to the FundChain backend, which verifies that the project has been successfully initialized on the ledger. The FundChain backend stores the project's form in its repository, generates the appropriate response, and responds successfully to the project developer. Additionally, the FundChain backend sets up timers and event handlers to manage various events, such as the end of the project's funding period. By following these steps, the project developer can create a new crowdfunding project and ensure that it is successfully initialized on the platform.

6.4.2 Investment

The FundChain platform provides to interested investors two schemes for investing in a crowdfunding project. The first scheme is based on real-world Internet payments, which are facilitated by the platform's payment gateway component. The second scheme involves the platform's novel stable coin construction. We dedicate separate subsections to introduce the steps involved in each of the aforementioned investment schemes.

6.4.2.1 Internet Payment

Investing in a crowdfunding project through the platform's payment gateway component involves a two-phase process.

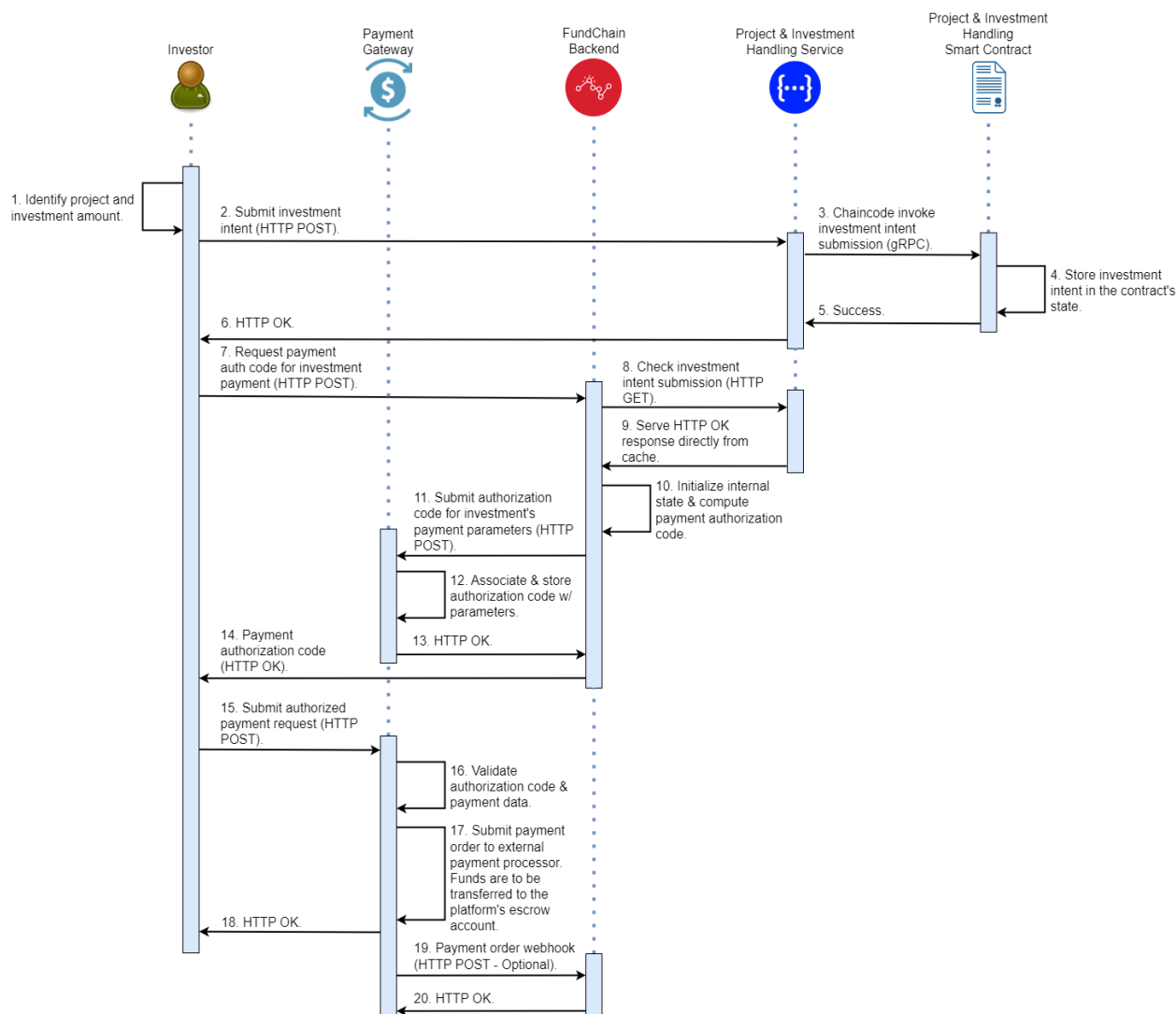


Figure 6.6: Sequence diagram illustrating the actors and the steps involved in the investment intent submission phase.

In the first phase (Figure 6.6), the investor selects the project and the investment amount, then submits an investment intent request to the Project & Investment Handling service, which invokes the appropriate function of the Project & Investment Handling smart contract. The smart contract stores the intent in its state and propagates a successful response back to the investor. The investor then requests a payment authorization code for the newly submitted investment intent from the FundChain backend. The FundChain backend verifies that the investor's intent has been successfully published on the ledger and initializes its internal state for this particular investment intent, and computes a fresh payment authorization code. This authorization code, along with data regarding the respective payment, is communicated to the platform's payment gateway, which then returns the payment authorization code to the investor. The investor then invokes the payment endpoint of the payment gateway with the newly acquired authorization code.

The payment gateway validates the input data provided by the investor and submits an appropriate payment order to the external payment processor to transfer the investment's funds to the platform's escrow account. The payment gateway returns a successful response to the investor, and may optionally issue a webhook to the FundChain backend to inform the latter that the payment order has been submitted.

In the second phase (Figure 6.7), upon receipt of a webhook by the payment processor indicating that the investment's payment has been settled, the payment gateway notifies the FundChain backend with an appropriate webhook. The FundChain backend then sets the investment's state to "Funds Reserved" by issuing an HTTP POST request to the Project & Investment Handling service, which in turn invokes the appropriate function of the smart contract to validate the state transition and update its state accordingly. A successful response is then propagated all the way back to the FundChain backend. Finally, the FundChain backend evaluates whether the project has achieved its funding goal and executes the respective business logic if so.

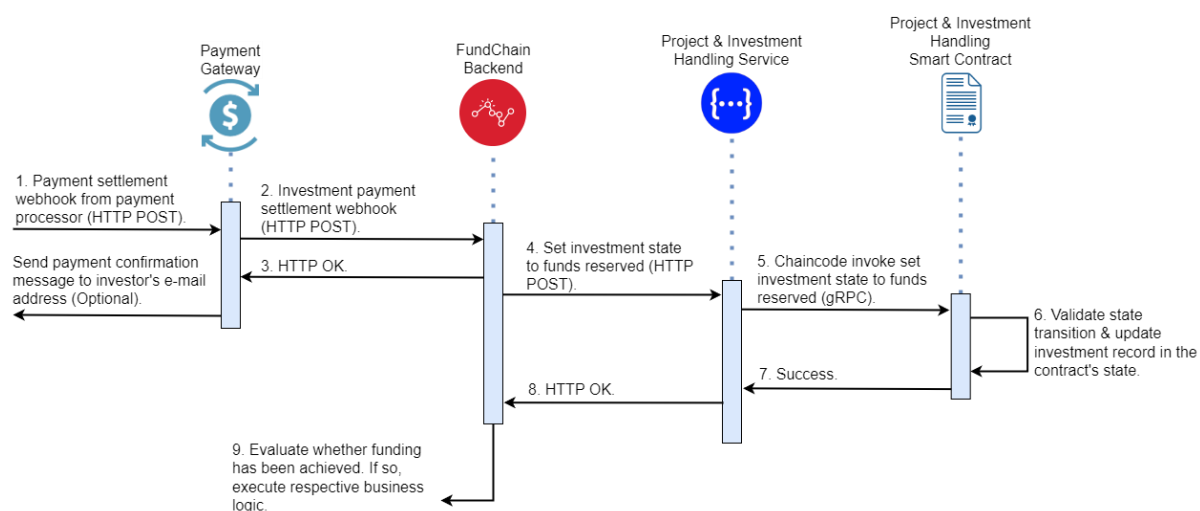


Figure 6.7: Sequence diagram illustrating the actors and the steps involved in the investment settlement phase.

This two-phase process ensures that investments are properly authorized and the funds are held in escrow until the project meets its funding goal (or not).

6.4.2.2 Stable Coin Payment

Investing in a crowdfunding project via the platform's novel stable coin mechanism involves a streamlined process that is depicted in Figure 6.8. The first step is for the investor to identify the project of interest and the amount she wishes to invest. The investor then submits a stable coin investment request to the Project & Investment Handling service, which invokes the appropriate function of the Project & Investment Handling smart contract. The smart contract issues a coin hold request to the fungible token smart contract, which validates the request, locks the coin balance from the investor's account, and modifies its internal state accordingly. The fungible token smart contract returns the identifier of the coin hold to the Project & Investment Handling smart contract.

Next, the Project & Investment Handling smart contract stores the appropriate investment record to its internal state, and a successful response is propagated all the way back to the investor. The investor then submits a stable coin investment request to the FundChain backend, which verifies that the stable coin investment has been successfully initialized on the ledger. If successful, the FundChain backend responds with a success message to the investor.

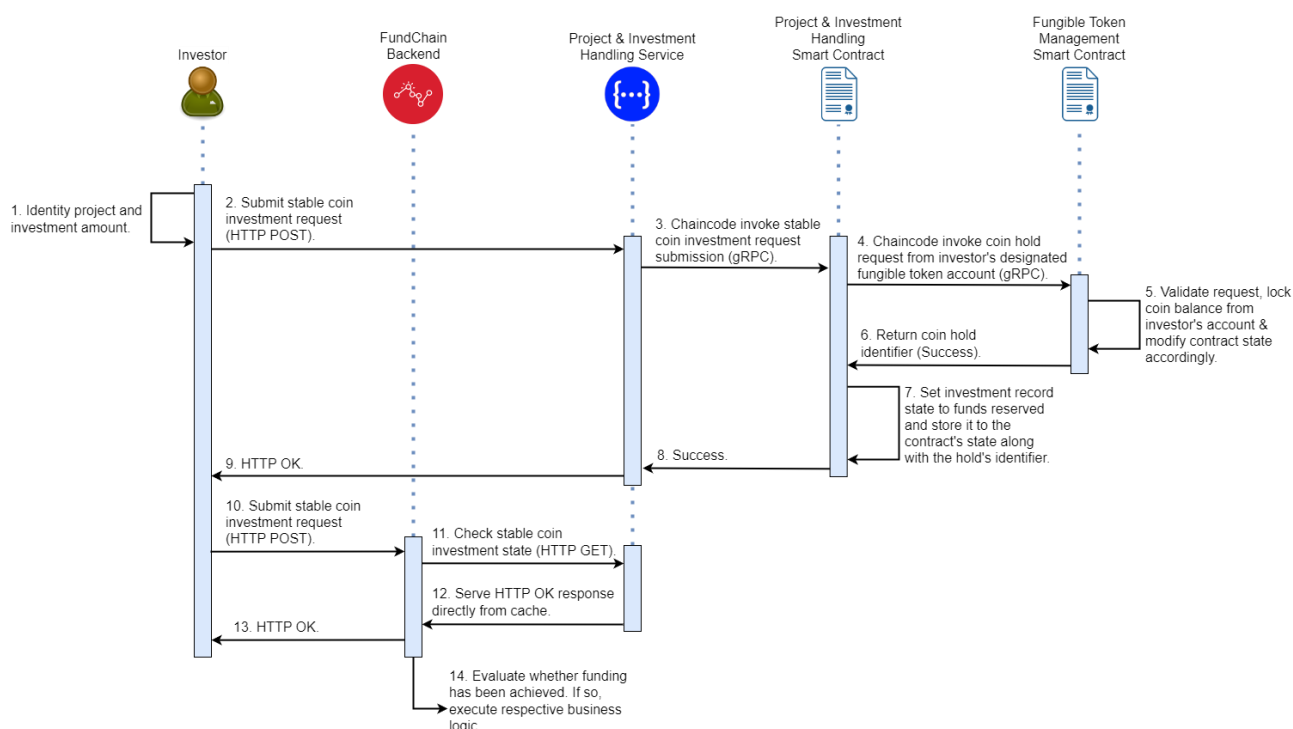


Figure 6.8: Sequence diagram illustrating the actors and the steps involved in investing in a crowdfunding project via stable coin payments.

Lastly, the FundChain backend evaluates whether the project has achieved its funding goal and, if so, executes the respective business logic.

6.4.3 Funding Failed

When crowdfunding projects fail to achieve their funding goal, investors must be reimbursed for their contributions. The process of reimbursing investment funds is, as is the case with investments, split into two cases. The first case addresses investments made with real-world Internet payments via the payment gateway. The second case addresses investments made with the platform's novel stable coin construction. In both cases, the platform must ensure that the reimbursement process is executed securely and efficiently. We dedicate the following two subsections to introduce the steps involved in each reimbursement case.

6.4.3.1 Internet Payment - Investment Reimbursement

For investments whose funds were committed via standard Internet payments facilitate through the platform's gateway component, the reimbursement process is split into two phases, namely, the investment reimbursement submission phase and the investment reimbursement settlement phase.

During the investment reimbursement submission phase (Figure 6.9), a timer in the FundChain backend fires when the project's funding extension period has ended. The FundChain backend retrieves the project's investments by requesting the Project & Investment Handling service through an HTTP GET request. The service queries the

respective function of the smart contract, which then returns an array containing all of the investments made for the project.

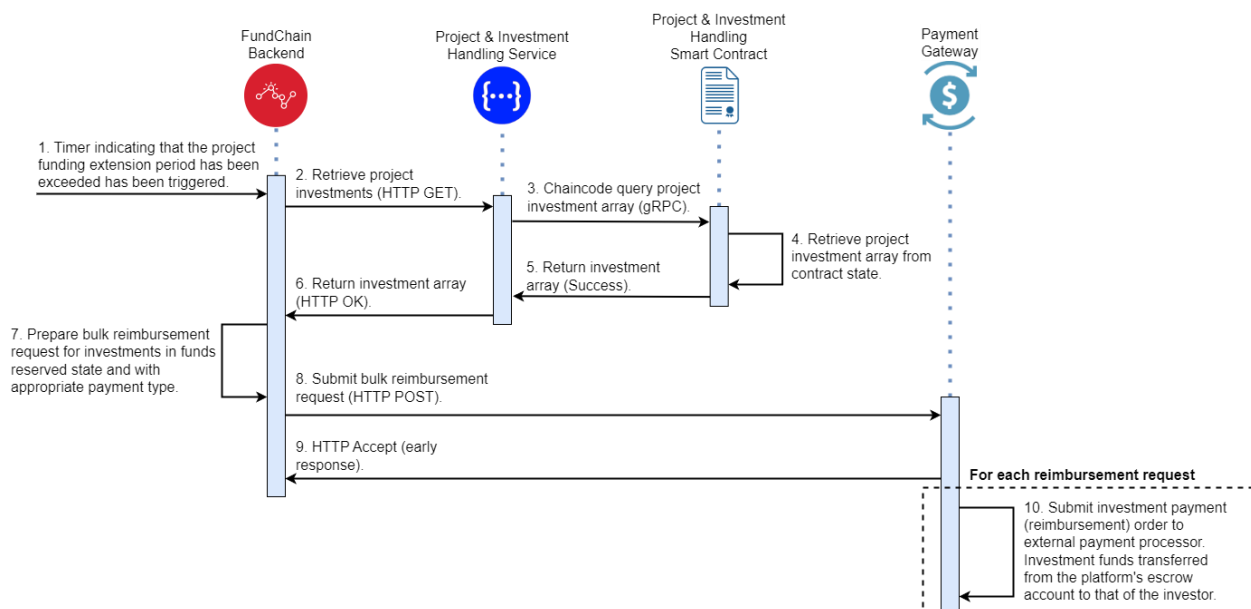


Figure 6.9: Sequence diagram illustrating the actors and the steps involved in the investment reimbursement submission phase.

Afterward, the FundChain backend prepares a bulk reimbursement request for all investments that are in a "Funds Reserved" state and whose payment was handled by the platform's payment gateway. This bulk reimbursement request is then submitted to the payment gateway, which returns an HTTP Accept response immediately, indicating the request's successful scheduling. Asynchronously, the payment gateway then iterates through each reimbursement request and submits an appropriately formed payment order to the respective external payment processor, requesting that funds are transferred from the platform's escrow account to that of the respective investor.

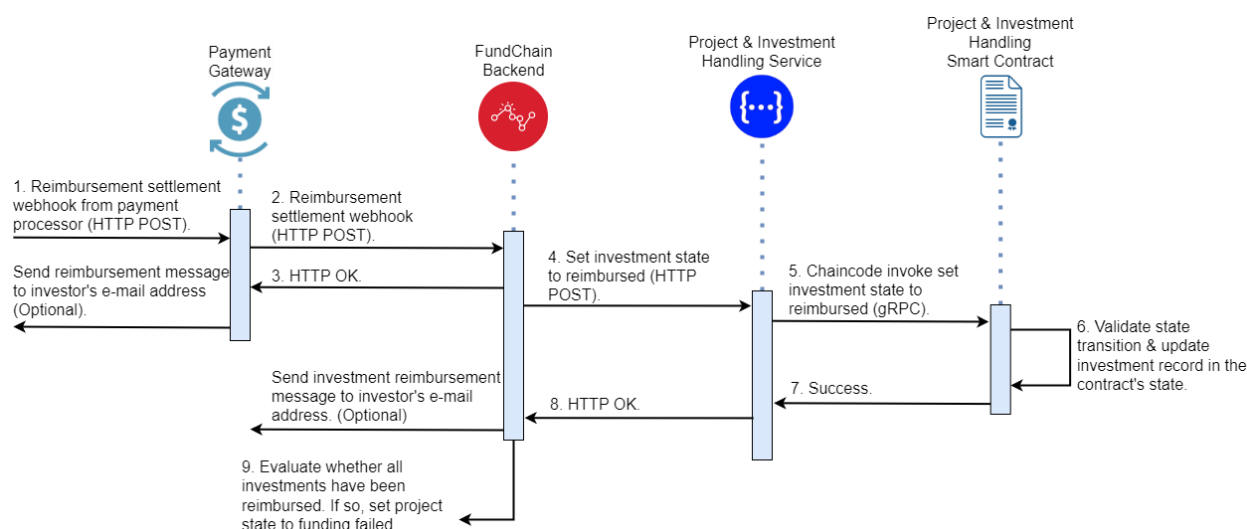


Figure 6.10: Sequence diagram illustrating the actors and the steps involved in the investment reimbursement settlement phase.

In the investment reimbursement settlement phase (Figure 6.10), the payment processor notifies the payment gateway of the successful reimbursement of funds through a webhook. The payment gateway then notifies the FundChain backend using an appropriate webhook. The FundChain backend then sets the investment's state to "Reimbursed" by issuing an HTTP POST request to the Project & Investment Handling service.

The Project & Investment Handling service invokes the appropriate function of the smart contract, which validates the state transition and updates its state accordingly. A successful response is propagated all the way back to the FundChain backend. Finally, the FundChain backend evaluates whether all project investments have been successfully reimbursed and sets the project's state to "Funding Failed" if that is the case.

6.4.3.2 Stable Coin Payment - Investment Reimbursement

In addition to standard Internet payments, FundChain platform also supports a novel stable coin mechanism for investing in crowdfunding projects. This mechanism requires a slightly different process for investment reimbursement. The process is depicted in the following sequence diagram (Figure 6.11) and involves the following steps.

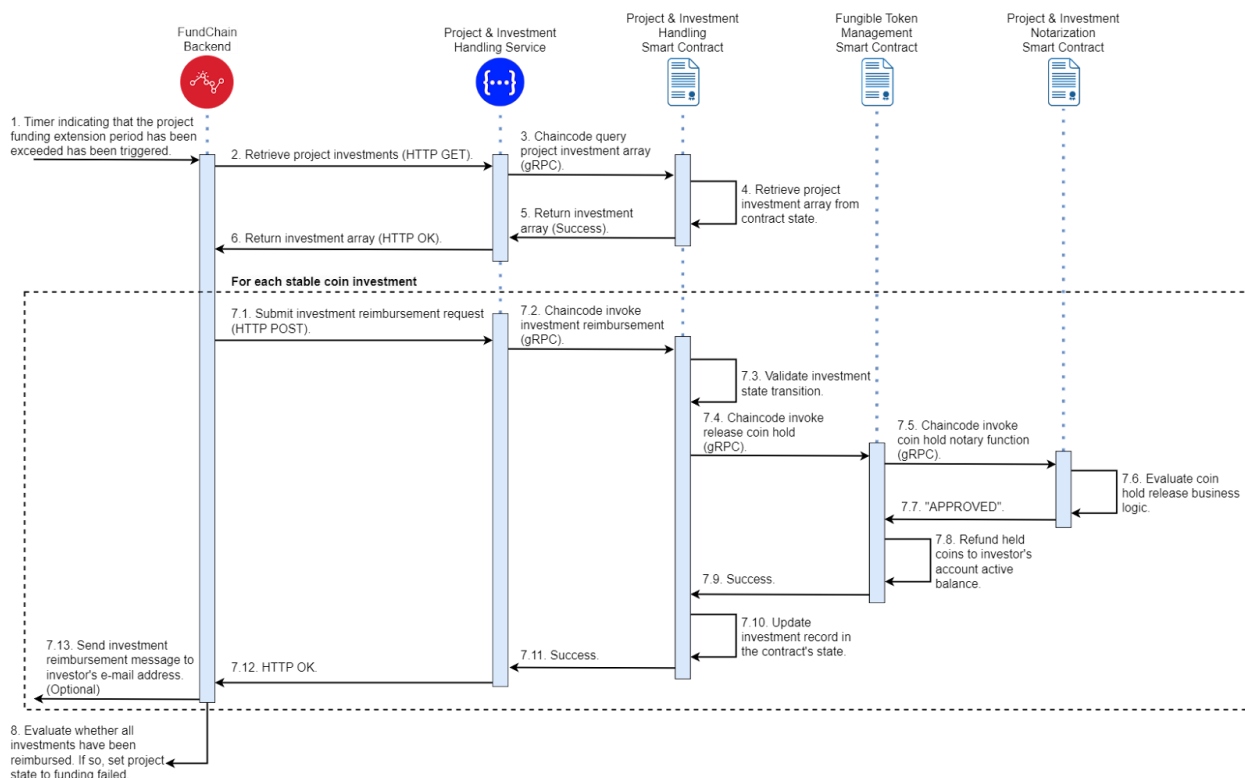


Figure 6.11: Sequence diagram illustrating the actors and the steps involved in reimbursing stable coin investments of a crowdfunding project.

Firstly, when the funding extension period for a project has been exceeded, a timer fires in the FundChain backend to initiate the investment reimbursement process. The FundChain backend then retrieves the project's investments by issuing a HTTP GET request to the Project & Investment Handling service, which in turn queries the smart contract function and returns an array of all the project's investments to the FundChain backend.

For each stable coin investment, the FundChain backend submits a reimbursement request to the Project & Investment Handling service. The service then invokes the appropriate function of the smart contract which validates the state transition of the investment. Assuming the state transition is valid, the release coin hold function of the fungible token smart contract is invoked.

The fungible token smart contract then calls the coin hold notary function of the Project & Investment notarization smart contract, which evaluates the business logic regarding the release of the coin holds. Once the release of the held coins is approved, the fungible token smart contract refunds the held coins to the investor's fungible token account.

After the coin hold has been released and the held coins have been refunded, the fungible token successfully returns the call stack back to the Project & Investment handling smart contract, which updates the investment's record accordingly in its internal state. A successful response is then propagated back to the FundChain backend. Optionally, the FundChain backend may transmit an investment reimbursement message to the investor's e-mail address.

Once all stable coin investments have been reimbursed, the FundChain backend evaluates whether all project investments have been successfully reimbursed. If they have, the respective business logic is executed, which sets the project's state to "Funding Failed".

6.4.4 Project Funded

When a crowdfunding project reaches its funding goal, the project developer can finally access the funds that have been committed by investors. In this section, we will discuss the steps involved in moving funds to the project developer's account. The process is split into two subsections to address the two possible scenarios for investments: regular Internet payment schemes and the platform's stable coin construction. In the first subsection, we will discuss the steps involved in moving funds for projects that received investments via standard Internet payment schemes. In the second subsection, we will discuss the steps involved in moving funds for projects that received investments via the platform's stable coin mechanism. The goal of both subsections is to provide a detailed overview of the process, highlighting the different parties involved and the roles they play, as well as the technical steps required to transfer funds securely and efficiently.

6.4.4.1 Internet Payment - Investment Allocation

When a crowdfunding project attains its funding goal, the investment funds that were committed via regular Internet payment schemes facilitated through the platform's payment gateway are moved to the project developer's account. This process is split into two phases: the investment allocation submission phase and the investment allocation settlement phase.

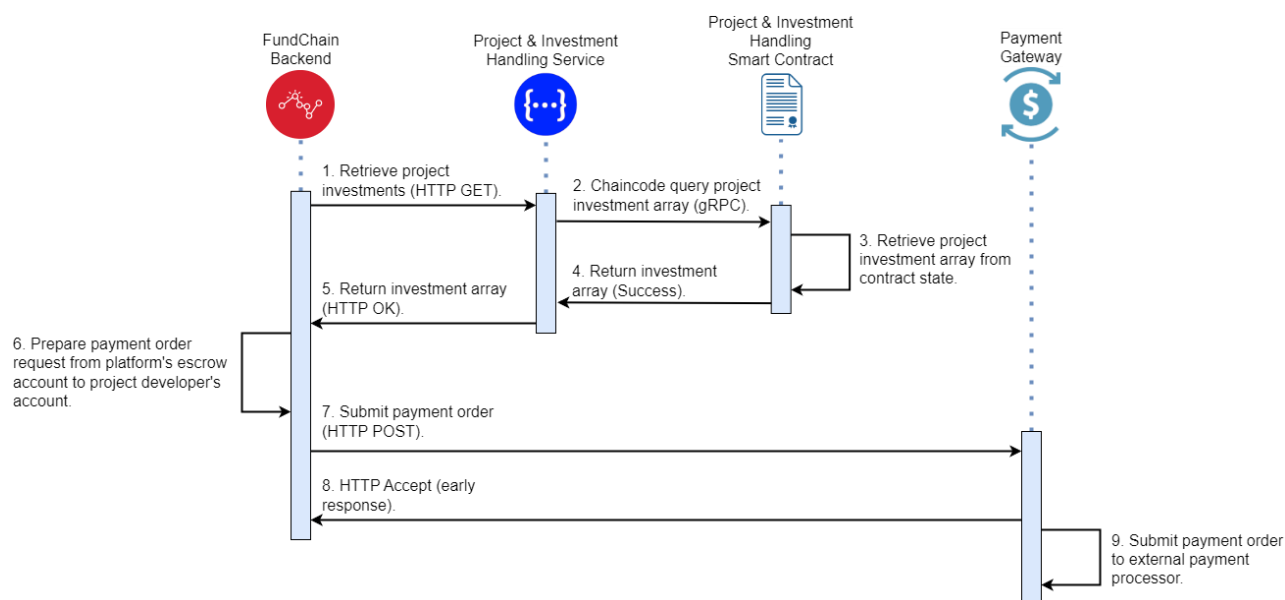


Figure 6.12: Sequence diagram illustrating the actors and the steps involved in the investment allocation submission phase.

In the investment allocation submission phase (Figure 6.12), the FundChain backend retrieves the project's investments, sums up the investment funds that were committed

via regular Internet payment schemes, and prepares a payment order request to transfer the computed sum from the platform's escrow account to the project developer's account. The payment order is submitted to the payment gateway, which responds immediately with an HTTP Accept status, indicating that the order has been accepted and will be processed asynchronously. The payment gateway then submits the payment order to the respective external payment processor.

In the investment allocation settlement phase (Figure 6.13), on receipt of a webhook by the payment processor that the funds have been successfully transferred to the project developer's account, the payment gateway notifies the FundChain backend with an appropriate webhook.

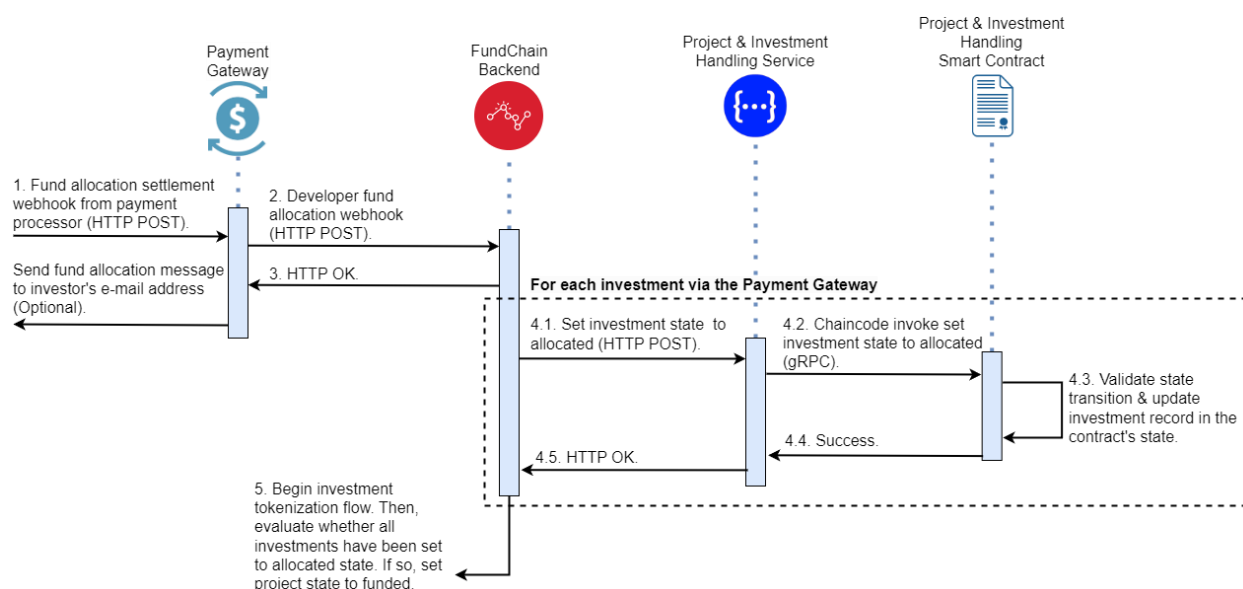


Figure 6.13: Sequence diagram illustrating the actors and the steps involved in the investment allocation settlement phase.

For each investment whose funds were committed via the payment gateway, the FundChain backend sets the investment's state to "Allocated" by issuing an HTTP POST request to the Project & Investment Handling service. The Project & Investment Handling service invokes the appropriate function of the smart contract, which validates the state transition and updates its state accordingly. A successful response is propagated all the way back to the FundChain backend, which initiates the investment's tokenization flow. Finally, the FundChain backend evaluates whether all investments have been set to "Allocated" state and, if so, proceeds to mark the project as "Funded".

6.4.4.2 Stable Coin Payment - Investment Allocation

When investment funds are committed via the platform's stable coin construction, the process of moving funds to the project developer's account differs slightly from the process used for regular Internet payment schemes. The steps involved in this process are depicted in the sequence diagram below.

The FundChain backend retrieves the project's investments by issuing a HTTP GET request to the Project & Investment Handling service. The Project & Investment Handling

service queries the respective function of the smart contract and an array containing all of the investments of this project is propagated back to the FundChain backend.

For each stable coin investment that is in "Funds Reserved" state, the FundChain backend sets the investment's state to "Allocated" by issuing an HTTP POST request to the Project & Investment Handling service. The Project & Investment Handling service then invokes the appropriate function of the smart contract, which validates the state transition and issues a coin hold execution request to the fungible token smart contract.

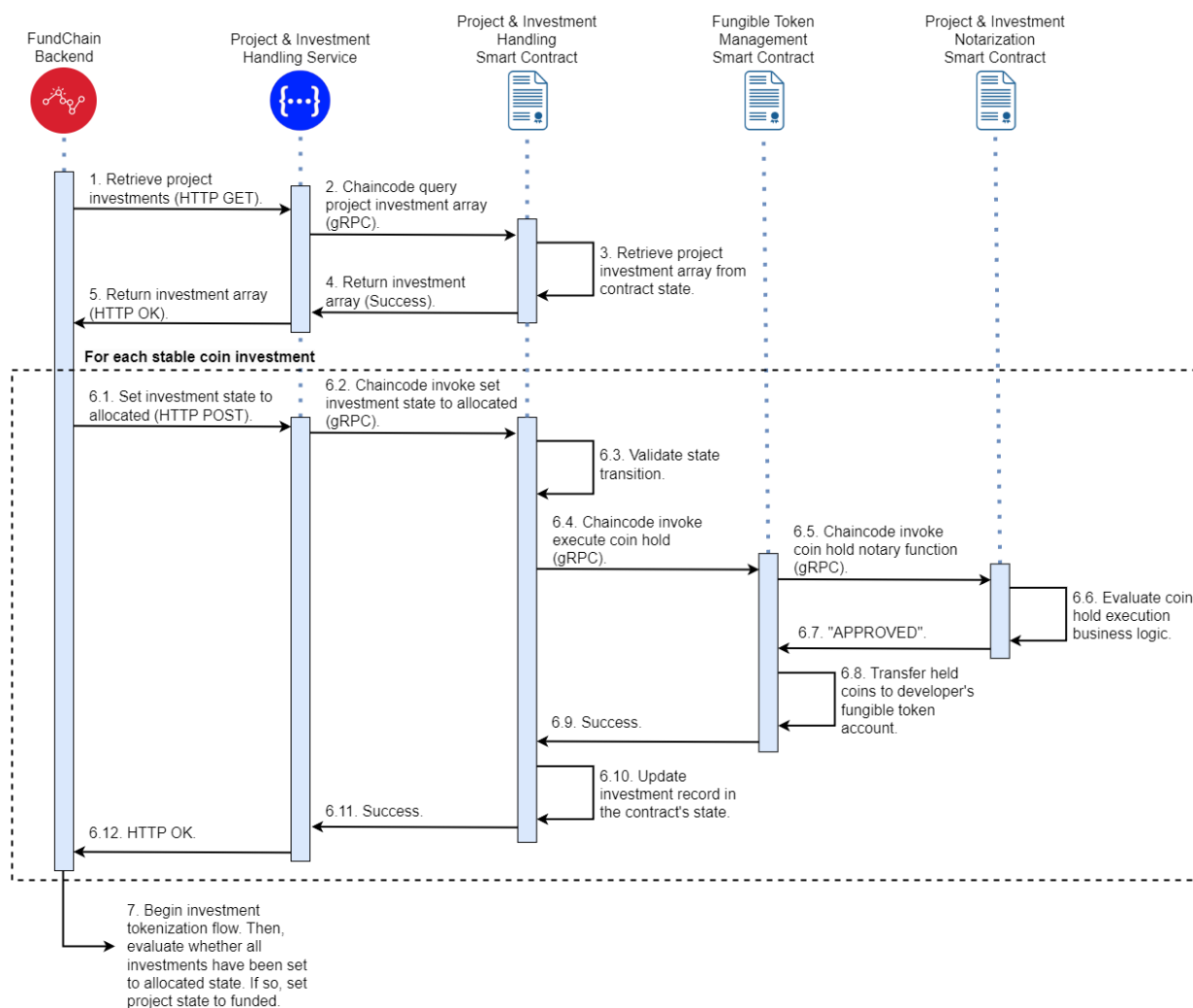


Figure 6.14: Sequence diagram illustrating the actors and the steps involved in transferring stable coin investment funds to the project developer's stable coin account.

The fungible token smart contract then validates the request and invokes the notary of the coin hold, which is the Project & Investment Notarization smart contract. The Project & Investment Notarization smart contract evaluates the business logic associated with the execution of the coin hold and responds with an approval status. Once the coin hold is approved, the fungible token smart contract executes the coin hold, crediting the coins to the project developer's fungible token account, and successfully returns the call stack to the Project & Investment Handling smart contract. The latter smart contract updates

the state of the stable coin investment to "Allocated," and a successful response is propagated all the way back to the FundChain backend.

Finally, the FundChain backend initiates the investment's tokenization flow, which is described in a future section. After all stable coin investments have been set to "Allocated" state, the FundChain backend evaluates whether all investments have been set to "Allocated" state and, if so, proceeds to mark the project as "Funded".

6.4.5 Investment Tokenization

After all investments have been successfully allocated to the project, the FundChain platform initiates the investment tokenization flow (Figure 6.15). This process involves creating a unique and tradable investment NFT (Non-Fungible Token) for each allocated investment. The NFT represents the investor's ownership of a specific portion of the project. The first step of the tokenization process is initiated by the FundChain backend, which sends an HTTP POST request to the Investment Tokenization Service, requesting the creation of an investment NFT for each investment that is in "Allocated" state.

On receipt of the request, the Investment Tokenization Service invokes the mint function of the respective smart contract. The Investment Tokenization smart contract is responsible for creating and managing the NFTs for the investments. When invoked, it creates a new NFT and assigns the investor's ownership of the investment to it.

To ensure that the minting process is secure and meets the business logic requirements, the Investment Tokenization smart contract sends a request to the notary smart contract to approve the mint request. The notary smart contract is responsible for verifying the authenticity of the request, checking the investor's balance, and validating that the requested investment NFT is unique.

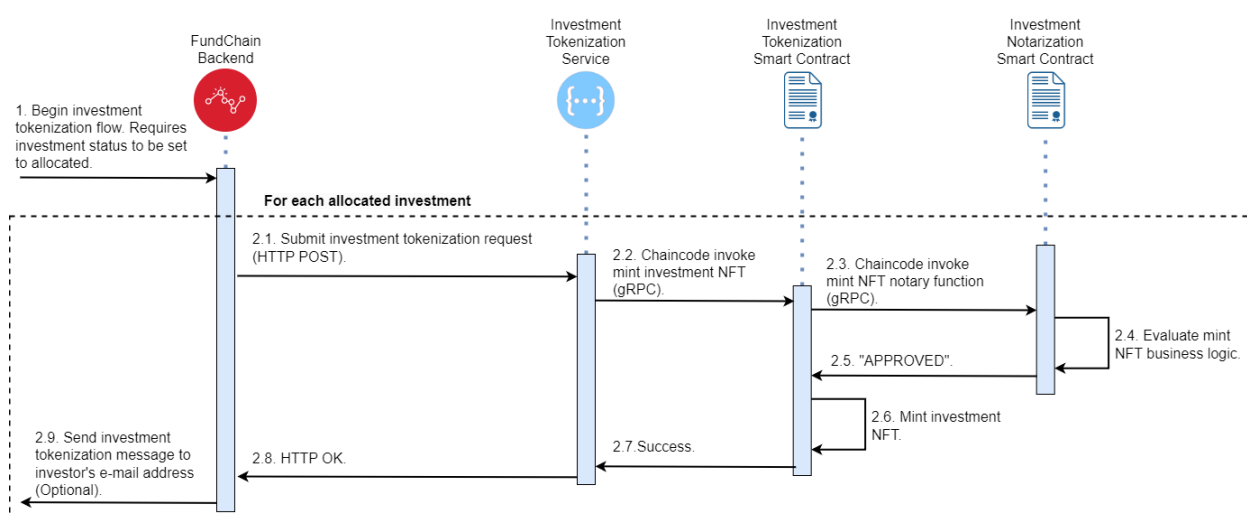


Figure 6.15: Sequence diagram illustrating the actors and the steps involved in the tokenization of investments.

Once the notary smart contract approves the request, the Investment Tokenization smart contract mints the investment NFT and a successful response is propagated to the FundChain backend. At this point, the FundChain backend may send a notification to the investor's e-mail address, informing them of their newly minted investment NFT. The

tokenization process ensures that each investor can now trade their investment NFT with other investors on any NFT marketplace, providing greater liquidity and flexibility to investors.

6.4.6 Equity Tokenization

Tokenizing equity shares is an important step in the process of investing in a project through FundChain. Once the project is marked as "Completed", the next step is to tokenize the equity shares (Figure 6.16).

For each investment that is in the "Allocated" state, the FundChain backend initiates the process by requesting the minting of an equity NFT. This is done by issuing an HTTP POST request to the Equity Tokenization Service. The Equity Tokenization Service then invokes the mint function of the respective smart contract.

Upon receipt of the request, the Equity Tokenization smart contract then invokes the notary's function regarding mint requests. The notary smart contract evaluates the business logic associated with the minting of equity NFTs and approves the request. Once the request is approved, the Equity Tokenization smart contract mints the equity NFT and propagates a successful response all the way back to the FundChain backend.

As a final step, the FundChain may send a message to the investor's e-mail address, notifying them about their newly minted equity NFT. This ensures that investors are kept informed and up-to-date with their investments' outcomes.

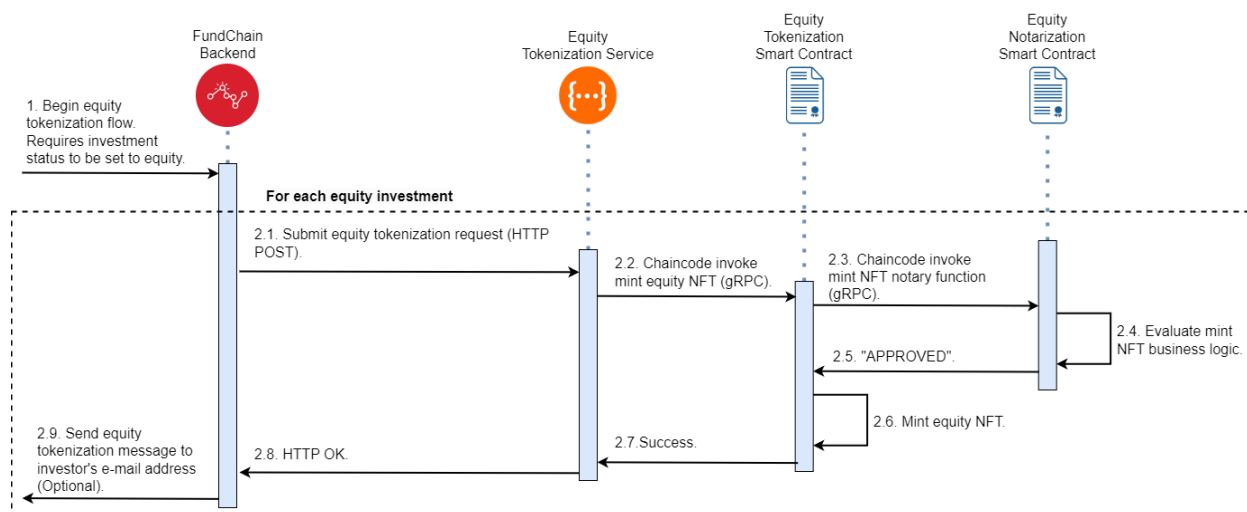


Figure 6.16: Sequence diagram illustrating the actors and the steps involved in the tokenization of equity shares.

7 Equity Crowdfunding Cloud Services

One of the main advantages of using a blockchain-based equity crowdfunding platform is the ability to leverage off-chain cloud services that can enhance the overall user experience and streamline important platform operations. These off-chain services are designed to complement the inherent features of the blockchain technology and provide additional functionality that is essential for running a successful equity crowdfunding platform.

The identity and access management (IAM) service is a critical component of our platform's security infrastructure. It allows for secure user authentication and access control, ensuring that only authorized users can access the platform. This is particularly important for equity crowdfunding platforms, as investment transactions involve sensitive financial information and require strict security measures to protect against fraud and unauthorized access.

The Hyperledger Fabric transaction signing manager is another important off-chain cloud service that is integrated into our platform. It facilitates the efficient signing and management of blockchain transactions, which is essential for maintaining the integrity and accuracy of the equity crowdfunding process. With this service, users can sign transactions with ease and speed, without having to worry about the technical complexities of blockchain technology.

The payment gateway is another critical service that we provide as part of our equity crowdfunding cloud services. This service enables investors to make secure payments using a variety of payment methods, including credit cards, bank transfers, and our innovative stable coin construction. With this service, investors can be confident that their financial transactions are secure and that their investment funds will be transferred quickly and efficiently.

Finally, the FundChain Backend is, on the one hand, responsible for storing and managing off-chain data. It utilizes a scalable NoSQL database, MongoDB, and real-time data processing and analytics technologies, Apache Kafka and Apache Spark, to provide valuable insights into platform operations. The backend is complemented by expressive web APIs that enable efficient querying and modification of the data while adhering to security and privacy requirements. On the other hand, it manages various events related to project and investment lifecycles, such as timers evaluating project funding status and orchestrating processes for refunding investments if projects fail to meet their funding goal.

In conclusion, our equity crowdfunding cloud services provide a comprehensive suite of features that enable investors and project owners to participate in the equity crowdfunding process with confidence and ease. These services are essential for creating a secure, efficient, and transparent equity crowdfunding platform that benefits all stakeholders involved.

Now that we have provided an overview of the equity crowdfunding cloud services offered by our blockchain-based platform, we will explore each service in more detail. In

the following subsections, we will provide technical details and explanations of how each service operates, and how they work together to provide a seamless and secure user experience. By understanding the technical intricacies of each service, readers will gain a better understanding of the benefits and advantages of our equity crowdfunding cloud services, and how they contribute to the success of the proposed platform.

7.1 Identity & Access Management (IAM)

At a high-level, the proposed equity crowdfunding platform consists of two separate sets of services or software components. The first set comprises a variety of off-chain software components that primarily use the REST architectural style for their interactions. The second set is the "blockchain world," which refers to the HLF-based blockchain infrastructure that houses a network of digital agents, including smart contracts.

As blockchains continue to revolutionize various industries, it has become increasingly important to ensure that the security of decentralized systems is maintained while providing access to authorized users. One area that requires careful consideration is identity and access management (IAM), to which we dedicate the content of this section. To ensure interoperability and standardized security, industry-wide standards such as OAuth 2.0, OpenID Connect, and the X.509 digital certificate standard are employed. This enables the IAM service to provide a unified approach for authentication and authorization in a blockchain-based environment. In this context, the IAM service is a crucial component of the platform that helps to manage identities and access control, as well as, providing a streamlined onboarding process for all platform participants.

The IAM service will be based on the Community Management DAPP, a decentralized application built on top of the Hyperledger Fabric blockchain platform that enables, among others, the creation and management of user communities in a decentralized and trustless fashion, which was developed as part of the efforts of the FEVER H2020 project. This component will undergo suitable modifications to fit the current application domain. The remainder of this section is dedicated to presenting an overview of its internal architecture, followed by a description of its main flows, which will provide a comprehensive understanding of how the IAM service will be implemented to address the requirements of the proposed blockchain-based equity crowdfunding platform.

Figure 7.1 presents an architecture diagram that illustrates the various services that together form the Identity & Access Management (IAM) service of the proposed blockchain-based equity crowdfunding platform. The figure provides a high-level overview of the functionalities provided by the IAM's exposed endpoints, as well as, the various services and components that make up the IAM service, and how they interact with one another to provide robust and secure user authentication and authorization

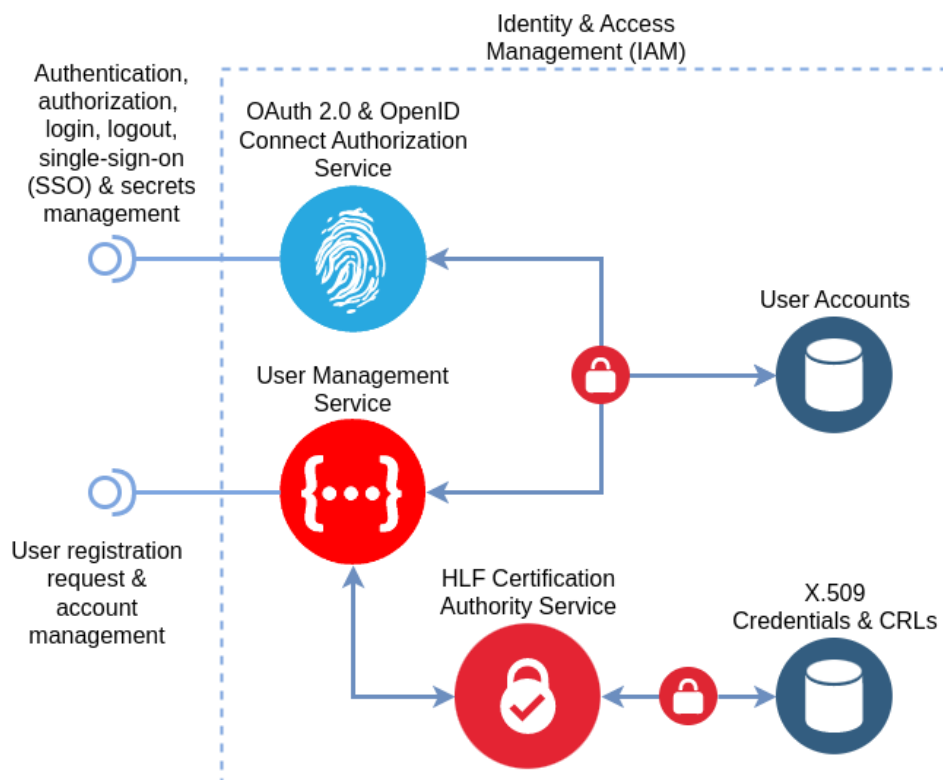


Figure 7.1: Architecture diagram illustrating the collection of services that compose the platform's Identity & Access Management (IAM) service.

The User Management Service is designed to provide a simple and secure mechanism for users to register and manage their accounts. It is responsible for managing the lifecycle of registration requests, from the initial submission of the request to the final approval or rejection of the account by the platform's administrators. It also provides a range of status updates and notifications to users throughout the registration process, keeping them informed of the current state of their requests. Furthermore, it provides a comprehensive set of endpoints to platform administrators, allowing them to manage the platform's membership with a high level of granularity. Through these endpoints, administrators can manage user accounts, approve or reject registration requests, modify user roles and permissions, and perform other tasks related to identity and access management. These endpoints provide a flexible and powerful way for administrators to ensure that only authorized users are granted access to the platform's services and resources.

The authorization service is a crucial component of the IAM that implements security standards such as OAuth 2.0 and OpenID Connect. It serves as a central authority that authenticates user identities and grants access to resources on behalf of users. This service provides a range of features such as single-sign-on (SSO), password management, and user consent management. SSO enables users to log in once and gain access to multiple resources without the need for multiple logins. Password management features allow users to change or reset their passwords, while user consent management enables users to manage the permissions granted to applications.

In conclusion, the Identity & Access Management (IAM) service of the proposed blockchain-based equity crowdfunding platform is a crucial component that ensures the secure and efficient management of user identities and access to platform resources. In the upcoming sections, we will dive into the registration flows for both investors and project developers, outlining the necessary steps and requirements for creating an account and gaining access to the platform's features. Lastly, we stress that we omit details regarding the login, logout and password management flows, as they are addressed by the respective OAuth 2.0 standard specifications (RFC 6749 [23], RFC 6750 [24]).

7.1.1 Investor Registration

The registration process for investors is a crucial step in accessing the investment opportunities offered by the proposed blockchain-based equity crowdfunding platform. As such, this section is dedicated to presenting a comprehensive overview of the steps involved in the registration flow for investors. We delve into the details of the registration process, including the creation of an investor account, the submission of identity verification documents, and the acceptance of the platform's terms and conditions. This will enable potential investors to gain a clear understanding of what is expected of them during the registration process and help them complete the process in a seamless manner.

The following sequence diagram (Figure 7.2) depicts the step-by-step process that an investor goes through to register to the blockchain-based equity crowdfunding platform. The sequence starts with the investor filling in the registration form and accepting the platform's terms and conditions. Once the form is submitted via an HTTP POST request, the User Management service takes over and validates the information provided in the form. It also checks for any conflicts or overlaps with the current state of user accounts to ensure that the registration is unique. If the validation succeeds, the User Management service stores the investor registration request record to its database and requests the HLF Certification Authority service to generate an X.509 digital certificate containing the investor's identity attributes. The Certification Authority service generates the cryptographic material and signs the X.509 digital certificate with MSP signing key. The Certification Authority service stores the generated investor credentials to its respective database and returns HTTP OK. The User Management service creates an activated user record for the investor in its database and returns HTTP OK to the platform's UI. This registration flow ensures that investors can quickly and securely join the platform while providing the necessary identity verification and management.

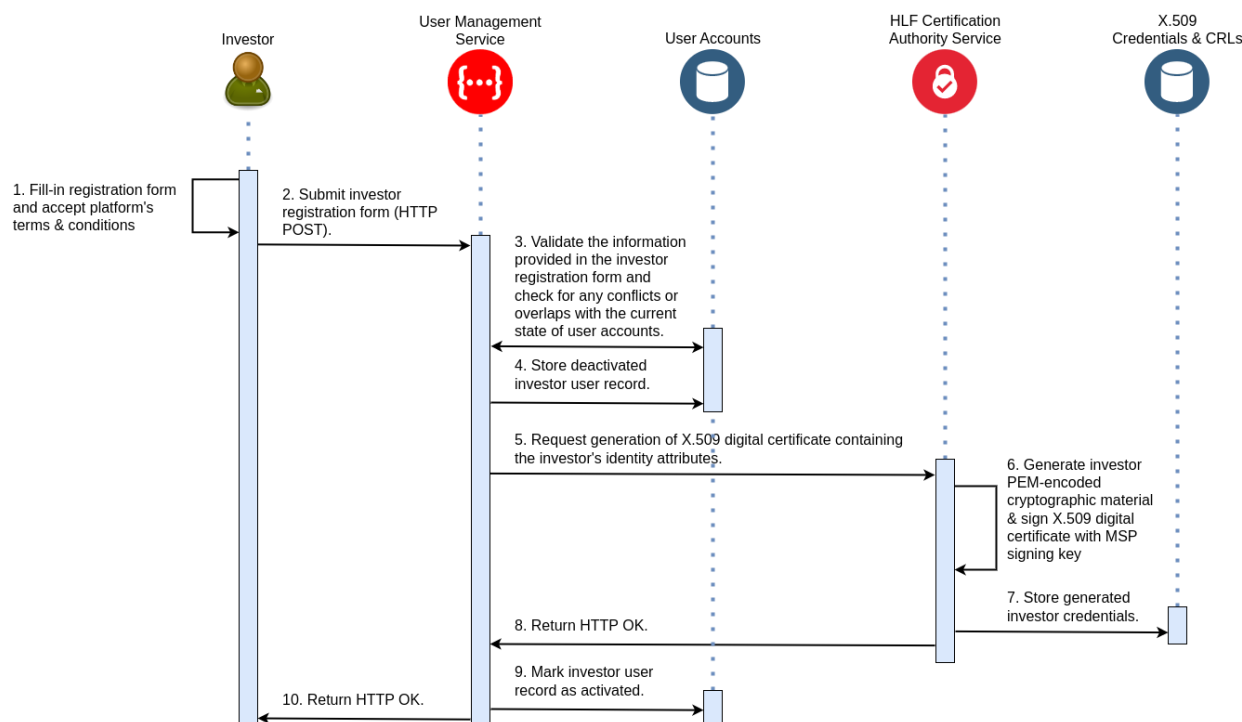


Figure 7.2: Sequence diagram illustrating the registration flow for an investor on the blockchain-based equity crowdfunding platform.

After completing the registration flow depicted in the sequence diagram above, the investor will be able to log in to the platform using her credentials. The registration process ensures that the investor's identity is securely verified and authenticated, and their user account is created in the platform's User Management service, allowing them to participate in the platform's activities.

7.1.2 Project Developer Registration

The process of registration for a project developer is a crucial step towards participating in the proposed blockchain-based equity crowdfunding platform. A successful registration allows the developer to create fundraising campaigns for their projects, which can attract investors to support their ventures. This section provides a comprehensive overview of the steps involved in the registration flow for a project developer, including the required information and documentation, as well as the verification process that must be completed before the developer can start creating campaigns. By following the outlined steps, project developers can quickly and easily join the platform and begin raising funds for their projects.

The sequence diagram presented above (Figure 7.3) shows the steps involved in the first part of the project developer's registration process. It starts with the developer filling in the registration form and attaching any necessary identity-related documents. Once the developer has accepted the platform's terms and conditions, the UI submits the registration form via an HTTP POST request to the User Management service. The User Management service then validates the information provided in the form and checks for any conflicts or overlaps with the current state of user accounts. If everything is in order, the User Management service stores the developer's registration request record in its

database. The developer is then presented with a notification that her account request has been received and that she will receive an email when her account has been approved. This marks the end of the first part of the project developer's on-boarding process.

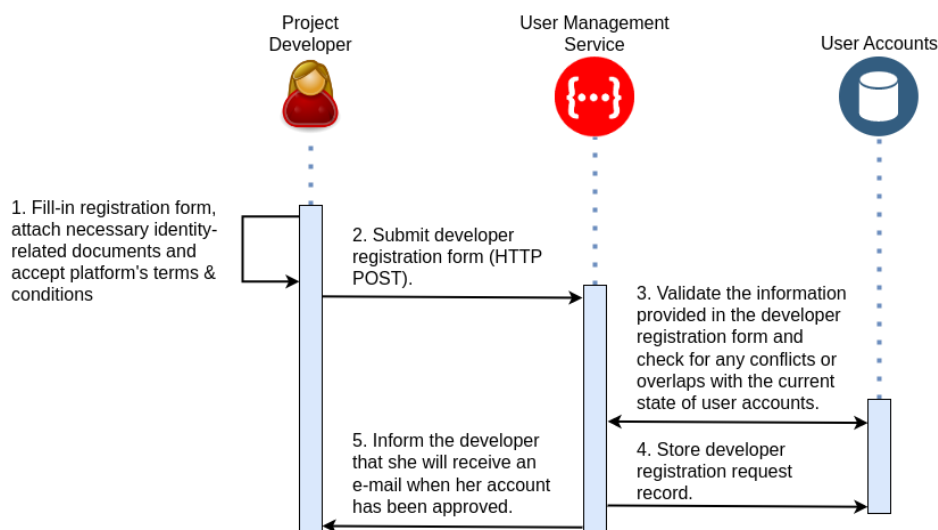


Figure 7.3: Sequence diagram of project developer's registration request submission process to the platform.

The following sequence diagram (Figure 7.4) depicts the user account activation process for a project developer who has submitted their registration request to the platform. After the initial submission of the registration form, the developer's request is stored in the User Management service's database. When the platform administrator logs in and navigates to the registration requests page, they can retrieve the developer's request by sending an HTTP GET request to the User Management service. Once retrieved, the administrator can perform any additional KYC and AML validations required before approving the request by sending an HTTP POST request back to the User Management service.

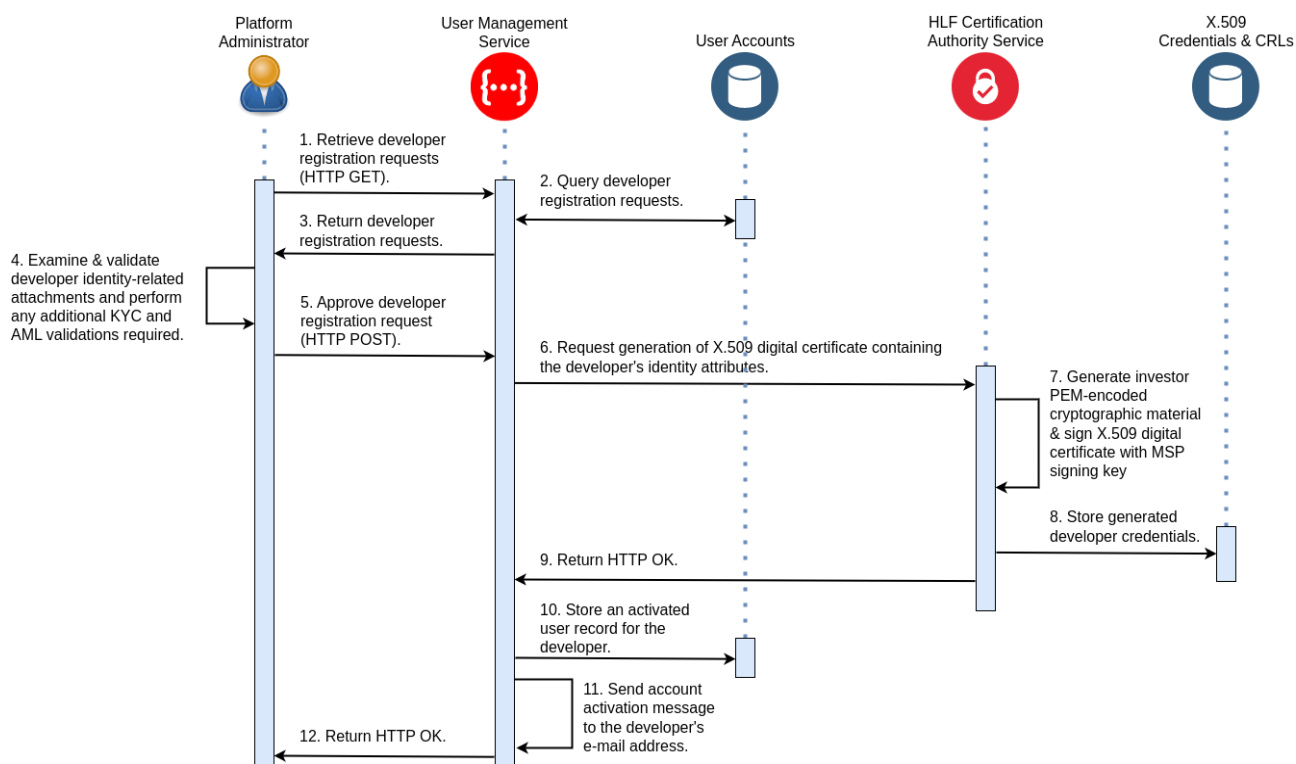


Figure 7.4: Sequence diagram illustrating the activation of a project developer's user account on the platform.

Upon approval, the User Management service requests the generation of a digital certificate containing the developer's identity attributes from the HLF Certification authority service. The HLF Certification authority service then generates cryptographic material and signs the X.509 digital certificate with an MSP signing key. The generated developer credentials are stored in the HLF Certification authority service's database, and the service returns an HTTP OK status to the User Management service.

Next, the User Management service creates an activated user record for the developer in its database and transmits an account activation message to the developer's email address. Once the email is sent, the User Management service returns an HTTP OK status to the platform administrator's UI. At this point, the developer's registration process is complete, and they can log in to the platform and access their account.

7.2 Payment Gateway

The integration of a payment gateway is a crucial part of any digital platform that handles financial transactions. In the case of our platform, it enables the exchange of funds between investors and project developers in a secure and transparent way. In this section, we will delve into the internal architecture of the payment gateway component, highlighting its key features and functionality. We will also explore how it works in conjunction with other platform components to provide a seamless user experience.

The following figure (Figure 7.5) illustrates the typical payment flow in an online store, which involves several key components and steps. The customer adds items to their shopping cart and proceeds to checkout. The checkout page prompts the customer to enter their payment information, such as credit card details or PayPal account information. Once the customer submits the payment information, the payment gateway component securely transmits the payment data to the payment processor, such as PayPal or GooglePay, for authorization.

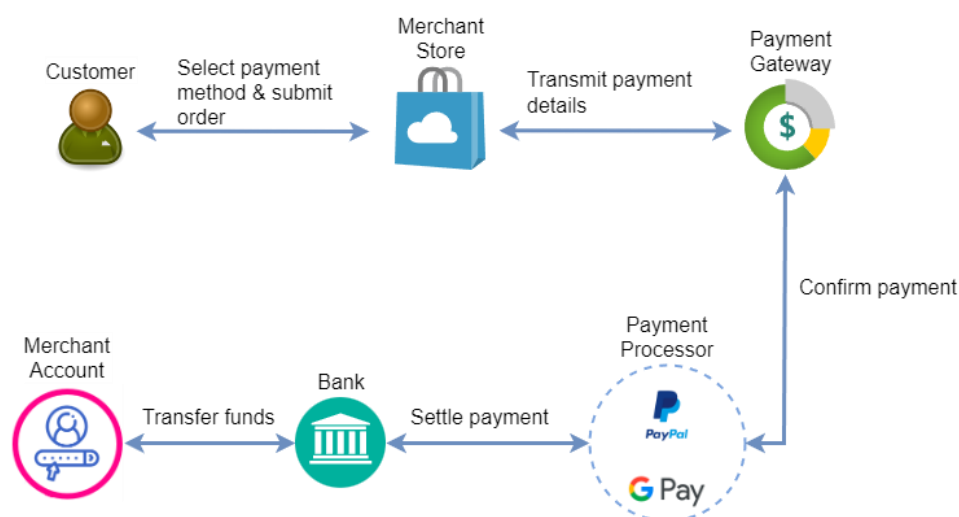


Figure 7.5: Payment flow in an online store; a graphical representation of the steps involved in handling payments via a payment gateway.

The payment processor validates the payment information and communicates the transaction status back to the payment gateway component, which in turn communicates the transaction status back to the online store. If the payment is approved, the payment processor component initiates the transfer of funds from the customer's account to the merchant's account, typically via an intermediary e.g., a bank. Finally, the funds are transferred from the customer's account to the merchant's account, usually via a settlement process that occurs a few days after the initial transaction.

The payment gateway component that is being proposed goes beyond the standard integration with popular internet payment platforms and introduces a novel stable coin construction mechanism that leverages the capabilities of the Hyperledger Fabric network. This component enables, among others, minting and burning of fungible tokens.

Minting and burning are two concepts that are related to the creation and destruction of fungible tokens. Minting refers to the process of creating new tokens and adding them to the token supply. When new tokens are minted, they are added to the total token supply and become available for use. Burning, on the other hand, refers to the process of destroying tokens and removing them from the token supply. This can be done for a number of reasons, such as when tokens are redeemed for a product or service or when they are no longer needed. When tokens are burned, they are removed from the total token supply and are no longer available for use. Together, minting and burning help to regulate the total token supply and ensure that there are always enough tokens available

for use. They also provide a way to control inflation and deflation, as well as to manage the value of the tokens in circulation.

The proposed stable coin construction can be used to facilitate transactions, which has several benefits over traditional online payment systems. First, using a stable coin constructed on top of reputable standards such as ERC-20 allows for cost and fee reduction in online transactions. This is because the use of a stable coin eliminates the need for intermediaries, such as banks, that charge fees for their services. Additionally, since stable coins are issued and redeemed on a blockchain network, the costs associated with fraud prevention and dispute resolution can be greatly reduced. Second, the stable coin construction enables faster settlements in online transactions. With traditional online payment systems, transactions can take several days to settle due to the involvement of multiple intermediaries and their respective processing times. However, using stable coins on a blockchain network allows for near-instantaneous settlement of transactions, greatly reducing the time it takes for funds to be transferred between parties.

Another benefit of using stable coins in online transactions is the increased security and transparency they provide. Since stable coins are issued on a blockchain network, all transactions are recorded on an immutable ledger that is transparent and accessible to all participants. This enhances the security and trustworthiness of online transactions, reducing the risks associated with fraudulent activity.

Finally, stable coin construction provides a mechanism for individuals and businesses to transact in a decentralized, borderless, and permissionless manner. This is because stable coins are not tied to any particular jurisdiction or financial institution, allowing users to transact globally without restrictions or the need for intermediaries.

In summary, the proposed payment gateway component provides significant innovation in the realm of online payments by allowing for the construction of stable coins using the Hyperledger Fabric network. This construction allows for cost and fee reduction, faster settlements, increased security and transparency, and decentralized transactions, providing significant benefits over traditional online payment systems. The payment gateway will provide a reliable payment system for project developers to receive funds and for investors to participate in crowdfunding activities. Overall, the stable coin construction facilitated via the proposed payment gateway will significantly benefit the equity crowdfunding platform and enhance its capacity to provide a secure, reliable and cost-effective service for all stakeholders.

7.2.1 Minting Fungible Tokens

The process of minting coins to a fungible token account is an essential feature of the platform's payment gateway that enables users to transfer their assets through the use of digital currencies. This process involves two phases, the mint request submission phase and the mint request settlement phase.

In the first phase (Figure 7.6), the fungible token account owner submits a mint request to the Fungible Token Service, which then invokes the appropriate function of the

respective smart contract. The smart contract validates and stores an association of the input mint request with the identified fungible token account in its internal state. Once this is done, the fungible token smart contract returns the identifier of the mint request to the Fungible Token Service, and a successful response is propagated back to the account owner.

The account owner then requests a payment authorization code for the newly submitted mint request from the FundChain backend. The backend verifies that the mint request has been successfully published on the ledger, initializes its internal state for this particular mint request, and computes a fresh payment authorization code. This authorization code, along with data regarding the respective payment, is then communicated to the platform's payment gateway, which returns the payment authorization code to the account owner.

The account owner then invokes the payment endpoint of the payment gateway with the newly acquired authorization code. The payment gateway validates the input data provided by the account owner and submits an appropriate payment order to the external payment processor. Once the payment order has been successfully submitted, a successful response is returned to the account owner by the payment gateway. Optionally, the payment gateway may issue a webhook to the FundChain backend to inform the latter that the payment order has been submitted.

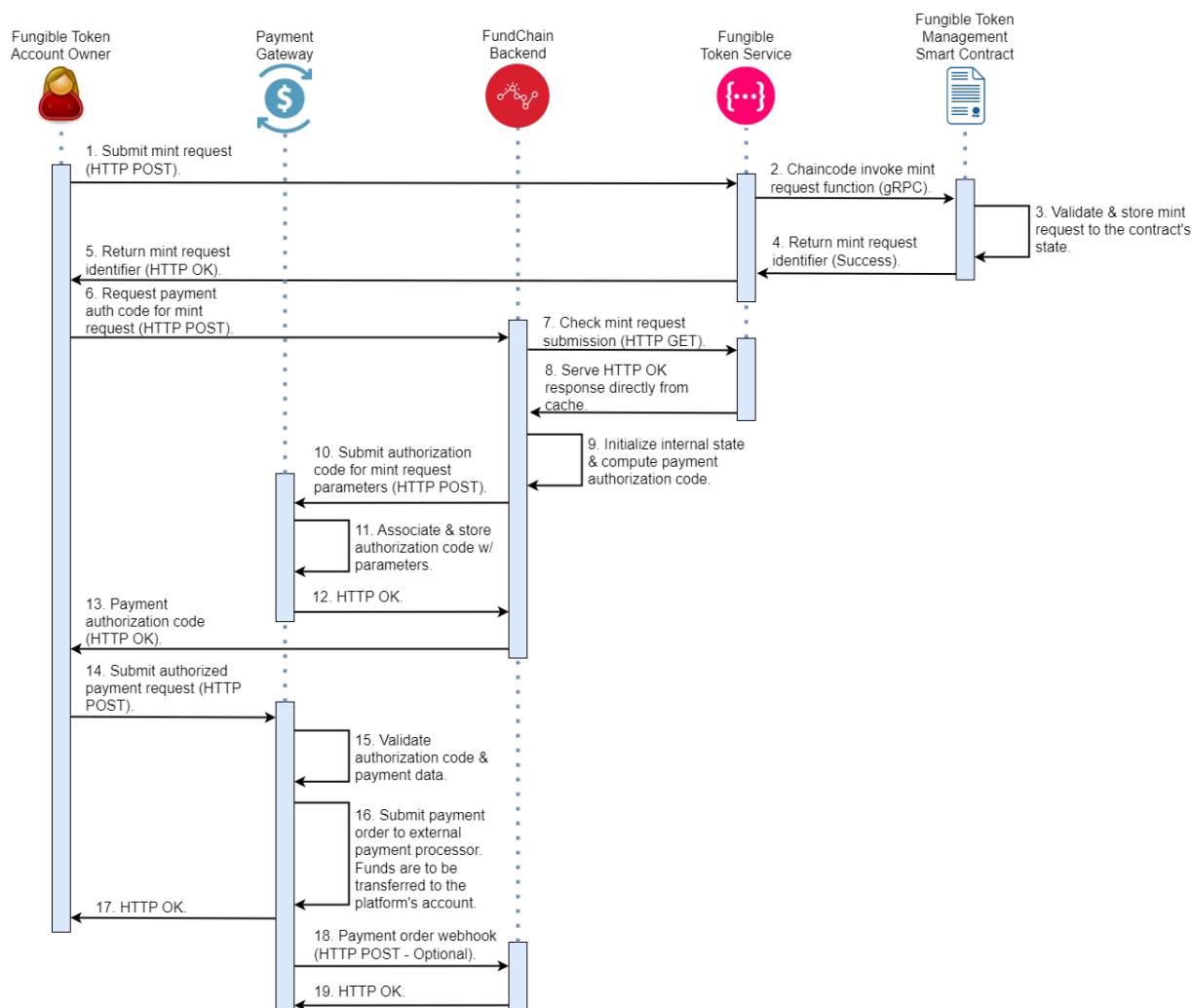


Figure 7.6: Sequence diagram illustrating the actors and the steps involved in the mint request submission phase.

In the second phase, on receipt of a webhook by the payment processor that the payment has been settled, the payment gateway notifies the FundChain backend with an appropriate webhook. The FundChain backend then issues an HTTP POST request to the Fungible Token Service requesting the execution of the mint request. The Fungible Token Service then invokes the mint execution function of the fungible token smart contract. The fungible token smart contract, in turn, invokes the mint execution notary function of its respective notarization smart contract, which evaluates the business logic regarding the mint request.

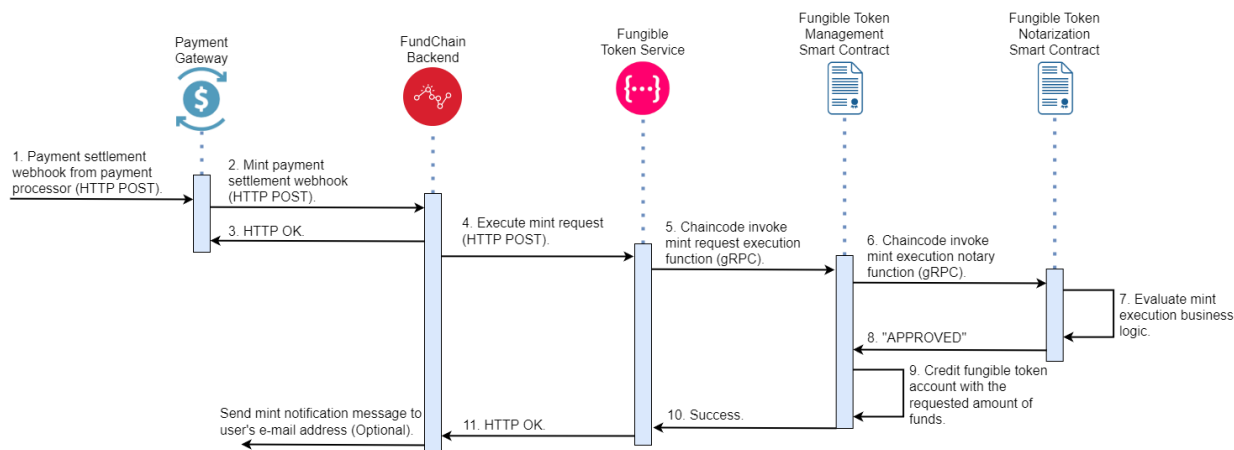


Figure 7.7: Sequence diagram illustrating the actors and the steps involved in the mint request settlement phase.

Once the execution of the mint request is approved, the fungible token smart contract credits the designated account with the requested amount of coins, and a successful response is propagated back to the FundChain backend.

7.2.2 Burning Fungible Tokens

This section is dedicated to describing the process of burning coins from a fungible token account, which can be accomplished through the platform's payment gateway. Similar to the minting process, this process also involves two phases, with each phase having distinct steps, as shown in a separate sequence diagram.

The first phase, known as the burn request submission phase, starts with the fungible token account owner submitting a burn request to the Fungible Token Service. The smart contract then validates the request, locks the designated number of coins from the account's active balance, and stores the request's association with the fungible token account in its internal state. After successful validation, the fungible token smart contract returns the identifier of the burn request to the Fungible Token Service, which then propagates a successful response back to the account owner.

Next, the account owner requests a payment authorization code for the newly submitted burn request from the FundChain backend. The FundChain backend verifies that the burn request has been published successfully on the ledger and initializes its internal state for this specific burn request. The backend then computes a new payment authorization code along with the relevant payment data, which is sent to the platform's

payment gateway. The payment gateway then returns the payment authorization code to the account owner.

The account owner invokes the payment endpoint of the payment gateway using the new authorization code acquired from the FundChain backend. The payment gateway validates the input data provided by the account owner and submits an appropriate payment order to the external payment processor. If the payment is successful, the payment gateway returns a successful response to the account owner. Optionally, the payment gateway may also send a webhook to the FundChain backend, informing it that the payment order has been submitted.

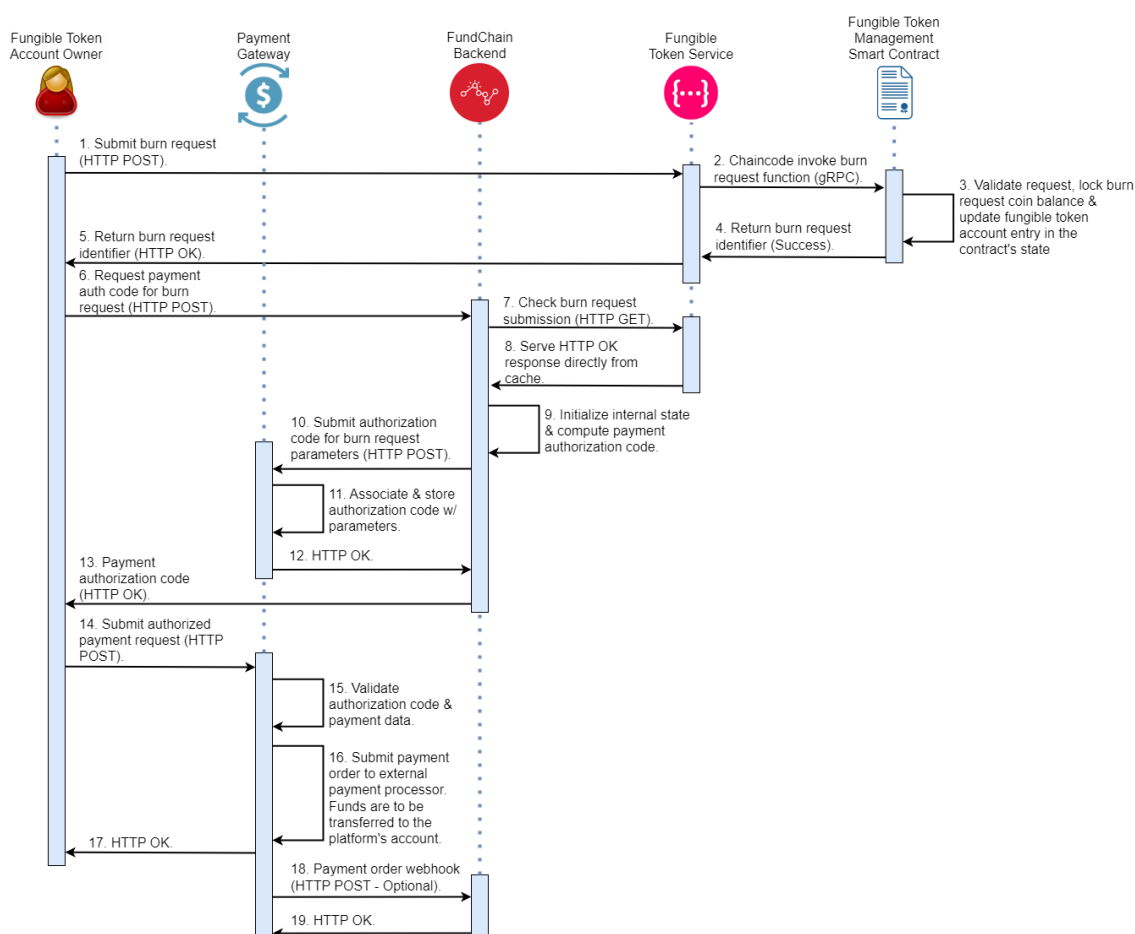


Figure 7.8: Sequence diagram illustrating the actors and the steps involved in the burn request submission phase.

The second phase of the burning process, known as the burn request settlement phase, starts with the payment gateway notifying the FundChain backend through a webhook that the payment has been settled. The FundChain backend then issues an HTTP POST request to the Fungible Token Service requesting the execution of the burn request. The Fungible Token Service then invokes the burn execution function of the fungible token smart contract. Next, the smart contract invokes the burn execution notary function of its respective notarization smart contract, which evaluates the business logic regarding the burn request. If the burn request is approved, the fungible token smart contract credits the designated account with the requested amount of coins.

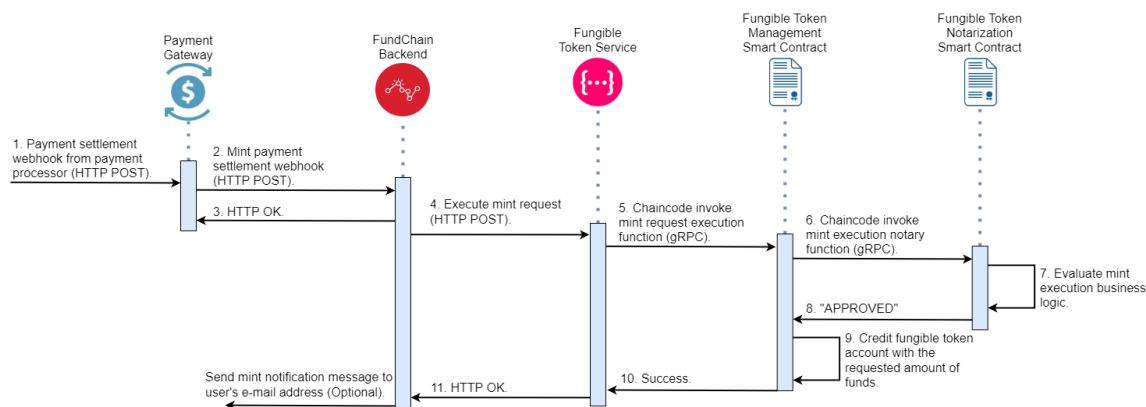


Figure 7.9: Sequence diagram illustrating the actors and the steps involved in the burn request settlement phase.

Finally, a successful response is propagated back to the FundChain backend, which may optionally send a message to the account owner's email address informing them about the successful burning of coins from their fungible token account.

8 Smart Contracts

Smart contracts are self-executing contracts with the terms of the agreement between involved parties being directly written into lines of code. They are programmed to execute automatically when certain predetermined conditions are met. Smart contracts have revolutionized the way transactions are conducted by providing a secure, transparent and efficient way of exchanging assets.

Hyperledger Fabric is an open-source blockchain platform that provides a modular and scalable infrastructure for building distributed ledger applications. One of the key features of Hyperledger Fabric is its support for smart contracts.

In Hyperledger Fabric, smart contracts are called "chaincodes" (we use the terms interchangeably), which is a piece of code written in a programming language such as Go, our language of choice, or JavaScript. A chaincode is deployed on the network as a separate container that runs on each peer node, providing the necessary functionality for the applications to interact with the blockchain.

One of the main benefits of using chaincodes is their flexibility. It allows developers to write business logic in a modular way and update the logic without having to modify the underlying blockchain protocol. This is achieved by separating the business logic from the core blockchain logic, which is responsible for maintaining the integrity of the ledger. Chaincodes in Hyperledger Fabric are executed in a container called the "chaincode container". The container is isolated from the rest of the system and runs in a secure sandbox environment. This ensures that the chaincode is executed in a secure and deterministic way, without being influenced by other components of the system.

The proposed blockchain-based equity crowdfunding platform involves a set of smart contracts that are designed to handle the entire lifecycle of crowdfunding projects and their respective investments, as well as to provide for fungible and non-fungible token functionality. In the following, we dedicate separate subsections to present the specifications of each smart contract involved in the proposed blockchain-based equity crowdfunding platform. The smart contracts are tailored to meet the specific requirements of the platform, such as encoding and evaluating the policies and conditions that govern state transitions of the finite state machines of crowdfunding projects and their respective investments. These smart contracts will enable secure and transparent crowdfunding activities, while also facilitating the issuance and verifiable ownership of equity and investments on the blockchain. In regards to the means under which the technical specifications of the smart contracts are presented in the following subsections, we note the following important points:

1. All smart contract functions receive an implicit argument, which is the X.509 digital certificate of the entity that invokes the function.
2. A response marked as "n/a", i.e., not applicable, entails that the function returns an empty value. This should not be mistaken with erroneous returns

More information about the specifications of a smart contract that provides fungible token functionality on Hyperledger Fabric is provided in Annex II: Smart Contracts Specifications.

9 IANOS Equity Crowdfunding platform

In this section, a glance at IANOS FundChain platform is going to be presented. The most basic pages of the platform are going to be presented in the following chapters with a short description.

9.1 Platform's structure

In the following figure, the main structure of IANOS FundChain platform is presented. Each user will be able to easily navigate and have access to the pages containing all the aforementioned features of the platform. More specifically, the different main pages that are going to be accessible are: Investment Opportunities, My Investments, My Projects, My Wallet, Notifications, and Account.

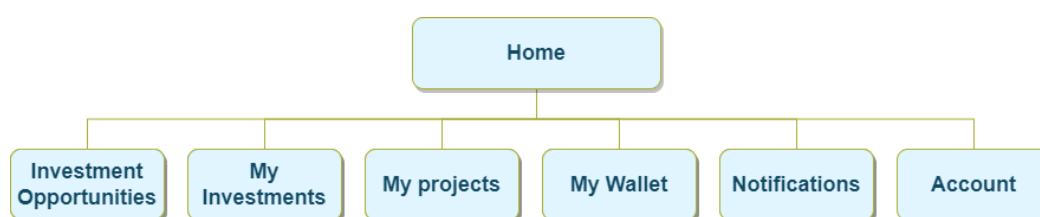


Figure 9.1: Platform's structure

An example of the structure can be observed by the sidebar menu for each actor presented in Figure 9.2. Based on the actors' descriptions each of them have different functionalities that can be observed in the sidebar menus.

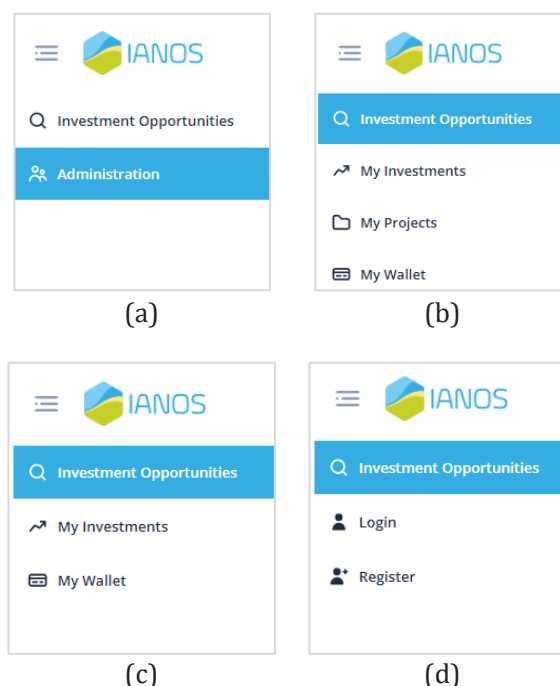


Figure 9.2: IANOS FundChain platform's sidebar menus for: (a) Administrators, (b) Developers, (c) Investors, (d) Visitors

9.2 Login & Registration page

The first page that the user encounters is the registration page. In the registration page, the user is able to securely register into the platform utilizing e-mail and password registration. As already mentioned, the registration phase differs depending on the actor. For an investor, the required fields contain the user's first and last name and obviously the e-mail and password. An example of the registration page is provided in Figure 9.3. For a developer there are much more information required. More specifically, the users need to provide their social security number, and their country, state, and city of residence, as well as a valid phone number. An example of the registration page for a potential developer can be observed in Figure 9.4.

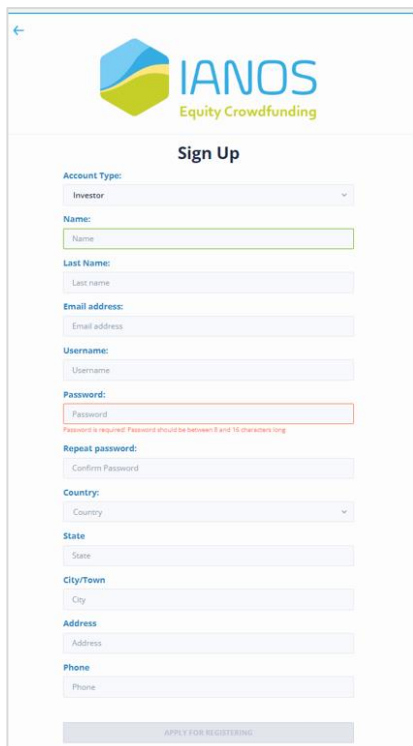


Figure 9.3: Platform's Registration Page for Investors

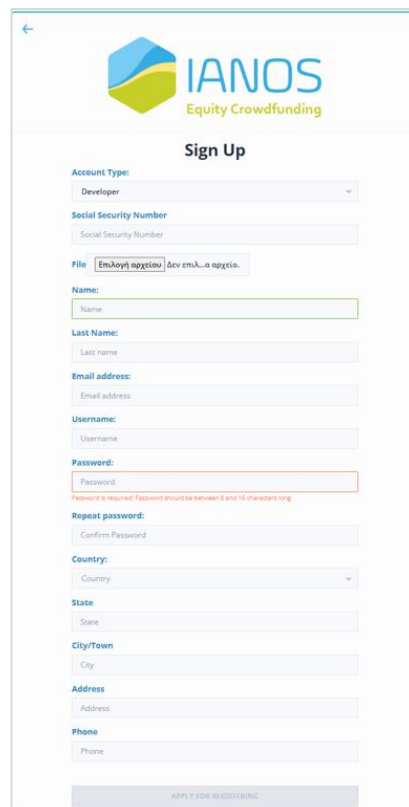


Figure 9.4: Platform's Registration Page for Developers

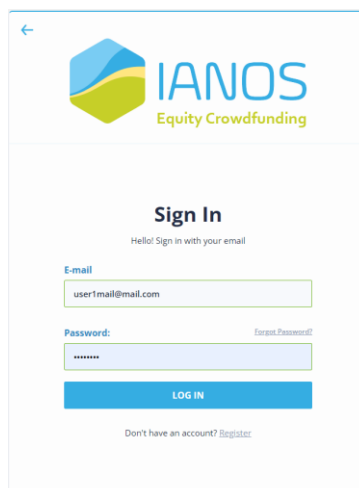


Figure 9.5: Platform's Login Page

After they are registered in the platform, the users are able to login via the Log In page using their email and password. The user can pick whether the platform will be able to remember the credential and automatically let them log in, as shown in Figure 9.5.

9.3 Account page

After the registration or login, the user is able to navigate to their account page in order to fill the information that they are willing to display on their social profile page. In the

account page, the details that the user is able to update are the fundamental user data obtained during the registration process, as well as the city of residence, and the user's interests, in the form of predefined tags, that could enable in matching the user with projects that are close to their interests. Additionally, in this page the users are able to connect with their IANOS prosumer wallet with their email and password. Finally, the platform gives the user the ability to change his or her password in this page. An example of the account page is shown in Figure 9.6.

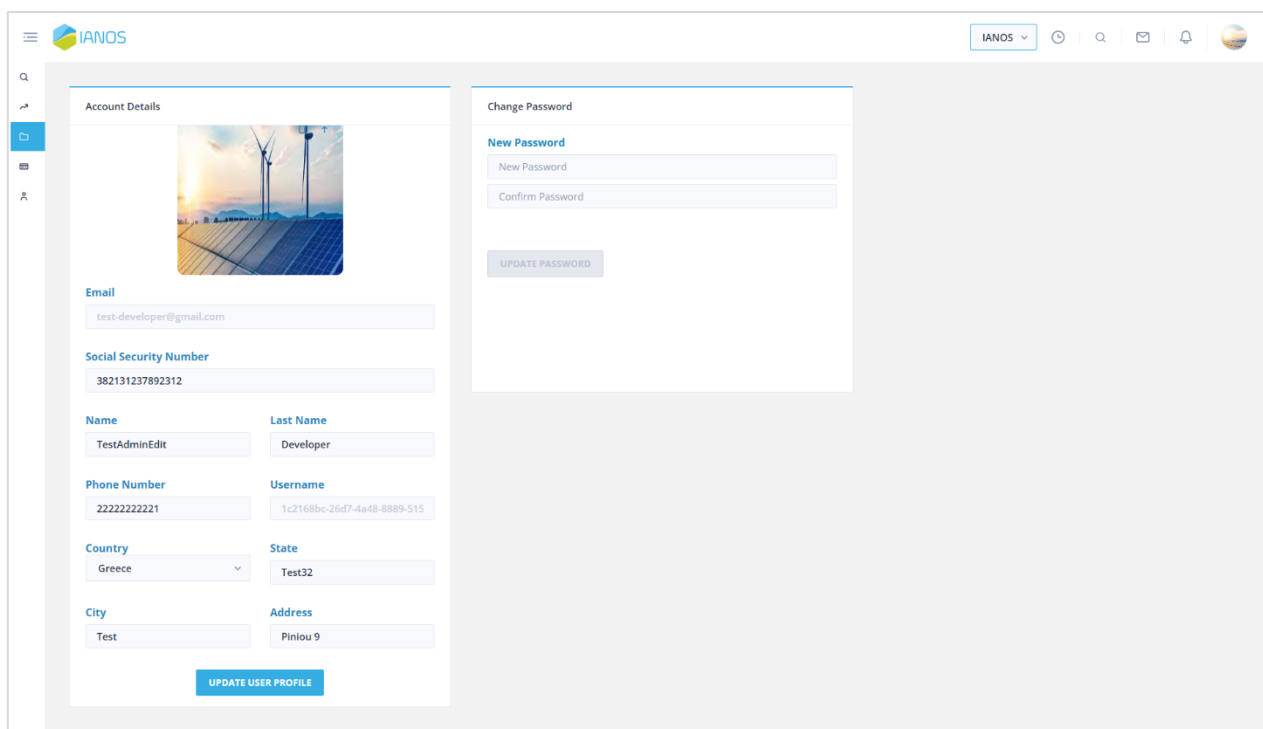


Figure 9.6: Platform's Account Page

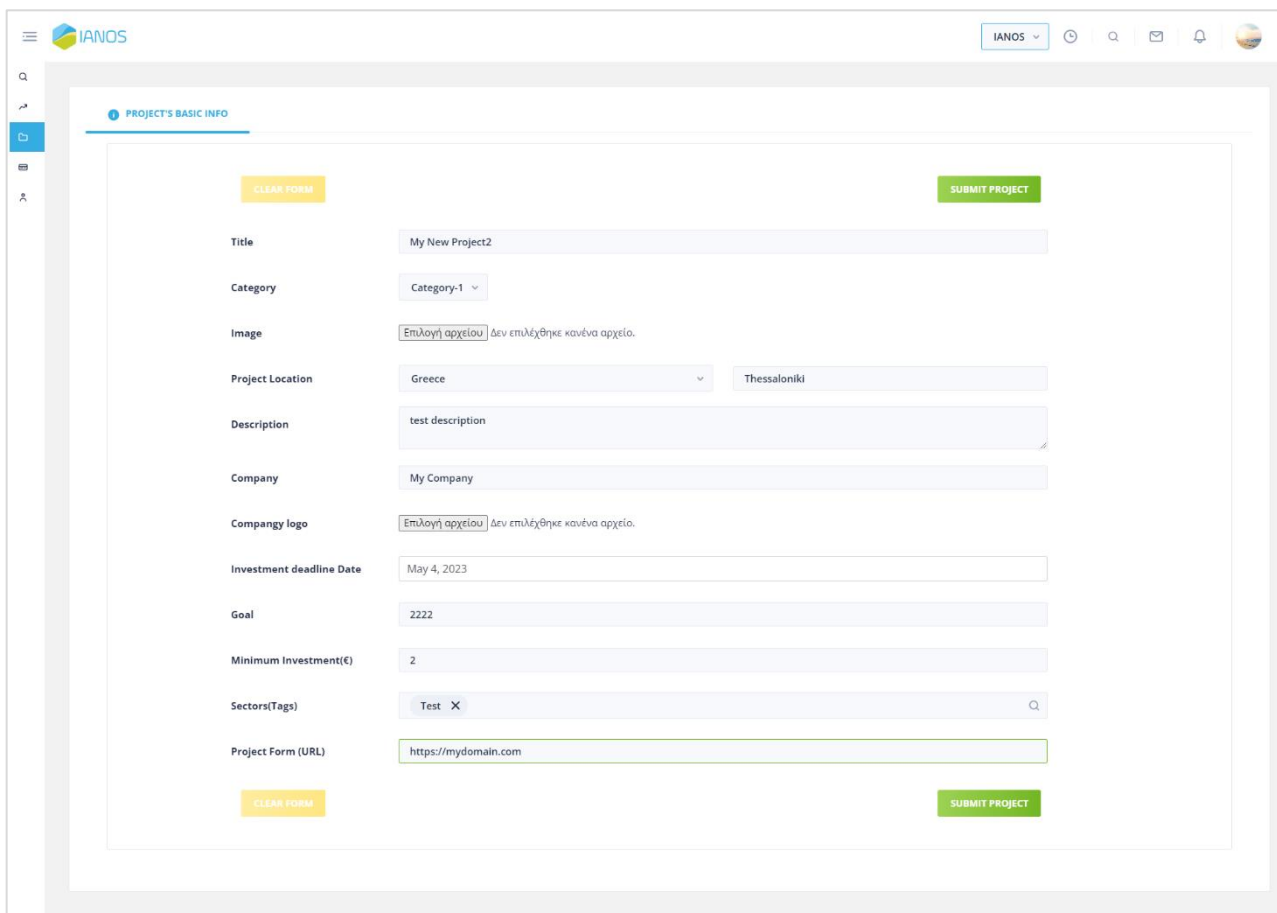
9.4 Project management page

Through the “My Projects” page, the user have the ability to create a new project available as a future investment opportunity. The fields that are required to be filled by the user in the project management page are the following:

- **Title.** It corresponds to the title of the project. The title needs to be descriptive, but compact and not too long, to properly describe the project.
- **Category.** The category describes the type of the project and may vary from PV systems, wind farms and more.
- **Project location.** The location where the new project will be positioned.
- **Description.** A complete and detailed description of the new project.
- **Managing company name.** The company or individual that is going to manage the newly created project.
- **Investment deadline date/time.** The date and time that corresponds to the deadline of the investment of the project.
- **Investment goal (€).** The financial objective in Euro (€) that the creator sets for their future renewable energy investment.

- **Minimum investment (€).** The minimum funds in Euro (€) that a potential investor is able to submit.
- **Sector (tags).** The sector field corresponds to tags will enable the matching of the project to the user's interests.

Furthermore, the creator will have the ability to complete some additional fields, namely upload an image for the project, and attach description and media files and the managing company logo. In Figure 9.7 there is an example of the project management page.



The screenshot shows the 'PROJECT'S BASIC INFO' form in the IANOS platform. The form includes the following fields and values:

- Title:** My New Project2
- Category:** Category-1
- Image:** Επιλογή αρχείου | Δεν επιλέχθηκε κανένα αρχείο.
- Project Location:** Greece (dropdown), Thessaloniki (text input)
- Description:** test description
- Company:** My Company
- Company logo:** Επιλογή αρχείου | Δεν επιλέχθηκε κανένα αρχείο.
- Investment deadline Date:** May 4, 2023
- Goal:** 2222
- Minimum Investment(€):** 2
- Sectors(Tags):** Test X
- Project Form (URL):** https://mydomain.com

The form has 'CLEAR FORM' and 'SUBMIT PROJECT' buttons at the top and bottom.

Figure 9.7: Platform's Project Management Page

9.5 Investment opportunities page

The investment opportunities page contains all the future projects that a potential investor is able to fund. An example of the layout of the page is provided in Figure 9.8. Based on the user's interests, they are matched with projects with similar tags. The user can share the project in order for the project to receive more funding. In addition, a percentage graphic is included to show the progress of the overall funding goal.

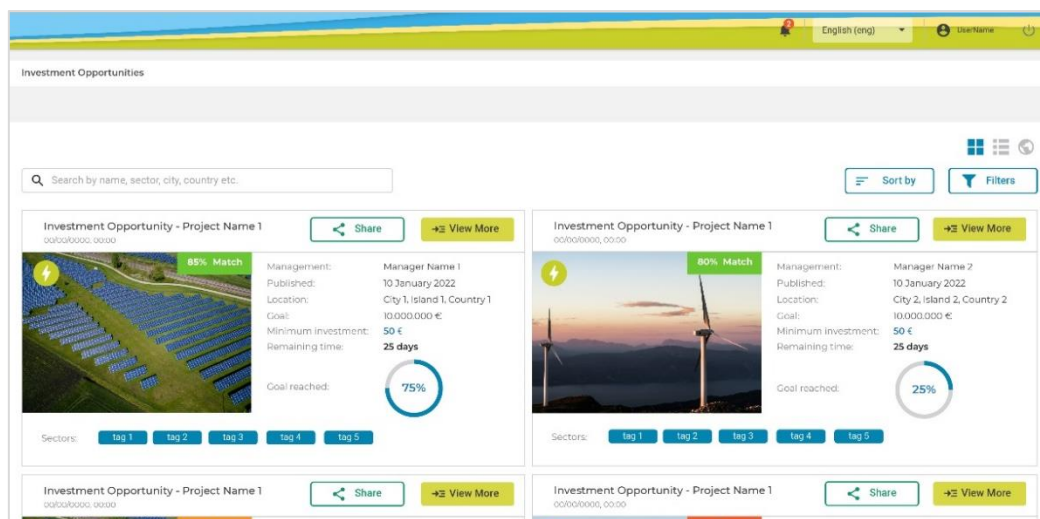


Figure 9.8: Platform's Investment Opportunities Page

9.6 Investment page

When clicking in a future project in the investment opportunities page the users find themselves on the investment page where the detailed information about the project is stored. In this page, more data that describe the potential investment opportunity are present. Namely, the user has access to the full description of the platform, together with other media, such as pictures or video. Finally, the potential investor can view the project's popularity and investor timeline. An example of the investment page is presented in Figure 9.9.

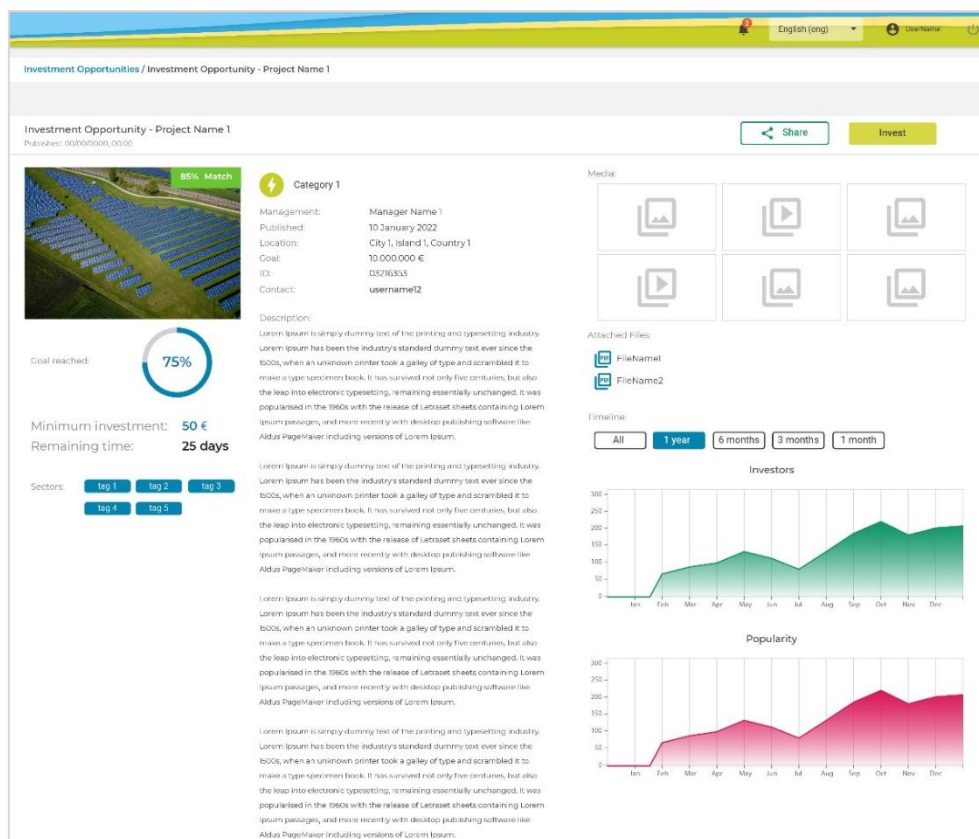


Figure 9.9: Platform's Investment Page

10 Conclusions

The current document has provided a detailed insight of IANOS FundChain platform, which corresponds to the second and final version of the platform. This document offers a comprehensive analysis and study of crowdfunding platforms for renewable energy projects. It includes an introduction and related work section that covers crowdfunding models, motivation, and analysis of existing European crowdfunding platforms for renewable energy projects. The platform analysis section provides a detailed analysis of the platform's general description, features, actors, and user scenarios, along with identification of functional and non-functional requirements. The technical study section discusses the programming languages and frameworks used in the platform's front-end and back-end development. The document also delves into the platform's architecture, including notarizing smart contract operations, finite state machines for projects and investments, and the core flow for project creation, investment, funding status, and tokenization. The equity crowdfunding cloud services section covers identity and access management, investor and project developer registration, and payment gateway for minting and burning fungible tokens. Finally, the document introduces the IANOS FundChain platform, discussing its structure, login and registration page, account page, project management page, and investment opportunities page.

In conclusion, this document serves as a comprehensive resource providing an overview of crowdfunding platforms for renewable energy projects, encompassing features, requirements, technical aspects, and architecture. It offers valuable insights into the complexities and considerations involved in the development and management of such platforms.

Annex I: Equity Crowdfunding Cloud Services

Reference APIs

Identity & Access Management (IAM) Reference API

In this section, we present a set of reference endpoint specifications for the Identity and Access Management service. These endpoints are designed to provide the necessary functionality for platform administrators, investors, and project developers to effectively manage their accounts and permissions within the platform. The endpoints specifications are provided below in a Swagger-like fashion.

POST	/api/v1/users/investors
Creates a new investor user.	
Parameters	
No parameters.	
Request body	
<pre>{ "firstName": "Alan", "lastName": "Turing", "email": "at@acm.com", "password": "oracle", "address": { "street": "Sackville Park, Fairfield St", "city": "Manchester", "state": "Greater Manchester" }, "country": "United Kingdom", "phoneno": "+441632960416" }</pre>	
Responses	
Code	Description
201	Created investor user response.
	Example Value
	<pre>{ "id": "9ba2ee96-7e24-4dbd-b01a-8537d31e7628", "firstName": "Alan", "lastName": "Turing", "email": "at@acm.com", "password": "oracle", "role": "investor", "address": { "street": "Sackville Park, Fairfield St", "city": "Manchester", "state": "Greater Manchester" } }</pre>


```

    },
    "country": "United Kingdom",
    "phoneno": "+441632960416",
    "createdAt": "2023-03-12T08:35:53Z",
    "updatedAt": "2023-03-12T08:35:53Z"
  }
}

```

POST

/api/v1/users/developers

Submit a registration request for a new developer user.

Parameters

No parameters.

Request body

```

{
  "firstName": "Alan",
  "lastName": "Turing",
  "email": "at@acm.com",
  "password": "oracle",
  "address": {
    "street": "Sackville Park, Fairfield St",
    "city": "Manchester",
    "state": "Greater Manchester"
  },
  "country": "United Kingdom",
  "phoneno": "+441632960416",
  "ssn": "6735623542345",
  "govid": {
    "type": "image/png",
    "data": "base64"
  }
}

```

Responses

Code

Description

201

Successful developer user registration submission.

Example Value

```

{
  "id": "9ba2ee96-7e24-4dbd-b01a-8537d31e7628",
  "firstName": "Alan",
  "lastName": "Turing",
  "email": "at@acm.com",
  "password": "oracle",
  "role": "developer",
  "address": {
    "street": "Sackville Park, Fairfield St",
    "city": "Manchester",
    "state": "Greater Manchester"
  },
  "country": "United Kingdom",
  "phoneno": "+441632960416",
  "ssn": "6735623542345",
  "govid": {
    "type": "image/png",

```

```

    "data": "base64"
  },
  "createdAt": "2023-03-12T08:35:53Z",
  "updatedAt": "2023-03-12T08:35:53Z"
}

```

POST

/api/v1/users/developers/{id}/approve

Approve a previously submitted developer user registration request.

Parameters

Name	Description
id <small>*required</small> string <i>(path)</i>	ID of the developer to approve.

Responses

Code	Description
200	OK response.
	Example Value <pre> { "id": "9ba2ee96-7e24-4dbd-b01a-8537d31e7628", "firstName": "Alan", "lastName": "Turing", "email": "at@acm.com", "password": "oracle", "role": "developer", "address": { "street": "Sackville Park, Fairfield St", "city": "Manchester", "state": "Greater Manchester" }, "country": "United Kingdom", "phoneno": "+441632960416", "ssn": "6735623542345", "govid": { "type": "image/png", "data": "base64" }, "createdAt": "2023-03-12T08:35:53Z", "updatedAt": "2023-03-12T08:35:53Z" } </pre>

POST

/api/v1/users/admin

Creates a new platform administrator user.

Parameters

No parameters.

Request body

```
{
  "firstName": "Alan",
  "lastName": "Turing",
  "email": "at@acm.com",
  "password": "oracle"
}
```

Responses

Code	Description
201	Created platform administrator user response.
	Example Value <pre>{ "id": "9ba2ee96-7e24-4dbd-b01a-8537d31e7628", "firstName": "Alan", "lastName": "Turing", "email": "at@acm.com", "password": "oracle", "role": "admin", "createdAt": "2023-03-12T08:35:53Z", "updatedAt": "2023-03-12T08:35:53Z" }</pre>

GET

/api/v1/users/developers/pending

Gets a list of pending developer registration requests.

Parameters

No parameters.

Responses

Code	Description
200	OK response.
	Example Value <pre>[{ "id": "9ba2ee96-7e24-4dbd-b01a-8537d31e7628", "firstName": "Alan", "lastName": "Turing", "email": "at@acm.com", "Password": "oracle", "role": "developer", "address": { "street": "Sackville Park, Fairfield St", "city": "Manchester", "state": "Greater Manchester" }, "country": "United Kingdom", "phoneno": "+441632960416", "ssn": "6735623542345", "govid": { "type": "image/png", </pre>

	<pre> "data": "base64" }, "createdAt": "2023-03-12T08:35:53Z", "updatedAt": "2023-03-12T08:35:53Z" }] </pre>
--	---

GET	/api/v1/users/{id}
Get a user by ID.	
Parameters	
Name	Description
id <small>* required</small> string <i>(path)</i>	ID of the user.
Responses	
Code	Description
200	OK response.
	Example Value
	<pre> { "id": "9ba2ee96-7e24-4dbd-b01a-8537d31e7628", "firstName": "Alan", "lastName": "Turing", "email": "at@acm.com", "password": "oracle", "role": "admin", "createdAt": "2023-03-12T08:35:53Z", "updatedAt": "2023-03-12T08:35:53Z" } </pre>

FundChain Backend Reference API

The API specifications for the platform's backend provide developers with the necessary tools to interact with the off-chain data that drives our platform's operations. These APIs are designed to be easy to use, efficient, and secure, allowing developers to build custom applications and services that enhance the user experience of our platform. In this section, we will provide an overview of the reference API specifications for this component, including the various endpoints and parameters that can be used to query and modify data.

POST	/api/v1/projects
Create a new project.	
Parameters	
No parameters.	

Request body

```
{
  "id": "91160ec7-1f30-425c-bc39-a0b467e98055",
  "title": "My Project",
  "category": "RES",
  "img": {
    "type": "image/png",
    "data": "base64"
  },
  "location": {
    "country": "Greece",
    "city": "Thessaloniki"
  },
  "description": "MyProject description",
  "management_company": {
    "img": {
      "type": "image/png",
      "data": "base64"
    },
    "name": "MyCompany"
  },
  "funding": {
    "goal": 5000000.50,
    "deadline": "2023-03-12T08:35:53Z",
    "extend_until": "2023-03-15T08:35:53Z",
    "min_investment": 50
  },
  "tags": [
    "tag1",
    "tag2"
  ],
  "attachments": [
    {
      "type": "text/html",
      "data": "base64"
    }
  ]
}
```

Responses

Code	Description
201	Successful project creation.
	<p>Example Value</p> <pre>{ "id": "91160ec7-1f30-425c-bc39-a0b467e98055", "ownerid": "8df90dca-2ec9-4f29-858e-110d61f5d719", "title": "My Project", "category": "RES", "created_at": "2023-03-08T08:35:53Z", "updated_at": "2023-03-08T08:35:53Z", "state": "draft", "img": { "type": "image/png", "data": "base64" }, "location": { "country": "Greece", "city": "Thessaloniki" }, }</pre>

```
{
  "description": "MyProject description",
  "management_company": {
    "img": {
      "type": "image/png",
      "data": "base64"
    },
    "name": "MyCompany"
  },
  "funding": {
    "goal": 5000000.50,
    "deadline": "2023-03-12T08:35:53Z",
    "extend_until": "2023-03-15T08:35:53Z",
    "min_investment": 50
  },
  "tags": [
    "tag1",
    "tag2"
  ],
  "attachments": [
    {
      "id": "1508b7e2-260a-47da-bd54-d7f03516ed83",
      "type": "text/html",
      "size": 500,
      "created_at": "2023-03-08T08:35:53Z",
      "updated_at": "2023-03-08T08:35:53Z",
      "data": "base64"
    }
  ]
}
```

GET	/api/v1/projects/{id}
Retrieve a particular project	
Parameters	
Name	Description
id <small>* required</small> string <i>(path)</i>	The project's identifier.
Responses	
Code	Description
200	OK response.
	Example Value <pre>{ "id": "91160ec7-1f30-425c-bc39-a0b467e98055", "ownerid": "8df90dca-2ec9-4f29-858e-110d61f5d719", "title": "My Project", "category": "RES", "created_at": "2023-03-08T08:35:53Z", "updated_at": "2023-03-08T08:35:53Z", "state": "draft", "img": { "type": "image/png", "data": "base64" } }</pre>

```

},
"location":{
  "country":"Greece",
  "city":"Thessaloniki"
},
"description":"MyProject description",
"management_company":{
  "img":{
    "type":"image/png",
    "data":"base64"
  },
  "name":"MyCompany"
},
"funding":{
  "goal":5000000.50,
  "deadline":"2023-03-12T08:35:53Z",
  "extend_until":"2023-03-15T08:35:53Z",
  "min_investment":50
},
"tags":[
  "tag1",
  "tag2"
],
"attachments":[
  {
    "id":"1508b7e2-260a-47da-bd54-d7f03516ed83",
    "type":"text/html",
    "size":500,
    "created_at":"2023-03-08T08:35:53Z",
    "updated_at":"2023-03-08T08:35:53Z",
    "data":"base64"
  }
]
}

```

GET	/api/v1/projects
Retrieve all projects (matching query parameters if specified).	
Parameters	
Name	Description
state string (query)	State of the crowdfunding projects to retrieve.
tag string (query)	Retrieved projects must have this tag assigned to them.
category string (query)	Retrieved projects must have this category assigned to them.
Responses	
Code	Description
200	OK response.
	Example Value

```
[
  {
    "id": "91160ec7-1f30-425c-bc39-a0b467e98055",
    "ownerid": "8df90dca-2ec9-4f29-858e-110d61f5d719",
    "title": "My Project",
    "category": "RES",
    "created_at": "2023-03-08T08:35:53Z",
    "updated_at": "2023-03-08T08:35:53Z",
    "state": "draft",
    "img": {
      "type": "image/png",
      "data": "base64"
    },
    "location": {
      "country": "Greece",
      "city": "Thessaloniki"
    },
    "description": "MyProject description",
    "management_company": {
      "img": {
        "type": "image/png",
        "data": "base64"
      },
      "name": "MyCompany"
    },
    "funding": {
      "goal": 5000000.50,
      "deadline": "2023-03-12T08:35:53Z",
      "extend_until": "2023-03-15T08:35:53Z",
      "min_investment": 50
    },
    "tags": [
      "tag1",
      "tag2"
    ],
    "attachments": [
      {
        "id": "1508b7e2-260a-47da-bd54-d7f03516ed83",
        "type": "text/html",
        "size": 500,
        "created_at": "2023-03-08T08:35:53Z",
        "updated_at": "2023-03-08T08:35:53Z",
        "data": "base64"
      }
    ]
  }
]
```

GET
/api/v1/projects/ids

Retrieve all project identifiers (matching query parameters if specified).

Parameters

Name	Description
state string (query)	State of the crowdfunding project identifiers to retrieve.

tag string (query)	Matched projects must have this tag assigned to them.
category string (query)	Matched projects must have this category assigned to them.
Responses	
Code	Description
200	OK response.
	Example Value
	<pre>["8c8e2b1b-5bc9-421d-9ae3-0e88ef1ec45a", "1c2a4d19-7c0f-453b-b9d6-6e28cd7c1c28"]</pre>

PATCH	/api/v1/projects/{id}
Update data associated with a crowdfunding project.	
Parameters	
Name	Description
id ^{required} string (path)	The project's identifier.
Request body	
<pre>{ "img":{ "type":"image/svg", "data":"base64" }, "description":"MyProject updated description" }</pre>	
Responses	
Code	Description
200	OK response.
	Example Value
	<pre>{ "id":"91160ec7-1f30-425c-bc39-a0b467e98055", "ownerid":"8df90dca-2ec9-4f29-858e-110d61f5d719", "title":"My Project", "category":"RES", "created_at":"2023-03-08T08:35:53Z", "updated_at":"2023-03-08T08:35:53Z", "state":"draft", "img":{ "type":"image/svg", "data":"base64" } }</pre>

```

},
"location":{
  "country":"Greece",
  "city":"Thessaloniki"
},
"description":"MyProject updated description",
"management_company":{
  "img":{
    "type":"image/png",
    "data":"base64"
  },
  "name":"MyCompany"
},
"funding":{
  "goal":5000000.50,
  "deadline":"2023-03-12T08:35:53Z",
  "extend_until":"2023-03-15T08:35:53Z",
  "min_investment":50
},
"tags":[
  "tag1",
  "tag2"
],
"attachments":[
  {
    "id":"1508b7e2-260a-47da-bd54-d7f03516ed83",
    "type":"text/html",
    "size":500,
    "created_at":"2023-03-08T08:35:53Z",
    "updated_at":"2023-03-08T08:35:53Z",
    "data":"base64"
  }
]
}

```

GET	/api/v1/projects/{id}/attachments
Retrieve all attachments (matching query parameters if needed) associated with a particular crowdfunding project.	
Parameters	
Name	Description
id <small>* required</small> string (path)	The project's identifier.
type string (query)	Content type filtering of the attachments to retrieve.
Responses	
Code	Description
200	OK response.
	Example Value
<pre>[{ "id":"1508b7e2-260a-47da-bd54-d7f03516ed83",</pre>	

```
[
  {
    "type": "text/html",
    "size": 500,
    "created_at": "2023-03-08T08:35:53Z",
    "updated_at": "2023-03-08T08:35:53Z",
    "data": "base64"
  }
]
```

GET

/api/v1/projects/{id}/attachments/ids

Retrieve all attachment identifiers (matching query parameters if needed) associated with a particular crowdfunding project.

Parameters

Name	Description
id <small>* required</small> string (path)	The project's identifier.
type string (query)	Content type filtering of the attachment identifiers to retrieve.

Responses

Code	Description
200	OK response.
	Example Value
	["8c8e2b1b-5bc9-421d-9ae3-0e88ef1ec45a", "1c2a4d19-7c0f-453b-b9d6-6e28cd7c1c28"]

GET

/api/v1/projects/{projectid}/attachments/{id}

Retrieve all attachment identifiers (matching query parameters if needed) associated with a particular crowdfunding project.

Parameters

Name	Description
projectid <small>* required</small> string (path)	The project's identifier.
id <small>* required</small> string (path)	The attachment's identifier.

Responses

Code	Description
200	OK response.

Example Value
<pre>{ "id": "1508b7e2-260a-47da-bd54-d7f03516ed83", "type": "text/html", "size": 500, "created_at": "2023-03-08T08:35:53Z", "updated_at": "2023-03-08T08:35:53Z", "data": "base64" }</pre>

PATCH	/api/v1/projects/{projectid}/attachments/{id}
Update data associated with a crowdfunding project's attachment.	
Parameters	
Name	Description
projectid string (path)	The project's identifier.
id string (path)	The attachment's identifier.
Request body	
<pre>{ "type": "application/pdf", "data": "base64" }</pre>	
Responses	
Code	Description
200	OK response.
	Example Value
	<pre>{ "id": "1508b7e2-260a-47da-bd54-d7f03516ed83", "type": "application/pdf", "size": 505, "created_at": "2023-03-08T08:35:53Z", "updated_at": "2023-03-08T08:35:53Z", "data": "base64" }</pre>

POST	/api/v1/projects/{id}/attachments
Add an attachment to a crowdfunding project.	
Parameters	
Name	Description

id <small>* required</small> string <i>(path)</i>	The project's identifier.
Request body	
<pre>{ "type": "application/pdf", "data": "base64" }</pre>	
Responses	
Code	Description
200	OK response.
	Example Value
	<pre>{ "id": "1508b7e2-260a-47da-bd54-d7f03516ed83", "type": "application/pdf", "size": 505, "created_at": "2023-03-08T08:35:53Z", "updated_at": "2023-03-08T08:35:53Z", "data": "base64" }</pre>

DELETE	/api/v1/projects/{projectid}/attachments/{id}
Delete a crowdfunding project's attachment.	
Parameters	
Name	Description
projectid <small>* required</small> string <i>(path)</i>	The project's identifier.
id <small>* required</small> string <i>(path)</i>	The attachment's identifier.
Responses	
Code	Description
200	OK response.

POST	/api/v1/investment-intents/auth
Authorization code endpoint for investment intents.	
Parameters	
No parameters.	

Request body	
<pre>{ "id": "e9e1acc5-c270-4bd9-8c5b-c3050cac063f", "type": "paypal", "amount": { "value": "10.99", "currency_code": "EUR" }, }</pre>	
Responses	
Code	Description
201	Created response.
	Example Value <pre>{ "id": "b76c4e2a-a3c0-495c-abb9-34102ceb87b5", "code": "73946336-9bbc-407b-89be-dddfb9b1c5bc" }</pre>

POST	/api/v1/mints/auth
Authorization code endpoint for mint requests.	
Parameters	
No parameters.	
Request body	
<pre>{ "id": "e9e1acc5-c270-4bd9-8c5b-c3050cac063f", "type": "paypal", "amount": { "value": "10.99", "currency_code": "EUR" }, }</pre>	
Responses	
Code	Description
201	Created response.
	Example Value <pre>{ "id": "b76c4e2a-a3c0-495c-abb9-34102ceb87b5", "code": "73946336-9bbc-407b-89be-dddfb9b1c5bc" }</pre>

POST	/api/v1/burns/auth
-------------	--------------------

Authorization code endpoint for burn requests.	
Parameters	
No parameters.	
Request body	
<pre>{ "id": "e9e1acc5-c270-4bd9-8c5b-c3050cac063f", "type": "paypal", "amount": { "value": "10.99", "currency_code": "EUR" }, }</pre>	
Responses	
Code	Description
201	Created response.
	Example Value
	<pre>{ "id": "b76c4e2a-a3c0-495c-abb9-34102ceb87b5", "code": "73946336-9bbc-407b-89be-dddfb9b1c5bc" }</pre>

POST	/api/v1/webhooks
Generic webhook endpoint for notifications originating from other components. To access this endpoint, a token issued by the FundChain backend must be provided in an agreed-upon authorization header.	
Parameters	
No parameters.	
Request body	
<pre>{ "topic": "payment_order", "id": "0631b37b-0533-4e24-a961-c05bae48ba14", "code": "438488fc-4ef9-49f8-9492-221990098688", "time": "2023-03-12T08:35:55Z", "status": "completed" }</pre>	
Responses	
Code	Description
200	OK response.

Payment Gateway Reference API

In this section, we will provide a reference API specification for the payment gateway component of our platform. The payment gateway serves as a crucial interface between the platform and external payment processors, facilitating various payment-related operations such as authorizing payments and processing reimbursements, among others. We will outline the endpoints, methods, and request/response structures that developers can use to interact with the payment gateway.

POST <code>/api/v1/payments/{id}/enact/{code}</code>	
Initiate the flow for a payment that has already been authorized by the FundChain backend. Assuming the input authorization code is valid, the invoker is redirected to the payment processor's authorization server to input all necessary payment data (e.g., PayPal account credentials, credit card information).	
Parameters	
Name	Description
id * required string (path)	The payment record's identifier.
code * required string (path)	The payment's authorization code.
Responses	
Code	Description
200	OK response.

POST <code>/api/v1/payments</code>	
Authorize a payment targeted to the platform's escrow account. This endpoint can only be invoked by the FundChain backend.	
Parameters	
No parameters.	
Request body	
<pre>{ "code": "73946336-9bbc-407b-89be-dddfb9b1c5bc", "user": { "id": "4179b71f-43d7-4752-8add-614d93a0cf7f", "mail": "user@mail.com" } },</pre>	


```
{
  "type": "paypal",
  "amount": {
    "value": "10.99",
    "currency_code": "EUR"
  },
  "webhook": {
    "url": "http://example.com/webhooks",
    "auth_token": "45ba8065-246b-408d-bb82-f1fa1d6e33ef"
  }
}
```

Responses

Code	Description
201	Created response.
	Example Value <pre>{ "id": "b76c4e2a-a3c0-495c-abb9-34102ceb87b5", "code": "73946336-9bbc-407b-89be-dddfb9b1c5bc", "user": { "id": "4179b71f-43d7-4752-8add-614d93a0cf7f", "mail": "user@mail.com" }, "type": "paypal", "amount": { "value": "10.99", "currency_code": "EUR" }, "webhook": { "url": "http://example.com/webhooks", "auth_token": "45ba8065-246b-408d-bb82-f1fa1d6e33ef" }, "status": "auth", "created_at": "2023-03-12T08:35:53Z", "completed_at": "" }</pre>

GET	/api/v1/payments
Retrieve all payments matching optional query parameter criteria. This endpoint can only be invoked by the FundChain backend.	
Parameters	
Name	Description
status string (query)	Status of payments to retrieve (e.g., auth).
type string (query)	The type of payments that are of interest (e.g., paypal).

created_from string (query)	Starting range timestamp for payment record creation.
created_to string (query)	Ending range timestamp for payment record creation.
completed_from string (query)	Starting range timestamp for payment completion.
completed_to string (query)	Ending range timestamp for payment completion.
Responses	
Code	Description
200	OK response.
	Example Value
	<pre>[{ "id": "b76c4e2a-a3c0-495c-abb9-34102ceb87b5", "code": "73946336-9bbc-407b-89be-dddfb9b1c5bc", "user": { "id": "4179b71f-43d7-4752-8add-614d93a0cf7f", "mail": "user@mail.com" }, "type": "paypal", "amount": { "value": "10.99", "currency_code": "EUR" }, "webhook": { "url": "http://example.com/webhooks", "auth_token": "45ba8065-246b-408d-bb82-f1fa1d6e33ef" }, "status": "auth", "created_at": "2023-03-12T08:35:53Z", "completed_at": "2023-03-12T08:35:55Z" }]</pre>

GET	/api/v1/payments/{id}
Retrieve a specific payment record. This endpoint can only be invoked by the FundChain backend.	
Parameters	
Name	Description
id * required string (path)	The payment record's identifier.
Responses	

Code	Description
200	OK response.
	Example Value <pre> { "id": "b76c4e2a-a3c0-495c-abb9-34102ceb87b5", "code": "73946336-9bbc-407b-89be-dddfb9b1c5bc", "user": { "id": "4179b71f-43d7-4752-8add-614d93a0cf7f", "mail": "user@mail.com" }, "type": "paypal", "amount": { "value": "10.99", "currency_code": "EUR" }, "webhook": { "url": "http://example.com/webhooks", "auth_token": "45ba8065-246b-408d-bb82-f1fa1d6e33ef" }, "status": "auth", "created_at": "2023-03-12T08:35:53Z", "completed_at": "2023-03-12T08:35:55Z" }</pre>

Annex II: Smart Contracts Specifications

Fungible Token Management

This section presents the specifications of a smart contract that provides fungible token functionality on Hyperledger Fabric, a highly specialized and optimized tool, designed to meet the specific needs of permissioned environments. It has been stripped of all superfluous functionalities, which may be present in other similar contracts, but have no practical use in real-world applications and offers increased privacy and scalability, making it an ideal choice for enterprise-level use cases.

Unlike typical ERC-20 fungible token smart contracts, which require the specification of the total number of coins and their denominations during setup, this smart contract allows for secure protocol implementations for minting and burning coins. This feature addresses one of the major limitations of the ERC-20 token standard, making it possible to adapt the smart contract to real-world scenarios, where such limitations can be quite restricting.

Moreover, to facilitate financial commitments among transacting parties, this smart contract comes with a novel “coin hold” mechanism, which ensures that users can securely lock a portion of their account balance, thus enabling them to make and honor financial commitments as needed. The mechanism works by locking the held coins completely, rendering them unusable until a designated notary smart contract determines whether the coins should be returned to the user's active balance or transferred to one or more accounts of other users.

Originally developed to support the efforts of the RENAISSANCE H2020 project, this smart contract has since been extended to support the efforts of the FEVER H2020 project. As a result, it is a robust and tested tool that can be used as-is, with confidence, to meet the fungible token needs of businesses and organizations operating in permissioned environments.

INVOKE		Init
Chaincode initialization function, invoked once when the contract is deployed on the network.		
Parameters		
#	Name	Example Value
0	FTConfig ^{required} JSON	<pre>{ "mintNotary":{ "channel":"ianos", "contract":"ianos-ft-notary", "checkoperation":"MintNotary" }, "burnNotary":{ "channel":"ianos", "contract":"ianos-ft-notary", "checkoperation":"BurnNotary" } }</pre>

		<pre> }, "govPrincipalNotary":{ "channel":"ianos", "contract":"ianos-ft-notary", "checkoperation":"CheckOwnerStatus" }, "instName":"ianos-ft", "instChannel":"ianos" } </pre>
Responses		
Status	Example Value	
Success	n/a	

INVOKE	CreateAccount
<p>When a user invokes this function, a new fungible token account is established and linked to her identity, with complete ownership and control over it. The account is initialized with a balance of zero, and there is no restriction on the number of accounts that a user can create.</p>	
Parameters	
No parameters.	
Responses	
Status	Example Value
Success	<pre> { "ID":"YTcyMD1hZWI5NmUxN2U1Z", "balance":0, "holds":[], "mints":[], "burns":[] } </pre>

QUERY		GetAccount
Retrieve information about a particular account using its unique identifier (accountID). Only the account owner is authorized to access this information.		
Parameters		
#	Name	Example Value
0	accountID <small>* required</small> string	YTcyMD1hZWI5NmUxN2U1Z
Responses		
Status	Example Value	
Success	{ "ID": "YTcyMD1hZWI5NmUxN2U1Z", "balance": 0, "holds": [],	

```
"mints":[],  
"burns":[]  
}
```

INVOKE		Transfer
The function facilitates the transfer of funds between accounts, with access limited to the account owner. In other words, the invoking user must possess ownership of the source account from which the coins are being transferred.		
Parameters		
#	Name	Example Value
0	TransferFT ^{required} JSON	<pre>{ "srcAccountID": "YTcyMD1hZWl5NmUxN2U1Z", "destAccountID": "MWUxZjljZDdlMGZlMzM5ZDd", "amount": 5.32 }</pre>
Responses		
Status		Example Value
Success		n/a

INVOKE		RequestMint
This function enables the submission of a mint request that is linked to a specific account. It necessitates a JSON encoded MintBurn object as input (refer to the example below). The request's identifier is returned as output, which may be used to approve or reject it by interfacing with the notary contract. Access to this function is restricted to the account owner.		
Parameters		
#	Name	Example Value
0	MintBurnFT ^{required} JSON	<pre>{ "accountID": "YTcyMD1hZWl5NmUxN2U1Z", "amount": 5.32, "assocComData": "any value", }</pre>
Status		Example Value
Success		ft:mint:ZdkwYzNIN2IXYThh

INVOKE		ExecuteMint
This function aims to execute a mint request based on the provided input identifier (mintID). However, the mint request must be approved for execution beforehand by interfacing with the notary smart contract.		
Parameters		
#	Name	Example Value
0	mintID ^{required} string	ft:mint:ZdkwYzNIN2IXYThh
Responses		
Status		Example Value
Success		n/a

QUERY		GetMintRequests
<p>This function is meant to be used by owners of the notary smart contract, enabling them to obtain all mint requests issued to the fungible token smart contract across all user accounts. This capability allows these entities to assess whether mint requests should be approved for execution or not before interfacing with the notary smart contract and declaring their decision.</p>		
Parameters		
No parameters.		
Responses		
Status	Example Value	
Success	<pre>[{ "accountID": "YTcyMDlhZWl5NmUxN2U1Z", "amount": 5.32, "assocComData": "any value" }, { "accountID": "YTcy65y35WI5NmUxN2U1Z", "amount": 15.32, "assocComData": "any value" }]</pre>	

INVOKE		RequestBurn
<p>This function enables the submission of a burn request that is linked to a specific account. It necessitates a JSON encoded MintBurn object as input (refer to the example below). The request's identifier is returned as output, which may be used to approve or reject it by interfacing with the notary contract. Access to this function is restricted to the account owner.</p>		
Parameters		
#	Name	Example Value
0	MintBurnFT ^{required} JSON	<pre>{ "accountID": "YTcyMDlhZWl5NmUxN2U1Z", "amount": 5.32, "assocComData": "any value", }</pre>
Responses		
Status	Example Value	
Success	ft:burn:ZdkwYzN1N2IxYThh	

INVOKE		ExecuteBurn
<p>This function aims to execute a burn request based on the provided input identifier (burnID). However, the burn request must be approved for execution beforehand by interfacing with the notary smart contract.</p>		
Parameters		
#	Name	Example Value

0	burnID ^{required} string	ft:burn:ZdkwYzNlN2IxYThh
Responses		
Status	Example Value	
Success	n/a	

QUERY	GetBurnRequests
<p>This function is meant to be used by owners of the notary smart contract, enabling them to obtain all burn requests issued to the fungible token smart contract across all user accounts. This capability allows these entities to assess whether burn requests should be approved for execution or not before interfacing with the notary smart contract and declaring their decision.</p>	
Parameters	
No parameters.	
Responses	
Status	Example Value
Success	<pre>[{ "accountID": "YTcyMD1hZWl5NmUxN2U1Z", "amount": 5.32, "assocComData": "any value" }, { "accountID": "YTcy65y35WI5NmUxN2U1Z", "amount": 15.32, "assocComData": "any value" }]</pre>

INVOKE		CreateHold
<p>This function generates a coin hold that is associated with a specific account. To create a coin hold, a JSON encoded HoldRequest object must be supplied as input (see example below). Only the account owner can invoke this function; otherwise, an access control error will be returned. The function returns the hold's identifier (holdID) as output, which can be used to authorize its execution or release by interacting with the designated notary.</p>		
Parameters		
#	Name	Example Value
0	HoldRequest ^{required} JSON	<pre>{ "accountID": "YTcyMD1hZWl5NmUxN2U1ZWJhODE1MzY1OWI", "amount": 5.3, "releaseNotary": { "channel": "fever", "contract": "dummy-hold-notary", "checkoperation": "CheckReleaseHold" }, "executeNotary": { "channel": "fever", "contract": "dummy-hold-notary",</pre>

		<pre>"checkoperation": "CheckExecuteHold" }</pre>
Responses		
Status	Example Value	
Success	ft:hold:MHg1YWFhMjNjMTc5ZGQ5NjQ2	

INVOKE		ExecuteHold
<p>This function generates a coin hold that is associated with a specific account. To create a coin hold, a JSON encoded HoldRequest object must be supplied as input (see example below). Only the account owner can invoke this function; otherwise, an access control error will be returned. The function returns the hold's identifier (holdID) as output, which can be used to authorize its execution or release by interacting with the designated notary.</p>		
Parameters		
#	Name	Example Value
0	HoldExec JSON <small>*required</small>	<pre>{ "ID": "ft:hold:Nzk4YWY5YTAzOTRjM2FiYmZhOGQ3YTRh", "payouts": [{ "accountID": "MwUxZjljZDJlMGZlMzM5ZDdkZGExMjUwYWQ0Z", "amount": 3.1 }] }</pre>
Responses		
Status	Example Value	
Success	n/a	

INVOKE		ReleaseHold
<p>To release a previously created coin hold, the invoking user must provide the corresponding hold identifier (holdID) as input. Before the hold can be released, it must first be approved for release by interacting with the designated hold notary smart contract. Once approval is obtained, the hold is successfully released, and the initially held coins are unlocked and returned to the account's active balance.</p>		
Parameters		
#	Name	Example Value
0	holdID string <small>*required</small>	ft:hold:MHg1YWFhMjNjMTc5ZGQ5NjQ2
Responses		
Status	Example Value	
Success	n/a	

Fungible Token Notarization

In any token-based system, it is crucial to have a well-defined set of policies and conditions that govern the minting and burning of tokens. These policies and conditions ensure that the token issuance process is transparent and secure, and that the token's value is maintained over time. In the context of the previously presented fungible token smart contract, the specification and implementation of such policies and conditions play a critical role in ensuring the integrity of the token economy. This section is dedicated to presenting the specifications of a Hyperledger Fabric smart contract that encodes and evaluates the policies and conditions that govern the minting and burning process of the fungible token smart contract. We will delve into the details of this smart contract and discuss how it ensures the secure and compliant issuance and destruction of tokens.

INVOKE		MintNotary
This function encodes and evaluates the policies governing the minting process of fungible tokens.		
Parameters		
#	Name	Example Value
0	MintBurnFT ^{required} JSON	<pre>{ "accountID": "YTcyMD1hZWl5NmUxN2U1Z", "amount": 5.32, "assocComData": "any value", }</pre>
Responses		
Status	Example Value	
Success	"APPROVED"	

INVOKE		BurnNotary
This function encodes and evaluates the policies governing the burning process of fungible tokens.		
Parameters		
#	Name	Example Value
0	MintBurnFT ^{required} JSON	<pre>{ "accountID": "YTcyMD1hZWl5NmUxN2U1Z", "amount": 5.32, "assocComData": "any value", }</pre>
Responses		
Status	Example Value	
Success	"APPROVED"	

QUERY		CheckOwnerStatus
This function checks, based on the invoker's identity, whether the invoker is an owner of this notary contract, or not and responds appropriately with a notary status code.		
Parameters		
No parameters.		
Responses		
Status	Example Value	
Success	"APPROVED"	

Project & Investment Handling

In this section, we will delve into the specifications of a Hyperledger Fabric smart contract that provides a complete solution for managing the lifecycle of crowdfunding projects and their associated investments. This smart contract is designed to operate as a finite state machine, modeling the state transitions of crowdfunding projects and investments, in accordance with the FSM diagrams presented in earlier sections.

Through the use of this smart contract, crowdfunding projects can be managed in a highly secure, transparent, and auditable manner, providing a trusted and reliable platform for investors and project developers alike. It provides a comprehensive solution for handling the entire lifecycle of crowdfunding projects, from, e.g., the funding stage to the disbursement of funds upon project completion.

Moreover, this smart contract has been designed to be scalable, enabling the platform to handle large volumes of transactions, while maintaining high levels of performance and reliability. With its support for Hyperledger Fabric's permissioned blockchain network, it ensures the privacy and security of all transactions and data.

INVOKE		projectCreate
Create a new crowdfunding project.		
Parameters		
#	Name	Example Value
0	ProjectReq <small>* required</small> JSON	<pre>{ "id": "91160ec7-1f30-425c-bc39-a0b467e98055", "ownerId": "8df90dca-2ec9-4f29-858e-110d61f5d719", "ftAccountId": "8558401d-2dcc-473e-838b-b739866800ae", "funding": { "goal": 5000000.50, "deadline": "2023-03-12T08:35:53Z", "extend_until": "2023-03-15T08:35:53Z", "min_investment": 50 } }</pre>

Responses	
Status	Example Value
Success	<pre>{ "id": "91160ec7-1f30-425c-bc39-a0b467e98055", "ownerId": "8df90dca-2ec9-4f29-858e-110d61f5d719", "ftAccountId": "8558401d-2dcc-473e-838b-b739866800ae", "funding": { "goal": 5000000.50, "deadline": "2023-03-12T08:35:53Z", "extend_until": "2023-03-15T08:35:53Z", "min_investment": 50 }, "state": "draft", "created_at": "2023-03-08T08:35:53Z", "updated_at": "2023-03-08T08:35:53Z" }</pre>

INVOKE		projectStateTransition
Advance, or modify the state of a crowdfunding project, according to the FSM's description and state transition policies.		
Parameters		
#	Name	Example Value
0	ProjFSMST ^{*required} JSON	<pre>{ "id": "91160ec7-1f30-425c-bc39-a0b467e98055", "state": "funding" }</pre>
Responses		
Status	Example Value	
Success	<pre>{ "id": "91160ec7-1f30-425c-bc39-a0b467e98055", "ownerId": "8df90dca-2ec9-4f29-858e-110d61f5d719", "funding": { "goal": 5000000.50, "deadline": "2023-03-12T08:35:53Z", "extend_until": "2023-03-15T08:35:53Z", "min_investment": 50 }, "state": "funding", "created_at": "2023-03-08T08:35:53Z", "updated_at": "2023-03-09T08:35:53Z" }</pre>	

INVOKE		investmentIntent
Submit an investment intent towards a crowdfunding project.		
Parameters		
#	Name	Example Value

0	InvIntent <small>required</small> JSON	<pre>{ "id": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3", "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055", "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873", "amount": 55, "payment_type": "paypal" }</pre>
Responses		
Status		Example Value
Success		<pre>{ "id": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3", "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055", "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873", "amount": 55, "payment_type": "paypal", "state": "intent", "created_at": "2023-03-08T08:35:53Z", "updated_at": "2023-03-09T08:35:53Z" }</pre>

INVOKE		investmentFungibleToken
Submit a fungible token-based investment towards a crowdfunding project.		
Parameters		
#	Name	Example Value
0	FTInvReq <small>required</small> JSON	<pre>{ "id": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3", "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055", "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873", "amount": 55, "ftAccountId": "abe4fac1-9c35-4092-8f89-02b7df819e0a" }</pre>
Responses		
Status		Example Value
Success		<pre>{ "id": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3", "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055", "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873", "amount": 55, "state": "reserved", "created_at": "2023-03-08T08:35:53Z", "updated_at": "2023-03-09T08:35:53Z", "payment_type": "ft", "ftAccountId": "abe4fac1-9c35-4092-8f89-02b7df819e0a", "holdId": "ft:hold:8ecf0e3b-ab8c-4d9d-9731-08b195622fa4" }</pre>

INVOKE	investmentStateTransition
---------------	----------------------------------

Advance, or modify the state of an investment, according to the FSM's description and state transition policies.

Parameters

#	Name	Example Value
0	InvFSMST JSON <small>* required</small>	<pre>{ "id": "91160ec7-1f30-425c-bc39-a0b467e98055", "state": "reimbursed" }</pre>

Responses

Status	Example Value
Success	<pre>{ "id": "91160ec7-1f30-425c-bc39-a0b467e98055", "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055", "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873", "amount": 55, "payment_type": "paypal", "state": "reimbursed", "created_at": "2023-03-08T08:35:53Z", "updated_at": "2023-03-09T08:35:53Z" }</pre>

QUERY

project

Get a project by its identifier.

Parameters

#	Name	Example Value
0	id <small>* required</small> string	The project's identifier.

Responses

Status	Example Value
Success	<pre>{ "id": "91160ec7-1f30-425c-bc39-a0b467e98055", "ownerId": "8df90dca-2ec9-4f29-858e-110d61f5d719", "ftAccountId": "8558401d-2dcc-473e-838b-b739866800ae", "funding": { "goal": 5000000.50, "deadline": "2023-03-12T08:35:53Z", "extend_until": "2023-03-15T08:35:53Z", "min_investment": 50 }, "state": "draft", "created_at": "2023-03-08T08:35:53Z", "updated_at": "2023-03-08T08:35:53Z" }</pre>

QUERY

projects

Retrieve all projects from the smart contract's internal state.

Parameters

No parameters.

Responses

Status	Example Value
--------	---------------

Success

```
[
  {
    "id": "91160ec7-1f30-425c-bc39-a0b467e98055",
    "ownerId": "8df90dca-2ec9-4f29-858e-110d61f5d719",
    "ftAccountId": "8558401d-2dcc-473e-838b-b739866800ae",
    "funding": {
      "goal": 5000000.50,
      "deadline": "2023-03-12T08:35:53Z",
      "extend_until": "2023-03-15T08:35:53Z",
      "min_investment": 50
    },
    "state": "draft",
    "created_at": "2023-03-08T08:35:53Z",
    "updated_at": "2023-03-08T08:35:53Z"
  }
]
```

QUERY

projectInvestments

Retrieve all investments associated with a specific project.

Parameters

#	Name	Example Value
---	------	---------------

0	id <small>* required</small> string	The project's identifier.
---	--	---------------------------

Responses

Status	Example Value
--------	---------------

Success

```
[
  {
    "id": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3",
    "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055",
    "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873",
    "amount": 55,
    "payment_type": "paypal",
    "state": "intent",
    "created_at": "2023-03-08T08:35:53Z",
    "updated_at": "2023-03-09T08:35:53Z"
  }
]
```

QUERY

investment

Get an investment by its identifier.

Parameters		
#	Name	Example Value
0	id <small>*required</small> string	The investment's identifier.

Responses	
Status	Example Value
Success	<pre>{ "id": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3", "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055", "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873", "amount": 55, "payment_type": "paypal", "state": "intent", "created_at": "2023-03-08T08:35:53Z", "updated_at": "2023-03-09T08:35:53Z" }</pre>

Project & Investment Notarization

In the previous section, we presented the specifications for a smart contract that handles the entire lifecycle of crowdfunding projects and their respective investments, modeled as finite state machines. In this section, we will dive deeper into the implementation details of the smart contract and present the specifications that encode and evaluate the policies and conditions that govern the state transitions of these finite state machines. These specifications will ensure that all state transitions are executed correctly and that the contract enforces the rules and policies defined in the FSM diagrams. Additionally, the specifications will ensure that the smart contract is secure, scalable, and efficient.

INVOKE		ValidateProjectFSMStateTransition
This function is responsible for encoding and assessing the rules and requirements that dictate the state changes of crowdfunding projects, according to the FSM specifications presented previously.		
Parameters		
#	Name	Example Value
0	ProjSTReq <small>*required</small> string	<pre>{ "project": { "id": "91160ec7-1f30-425c-bc39-a0b467e98055", "ownerId": "8df90dca-2ec9-4f29-858e-110d61f5d719", "ftAccountId": "8558401d-2dcc-473e-838b-b739866800ae", "funding": { "goal": 5000000.50, "deadline": "2023-03-12T08:35:53Z", "extend_until": "2023-03-15T08:35:53Z", "min_investment": 50 }, "state": "draft", "created_at": "2023-03-08T08:35:53Z", "updated_at": "2023-03-08T08:35:53Z" }, }</pre>

		<pre> "investments":[{ "id":"71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3", "projectId":"91160ec7-1f30-425c-bc39-a0b467e98055", "investorId":"664f57a1-e5e2-4230-9d48-611c0229b873", "amount":55, "payment_type":"paypal", "state":"intent", "created_at":"2023-03-08T08:35:53Z", "updated_at":"2023-03-09T08:35:53Z" }, { "id":"71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3", "projectId":"91160ec7-1f30-425c-bc39-a0b467e98055", "investorId":"664f57a1-e5e2-4230-9d48-611c0229b873", "amount":55, "state":"reserved", "created_at":"2023-03-08T08:35:53Z", "updated_at":"2023-03-09T08:35:53Z", "payment_type":"ft", "ftAccountId":"abe4fac1-9c35-4092-8f89-02b7df819e0a", "holdId":"ft:hold:8ecf0e3b-ab8c-4d9d-9731-08b195622fa4" }], "target_state":"funding" } </pre>
Responses		
Status	Example Value	
Success	"APPROVED"	

INVOKE		ValidateInvestmentFSMStateTransition
This function is responsible for encoding and assessing the rules and requirements that dictate the state changes of investments, according to the FSM specifications presented previously.		
Parameters		
#	Name	Example Value
0	InvSTReq ^{required} string	<pre> { "project":{ "id":"91160ec7-1f30-425c-bc39-a0b467e98055", "ownerId":"8df90dca-2ec9-4f29-858e-110d61f5d719", "ftAccountId":"8558401d-2dcc-473e-838b-b739866800ae", "funding":{ "goal":5000000.50, "deadline":"2023-03-12T08:35:53Z", "extend_until":"2023-03-15T08:35:53Z", "min_investment":50 }, "state":"draft", "created_at":"2023-03-08T08:35:53Z", "updated_at":"2023-03-08T08:35:53Z" }, "investments":[{ "id":"71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3", "projectId":"91160ec7-1f30-425c-bc39-a0b467e98055", </pre>

		<pre> "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873", "amount": 55, "payment_type": "paypal", "state": "intent", "created_at": "2023-03-08T08:35:53Z", "updated_at": "2023-03-09T08:35:53Z" }, { "id": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3", "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055", "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873", "amount": 55, "state": "reserved", "created_at": "2023-03-08T08:35:53Z", "updated_at": "2023-03-09T08:35:53Z", "payment_type": "ft", "ftAccountId": "abe4fac1-9c35-4092-8f89-02b7df819e0a", "holdId": "ft:hold:8ecf0e3b-ab8c-4d9d-9731-08b195622fa4" }], "investmentId": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3", "target_state": "reserved" } </pre>
Responses		
Status	Example Value	
Success	"APPROVED"	

Investment & Equity Tokenization

In the context of the proposed blockchain-based equity crowdfunding platform, the ownership of investments and equity is an essential aspect that requires a smart contract solution. To this end, a Hyperledger Fabric smart contract will be employed to provide non-fungible token (NFT) functionality, which is specifically tailored to support the needs of the project.

Compared to typical ERC-721 non-fungible token smart contracts, the smart contract presented here offers a set of distinct advantages. First, it builds on top of the notary interaction pattern introduced previously, providing secure protocol implementations for minting and burning tokens. Second, the smart contract enforces an integrity check for NFTs that are minted and associated with metadata stored in off-chain repositories, ensuring the immutability and accuracy of the token's metadata information. Finally, the smart contract provides a range of query-related functionalities, addressing the severe lack of query-related functionality of ERC-721 like contracts.

It is important to note that this smart contract was originally developed to support the efforts of the knowEdge H2020 project and will be used as is. The specifications presented here represent the culmination of a collaborative effort to design a robust, flexible, and scalable solution for arbitrary NFTs.

INVOKE	Init
---------------	-------------

This is the chaincode's initialization function that must be invoked as part of its lifecycle process, i.e., during its deployment on a channel.

Parameters

#	Name	Example Value
0	NFTConfig ^{required} JSON	<pre>{ "name": "NFT", "symbol": "Symbol", "mintNotary": { "channel": "ianos", "contract": "ianos-nft-notary", "function": "MintNotary" } }</pre>

Responses

Status	Example Value
Success	n/a

QUERY

OwnerOf

Outputs the identifier of the owner (ownerID) of the NFT.

Parameters

#	Name	Example Value
0	tokenID ^{required} string	123e4567-e89b-12d3-a456-426614174000

Responses

Status	Example Value
Success	nft:userid:eDUwOTpUZXN0TVNQSUQ6ZURVd09U

INVOKE

Transfer

This function is meant for NFT owners solely. It allows them to transfer NFT ownership to a receiver of their choice. The identifiers of the NFT owner, the receiver and the NFT itself are encoded in the input payload.

Parameters

#	Name	Example Value
0	TransferNFT ^{required} JSON	<pre>{ "receiverID": "nft:userid:eDUwOTpUZXN0TVNQSUQ6ZURVd09U", "tokenID": "123e4567-e89b-12d3-a456-426614174000" }</pre>

Responses

Status	Example Value
--------	---------------

Success	n/a
---------	-----

INVOKE		Mint
Mints, or creates, a new NFT that is assigned to a specific owner. The NFT Chaincode invokes the notary with which it is configured and proceeds with the minting process only if it is approved by the notary. Note that the combination of tokenURI and metadata fields are optional, i.e., either both will have a value, or none at all.		
Parameters		
#	Name	Example Value
0	MintNFT <small>required</small> JSON	<pre>{ "tokenId": "123e4567-e89b-12d3-a456-426614174000", "ownerID": "nft:userId:eDUwOTpUZXN0TVNQSUQ6ZURVd09U", "metadata": { "algo": "SHA256", "tokenURI": "uri:nft:123e4567-e89b-12d3-a456-426614174000", "hash": "eDUwOTpUZXN0TVNQSUQ6ZURVd09U" }, "data": "any value" }</pre>
Responses		
Status	Example Value	
Success	n/a	

QUERY		GetInvokerID
This is a utility function that allows the invoking client to compute her identifier that can later on be used for, e.g., minting, transferring NFTs etc.		
Parameters		
No parameters.		
Responses		
Status	Example Value	
Success	nft:userId:eDUwOTpUZXN0TVNQSUQ6ZURVd09U	

QUERY		Exists
This function allows the invoker to infer whether a token, identified by the function's input, exists or not.		
Parameters		
#	Name	Example Value
0	tokenId <small>required</small> string	123e4567-e89b-12d3-a456-426614174000

Responses	
Status	Example Value
Success	true

QUERY		GetOwnedNFTIDs
This function queries the contract's state for the identifiers of all the tokens that are owned by the invoker of the function.		
Parameters		
No parameters.		
Responses		
Status	Example Value	
Success	<pre>["123e4567-e89b-12d3-a456-426614174000", "123e4567-e89b-12d3-a456-426614174123"]</pre>	

QUERY		GetAllNFTIDs
This function outputs the entire list of token identifiers that are maintained in the contract's state.		
Parameters		
No parameters.		
Responses		
Status	Example Value	
Success	<pre>["123e4567-e89b-12d3-a456-426614174000", "123e4567-e89b-12d3-a456-426614174123"]</pre>	

QUERY		GetNFT
This function, on input a token's identifier, retrieves the entire NFT object from the contract's state.		
Parameters		
#	Name	Example Value
0	tokenID <small>* required</small> string	123e4567-e89b-12d3-a456-426614174000
Responses		
Status	Example Value	
Success	<pre>{ "tokenId": "123e4567-e89b-12d3-a456-426614174000", "ownerID": "nft:userId:EDUwOTpUZXN0TVNQSUQ6ZURVd09U", "metadata": { "algo": "SHA256", "tokenURI": "uri:nft:123e4567-e89b-12d3-a456-426614174000", "hash": "EDUwOTpUZXN0TVNQSUQ6ZURVd09U" }, }</pre>	

```
"data": "any value"
}
```

QUERY

GetAllNFTs

Outputs the entire list of NFTs that are maintained in the contract's state.

Parameters

No parameters.

Responses

Status

Example Value

Success

```
[
  {
    "tokenId": "123e4567-e89b-12d3-a456-426614174000",
    "ownerID": "nft:userid:EDUwOTpUZXN0TVNQSUQ6ZURVd09U",
    "metadata": {
      "algo": "SHA256",
      "tokenURI": "uri:nft:123e4567-e89b-12d3-a456-426614174000",
      "hash": "EDUwOTpUZXN0TVNQSUQ6ZURVd09U"
    },
    "data": "any value"
  },
  {
    "tokenId": "123e4567-e89b-12d3-a456-426614174123",
    "ownerID": "nft:userid:EDUwOTpUZXN0TVNQSUQ6ZURVd123",
    "metadata": {
      "algo": "SHA256",
      "tokenURI": "uri:nft:123e4567-e89b-12d3-a456-426614174123",
      "hash": "EDUwOTpUZXN0TVNQSUQ6ZURVd123"
    },
    "data": "any value"
  }
]
```

QUERY

GetAllOwnedNFTs

This function queries the contract's state for the concrete NFT objects that are owned by the invoker.

Parameters

No parameters.

Responses

Status

Example Value

Success

```
[
  {
    "tokenId": "123e4567-e89b-12d3-a456-426614174000",
    "ownerID": "nft:userid:EDUwOTpUZXN0TVNQSUQ6ZURVd09U",
    "metadata": {
      "algo": "SHA256",
      "tokenURI": "uri:nft:123e4567-e89b-12d3-a456-426614174000",
      "hash": "EDUwOTpUZXN0TVNQSUQ6ZURVd09U"
    }
  }
]
```


	<pre> }, "data": "any value" }, { "tokenId": "123e4567-e89b-12d3-a456-426614174123", "ownerID": "nft:userid:eDUwOTpUZXN0TVNQSUQ6ZURVd123", "metadata": { "algo": "SHA256", "tokenURI": "uri:nft:123e4567-e89b-12d3-a456-426614174123", "hash": "eDUwOTpUZXN0TVNQSUQ6ZURVd123" }, "data": "any value" }] </pre>
--	---

Investment Notarization

In the proposed blockchain-based equity crowdfunding platform, investments will be represented as Non-Fungible Tokens (NFTs) and managed using a Hyperledger Fabric smart contract. In this section, we provide the specification of a smart contract that encodes and evaluates the policies and conditions that govern the minting of investments, ensuring their secure creation and management, thus, ensuring their validity.

INVOKE		MintNotary
This function encodes and evaluates the policies governing the minting process of investment NFTs.		
Parameters		
#	Name	Example Value
0	MintNFT <small>required</small> JSON	<pre> { "tokenId": "123e4567-e89b-12d3-a456-426614174000", "ownerID": "nft:userid:eDUwOTpUZXN0TVNQSUQ6ZURVd09U", "metadata": { "algo": "SHA256", "tokenURI": "uri:nft:123e4567-e89b-12d3-a456-426614174000", "hash": "eDUwOTpUZXN0TVNQSUQ6ZURVd09U" }, "data": "any value" } </pre>
Responses		
Status	Example Value	
Success	"APPROVED"	

Equity Notarization

This section presents the specifications of a Hyperledger Fabric smart contract that enables the modeling of equity ownership as non-fungible tokens (NFTs). The smart

contract encodes and evaluates the policies and conditions that govern the minting process of equity NFTs, ensuring the integrity and security of the platform.

INVOKE		MintNotary
This function encodes and evaluates the policies governing the minting process of equity NFTs.		
Parameters		
#	Name	Example Value
0	MintNFT ^{*required} JSON	<pre>{ "tokenId": "123e4567-e89b-12d3-a456-426614174000", "ownerID": "nft:userid:EDUwOTpUZXN0TVNQSUQ6ZURVd09U", "metadata": { "algo": "SHA256", "tokenURI": "uri:nft:123e4567-e89b-12d3-a456-426614174000", "hash": "EDUwOTpUZXN0TVNQSUQ6ZURVd09U" }, "data": "any value" }</pre>
Responses		
Status	Example Value	
Success	"APPROVED"	

Annex III: Blockchain Cloud Services

The blockchain-based equity crowdfunding platform comprises a collection of individual services that operate together in order to deliver a seamless user experience. These services have been designed to provide a robust and secure infrastructure that can handle the various aspects of equity crowdfunding. In order to ensure that the platform is easy to use and operate, a comprehensive set of API specifications has been developed for each of the individual services. These specifications have been designed with the goal of making it as easy as possible for developers to integrate with the platform and create new applications and services that can take advantage of the underlying infrastructure. In this section, we will present the API specifications for each of the individual services that compose the blockchain cloud services of the platform. These services include the fungible token, project, investment and equity services, to each of which we dedicate a separate subsection. By providing these specifications, we hope to provide a detailed understanding of how each of the services operates, as well as the specific parameters that need to be provided in order to interact with them.

Fungible Token Service

In the world of blockchain and smart contracts, fungible tokens are one of the most popular use cases. These tokens are digital assets that are interchangeable with other tokens of the same type and value. The management of these tokens is critical to any blockchain-based platform that deals with transactions of value. This is where the fungible token management service comes into play. This service provides a set of APIs that enable developers to create, issue, and manage fungible tokens on the blockchain network. In this section, we will delve into the API specifications of the fungible token management service, outlining the various endpoints that it provides.

POST	/api/v1/account
Creates a new fungible token account.	
Parameters	
No parameters.	
Responses	
Code	Description
201	Successful fungible token account creation.
	Example Value
	<pre>{ "ID": "YTcyMDlhZWl5NmUxN2U1Z", "balance": 0, "holds": [], "mints": [],</pre>

```
"burns": []  
}
```

GET	/api/v1/account/{accountid}
This endpoint allows the transfer of funds between fungible token accounts.	
Parameters	
Name	Description
accountid <small>* required</small> string <i>(path)</i>	Fungible token account identifier.
Responses	
Code	Description
200	OK response.
	Example Value
	<pre>{ "ID": "YTcyMDlhZWl5NmUxN2U1Z", "balance": 0, "holds": [], "mints": [], "burns": [] }</pre>

POST	/api/v1/transfer
This endpoint allows the transfer of funds between fungible token accounts.	
Parameters	
No parameters.	
Request body	
<pre>{ "srcAccountID": "YTcyMDlhZWl5NmUxN2U1Z", "destAccountID": "MWUxZjljZDl1MGZlMzMSZDd", "amount": 5.32 }</pre>	
Responses	
Code	Description
201	OK response.

POST	/api/v1/mint
Issue a mint request that is bound to a particular fungible token account.	
Parameters	
No parameters.	

Request body	
<pre>{ "accountID": "YTcyMD1hZWl5NmUxN2U1Z", "amount": 5.32, "assocComData": "any value", }</pre>	
Responses	
Code	Description
201	Successful mint request issuance.
	Example Value
	ft:mint:ZdkwYzN1N2IxYThh

POST	/api/v1/mint/{mintid}/execute
Attempt to execute a mint request based on its input identifier.	
Parameters	
Name	Description
mintID <small>required</small> string <i>(path)</i>	The identifier of the mint request.
Responses	
Code	Description
200	OK response.

GET	/api/v1/mints
Retrieve all mint requests.	
Parameters	
No parameters.	
Responses	
Code	Description
200	OK response.
	Example Value
	<pre>[{ "accountID": "YTcyMD1hZWl5NmUxN2U1Z", "amount": 5.32, "assocComData": "any value" },]</pre>

	<pre>{ "accountID": "YTcy65y35WI5NmUxN2U1Z", "amount": 15.32, "assocComData": "any value" }</pre>
--	---

POST	/api/v1/burn
Issue a burn request that is bound to a particular fungible token account.	
Parameters	
No parameters.	
Request body	
<pre>{ "accountID": "YTcyMD1hZWI5NmUxN2U1Z", "amount": 5.32, "assocComData": "any value", }</pre>	
Responses	
Code	Description
201	Successful burn request issuance.
	Example Value
	ft:burn:ZdkwYzN1N2IxYThh

POST	/api/v1/burn/{burnid}/execute
Attempt to execute a burn request based on its input identifier.	
Parameters	
Name	Description
burnID string <i>(path)</i>	The identifier of the burn request.
Responses	
Code	Description
200	OK response.

GET	/api/v1/burns
Retrieve all burn requests.	

Parameters	
No parameters.	
Responses	
Code	Description
200	OK response.
	Example Value <pre>[{ "accountID": "YTcyMDlhZWI5NmUxN2U1Z", "amount": 5.32, "assocComData": "any value" }, { "accountID": "YTcy65y35WI5NmUxN2U1Z", "amount": 15.32, "assocComData": "any value" }]</pre>

Project & Investment Handling Service

In this section, we provide a detailed description of the API specifications for a service that allows the management of crowdfunding projects and their associated investments throughout their lifecycle. This service provides a user-friendly interface for the creation, as well as for the monitoring and management of their associated investments. The API specifications outlined in this section allow third-party applications to integrate with this service, enabling developers to leverage its features to build their own applications and services that interact with crowdfunding projects and investments. The API specifications are designed to be simple, easy-to-use, and flexible, providing developers with a wide range of capabilities to manage crowdfunding projects and investments.

POST	/api/v1/projects
Create a new crowdfunding project.	
Parameters	
No parameters.	
Request body	
<pre>{ "id": "91160ec7-1f30-425c-bc39-a0b467e98055", "ownerId": "8df90dca-2ec9-4f29-858e-110d61f5d719", "ftAccountId": "8558401d-2dcc-473e-838b-b739866800ae", "funding": { "goal": 5000000.50, "deadline": "2023-03-12T08:35:53Z", "extend_until": "2023-03-15T08:35:53Z", "min_investment": 50 } }</pre>	

<pre>} }</pre>	
Responses	
Code	Description
201	Created response.
	Example Value <pre>{ "id": "91160ec7-1f30-425c-bc39-a0b467e98055", "ownerId": "8df90dca-2ec9-4f29-858e-110d61f5d719", "ftAccountId": "8558401d-2dcc-473e-838b-b739866800ae", "funding": { "goal": 5000000.50, "deadline": "2023-03-12T08:35:53Z", "extend_until": "2023-03-15T08:35:53Z", "min_investment": 50 }, "state": "draft", "created_at": "2023-03-08T08:35:53Z", "updated_at": "2023-03-08T08:35:53Z" }</pre>

POST	/api/v1/projects/{id}/state/{state}
Advance, or modify the state of a crowdfunding project, according to the FSM's description and state transition policies.	
Parameters	
Name	Description
id <small>* required</small> string <i>(path)</i>	The identifier of the crowdfunding project.
state <small>* required</small> string <i>(path)</i>	The target state of the crowdfunding project's FSM.
Responses	
Code	Description
200	OK response.

POST	/api/v1/investments/intent
Submit an investment intent towards a crowdfunding project.	
Parameters	
No parameters.	
Request body	

```
{
  "id": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3",
  "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055",
  "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873",
  "amount": 55,
  "payment_type": "paypal"
}
```

Responses

Code	Description
------	-------------

201	Created response.
-----	-------------------

Example Value

```
{
  "id": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3",
  "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055",
  "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873",
  "amount": 55,
  "payment_type": "paypal",
  "state": "intent",
  "created_at": "2023-03-08T08:35:53Z",
  "updated_at": "2023-03-09T08:35:53Z"
}
```

POST

/api/v1/investments/ft

Submit a fungible token-based investment towards a crowdfunding project.

Parameters

No parameters.

Request body

```
{
  "id": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3",
  "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055",
  "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873",
  "amount": 55,
  "ftAccountId": "abe4fac1-9c35-4092-8f89-02b7df819e0a"
}
```

Responses

Code	Description
------	-------------

201	Created response.
-----	-------------------

Example Value

```
{
  "id": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3",
  "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055",
  "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873",
  "amount": 55,
  "state": "reserved",
  "created_at": "2023-03-08T08:35:53Z",
  "updated_at": "2023-03-09T08:35:53Z"
}
```

	<pre> "payment_type": "ft", "ftAccountId": "abe4fac1-9c35-4092-8f89-02b7df819e0a", "holdId": "ft:hold:8ecf0e3b-ab8c-4d9d-9731-08b195622fa4" } </pre>
--	--

POST	/api/v1/investments/{id}/state/{state}
Advance, or modify the state of an investment, according to the FSM's description and state transition policies.	
Parameters	
Name	Description
id <small>* required</small> string (path)	The identifier of the investment.
state <small>* required</small> string (path)	The target state of the investment's FSM.
Responses	
Code	Description
200	OK response.

GET	/api/v1/projects/{id}
Get a project by its identifier.	
Parameters	
Name	Description
id <small>* required</small> string (path)	The identifier of the crowdfunding project.
Responses	
Code	Description
200	OK response.
	Example Value <pre> { "id": "91160ec7-1f30-425c-bc39-a0b467e98055", "ownerId": "8df90dca-2ec9-4f29-858e-110d61f5d719", "ftAccountId": "8558401d-2dcc-473e-838b-b739866800ae", "funding": { "goal": 5000000.50, "deadline": "2023-03-12T08:35:53Z", "extend_until": "2023-03-15T08:35:53Z", "min_investment": 50 }, "state": "draft", </pre>

	<pre>"created_at": "2023-03-08T08:35:53Z", "updated_at": "2023-03-08T08:35:53Z" }</pre>
GET	/api/v1/projects
Retrieve all crowdfunding projects from the ledger.	
Parameters	
No parameters.	
Responses	
Code	Description
200	OK response.
	Example Value
	<pre>[{ "id": "91160ec7-1f30-425c-bc39-a0b467e98055", "ownerId": "8df90dca-2ec9-4f29-858e-110d61f5d719", "ftAccountId": "8558401d-2dcc-473e-838b-b739866800ae", "funding": { "goal": 5000000.50, "deadline": "2023-03-12T08:35:53Z", "extend_until": "2023-03-15T08:35:53Z", "min_investment": 50 }, "state": "draft", "created_at": "2023-03-08T08:35:53Z", "updated_at": "2023-03-08T08:35:53Z" }]</pre>

GET	/api/v1/projects/{id}/investments
Retrieve all investments associated with a specific project from the ledger.	
Parameters	
Name	Description
id string (path)	The crowdfunding project's identifier.
Responses	
Code	Description
200	OK response.
	Example Value
	<pre>[{ "id": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3", </pre>

	<pre> "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055", "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873", "amount": 55, "payment_type": "paypal", "state": "intent", "created_at": "2023-03-08T08:35:53Z", "updated_at": "2023-03-09T08:35:53Z" }] </pre>
--	---

GET	/api/v1/investments/{id}
Get an investment by its identifier from the ledger.	
Parameters	
Name	Description
id <small>* required</small> string (path)	The investment's identifier.
Responses	
Code	Description
200	OK response.
	Example Value
	<pre> { "id": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3", "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055", "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873", "amount": 55, "payment_type": "paypal", "state": "intent", "created_at": "2023-03-08T08:35:53Z", "updated_at": "2023-03-09T08:35:53Z" } </pre>

Investment Tokenization Service

In any equity crowdfunding platform, the management and proof of ownership of investment certificates are of paramount importance. These certificates represent a non-fungible asset, which implies that each certificate is unique and can't be replaced by another certificate with an equal value. In the proposed blockchain-based equity crowdfunding platform, non-fungible tokens (NFTs) can be used to represent these investment certificates, and a dedicated service can provide the necessary functionality to manage and interact with them. This section presents the API specifications of a service that is responsible for the management and proof of ownership of investment certificates in the proposed platform. The service will allow the issuance of new certificates and query their ownership status, among other functionalities.

POST	/api/v1/investments/mint
Mint a new investment NFT.	
Parameters	
No parameters.	
Request body	
<pre>{ "id": "c58e6c29-0eaf-49ec-ac65-39de", "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873", "investmentId": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3", "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055", "amount": 55, "metadata": { "algo": "SHA256", "tokenURI": "uri:nft:c58e6c29-0eaf-49ec-ac65-39de", "hash": "eDUwOTpUZXN0TVNQSUQ6ZURVd09U" } }</pre>	
Responses	
Code	Description
201	Created response.
	Example Value
	<pre>{ "id": "c58e6c29-0eaf-49ec-ac65-39de", "type": "investment", "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873", "investmentId": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3", "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055", "amount": 55, "metadata": { "algo": "SHA256", "tokenURI": "uri:nft:c58e6c29-0eaf-49ec-ac65-39de", "hash": "eDUwOTpUZXN0TVNQSUQ6ZURVd09U" }, "minted_at": "2023-03-08T08:35:53Z" }</pre>

GET	/api/v1/investments
Retrieve all investment NFTs.	
Parameters	
No parameters.	
Responses	
Code	Description
200	OK response.

	Example Value <pre>[{ "id": "c58e6c29-0eaf-49ec-ac65-39de", "type": "investment", "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873", "investmentId": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3", "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055", "amount": 55, "metadata": { "algo": "SHA256", "tokenURI": "uri:nft:c58e6c29-0eaf-49ec-ac65-39de", "hash": "eDUwOTpUZXN0TVNQSUQ6ZURVd09U" }, "minted_at": "2023-03-08T08:35:53Z" }]</pre>
--	--

GET	/api/v1/investments/{id}
Retrieve an investment NFT.	
Parameters	
Name	Description
id <small>*required</small> string <i>(path)</i>	The investment NFT's identifier.
Responses	
Code	Description
200	OK response.
	Example Value <pre>{ "id": "c58e6c29-0eaf-49ec-ac65-39de", "type": "investment", "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873", "investmentId": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3", "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055", "amount": 55, "metadata": { "algo": "SHA256", "tokenURI": "uri:nft:c58e6c29-0eaf-49ec-ac65-39de", "hash": "eDUwOTpUZXN0TVNQSUQ6ZURVd09U" }, "minted_at": "2023-03-08T08:35:53Z" }</pre>

GET	/api/v1/investments/owned
Retrieve all investment NFTs owned by the invoking user.	
Parameters	

No parameters.	
Responses	
Code	Description
200	OK response.
	Example Value <pre>[{ "id": "c58e6c29-0eaf-49ec-ac65-39de", "type": "investment", "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873", "investmentId": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3", "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055", "amount": 55, "metadata": { "algo": "SHA256", "tokenURI": "uri:nft:c58e6c29-0eaf-49ec-ac65-39de", "hash": "eDUwOTpUZXN0TVNQSUQ6ZURVd09U" }, "minted_at": "2023-03-08T08:35:53Z" }]</pre>

GET	/api/v1/investments/{id}/owner
Retrieve the identifier of the owner of an investment NFT.	
Parameters	
Name	Description
id <small>* required</small> string <i>(path)</i>	The investment NFT's identifier.
Responses	
Code	Description
200	OK response.
	Example Value <pre>664f57a1-e5e2-4230-9d48-611c0229b873</pre>

Equity Tokenization Service

This section outlines the API specifications of a service that handles the management and (proof of) ownership of equity certificates in the proposed blockchain-based equity crowdfunding platform. Equity certificates are modeled as non-fungible tokens (NFTs) that represent a share of ownership in a particular crowdfunding project. This service provides functionalities related to the issuance and tracking of these NFTs, as well as the

verification of ownership and associated metadata. The API specifications of this service are essential for implementing the core functionalities of the platform that enable investors to invest in crowdfunding projects and receive their equity shares in a secure and transparent manner.

POST /api/v1/equities/mint	
Mint a new equity NFT.	
Parameters	
No parameters.	
Request body	
<pre>{ "id": "d972ccca-5e04-40c8-a7bd-4035", "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873", "investmentId": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3", "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055", "percentage": 10.50, "metadata": { "algo": "SHA256", "tokenURI": "uri:nft:d972ccca-5e04-40c8-a7bd-4035", "hash": "eDUwOTpUZXN0TVNQSUQ6ZURVd09U" } }</pre>	
Responses	
Code	Description
201	Created response.
	Example Value <pre>{ "id": "d972ccca-5e04-40c8-a7bd-4035", "type": "equity", "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873", "investmentId": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3", "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055", "percentage": 10.50, "metadata": { "algo": "SHA256", "tokenURI": "uri:nft:d972ccca-5e04-40c8-a7bd-4035", "hash": "eDUwOTpUZXN0TVNQSUQ6ZURVd09U" }, "minted_at": "2026-05-12T08:35:53Z" }</pre>

GET /api/v1/equities	
Retrieve all equity NFTs.	
Parameters	

No parameters.	
Responses	
Code	Description
200	OK response.
	Example Value <pre>[{ "id": "d972ccca-5e04-40c8-a7bd-4035", "type": "equity", "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873", "investmentId": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3", "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055", "percentage": 10.50, "metadata": { "algo": "SHA256", "tokenURI": "uri:nft:d972ccca-5e04-40c8-a7bd-4035", "hash": "eDUwOTpUZXN0TVNQSUQ6ZURVd09U" }, "minted_at": "2026-05-12T08:35:53Z" }]</pre>

GET	/api/v1/equities/{id}
Retrieve an equity NFT.	
Parameters	
Name	Description
id <small>* required</small> string <i>(path)</i>	The equity NFT's identifier.
Responses	
Code	Description
200	OK response.
	Example Value <pre>{ "id": "d972ccca-5e04-40c8-a7bd-4035", "type": "equity", "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873", "investmentId": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3", "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055", "percentage": 10.50, "metadata": { "algo": "SHA256", "tokenURI": "uri:nft:d972ccca-5e04-40c8-a7bd-4035", "hash": "eDUwOTpUZXN0TVNQSUQ6ZURVd09U" }, "minted_at": "2026-05-12T08:35:53Z" }</pre>

GET		/api/v1/equities/owned
Retrieve all equity NFTs owned by the invoking user.		
Parameters		
No parameters.		
Responses		
Code	Description	
200	OK response.	
	Example Value <pre>[{ "id": "d972ccca-5e04-40c8-a7bd-4035", "type": "equity", "investorId": "664f57a1-e5e2-4230-9d48-611c0229b873", "investmentId": "71e858cc-4d79-42b2-a3fe-3c45e5e3e8c3", "projectId": "91160ec7-1f30-425c-bc39-a0b467e98055", "percentage": 10.50, "metadata": { "algo": "SHA256", "tokenURI": "uri:nft:d972ccca-5e04-40c8-a7bd-4035", "hash": "eDUwOTpUZXN0TVNQSUQ6ZURVd09U" }, "minted_at": "2026-05-12T08:35:53Z" }]</pre>	

GET		/api/v1/equities/{id}/owner
Retrieve the identifier of an equity NFT’s owner.		
Parameters		
Name	Description	
id <small>required</small> string (path)	The equity NFT’s identifier.	
Responses		
Code	Description	
200	OK response.	
	Example Value	
	664f57a1-e5e2-4230-9d48-611c0229b873	

References

- [1] "Kickstarter," Kickstarter, [Online]. Available: <https://www.kickstarter.com/>. [Accessed 22 10 2021].
- [2] "Indiegogo: Crowdfund Innovations & Support Entrepreneurs," Indiegogo, [Online]. Available: <https://www.indiegogo.com/>. [Accessed 22 10 2021].
- [3] "Patreon," Patreon, [Online]. Available: <https://www.patreon.com/>. [Accessed 22 11 2021].
- [4] "GoFundMe," GoFundMe, [Online]. Available: <https://www.gofundme.com/>. [Accessed 25 10 2021].
- [5] "Chuffed | Non-profit charity and social enterprise fundraising," Chuffed, [Online]. Available: <https://chuffed.org/eu>. [Accessed 25 10 2021].
- [6] "ArtistShare," ArtistShare, [Online]. Available: <https://www.artistshare.com/>. [Accessed 25 10 2021].
- [7] "Bettervest | your money, your impact," Bettervest, [Online]. Available: <https://www.bettervest.com/en/>. [Accessed 26 10 2021].
- [8] "GreenVesting: Crowdfunding für nachhaltige Energieprojekte," GreenVesting, [Online]. Available: <https://www.greenvesting.com/>. [Accessed 26 10 2021].
- [9] "Mobilise your money for good - Abundance," Abundance, [Online]. Available: <https://www.abundanceinvestment.com/>. [Accessed 26 10 2021].
- [10] "Windcentrale," [Online]. Available: <https://www.windcentrale.nl/>. [Accessed 28 10 2021].
- [11] "Econeers | Crowdfunding-Plattform," Econeers, [Online]. Available: <https://www.econeers.de/>. [Accessed 28 10 2021].
- [12] "Lendopolis," Lendopolis, [Online]. Available: <https://www.lendopolis.com/>. [Accessed 28 10 2021].
- [13] "1miljoenwatt," 1miljoenwatt.nl, [Online]. Available: <https://www.1miljoenwatt.nl/>. [Accessed 28 10 2021].
- [14] "Citizenergy," Citizenergy, [Online]. Available: <https://citizenergy.eu/>. [Accessed 29 10 2021].
- [15] "HTML," [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTML>. [Accessed 15 03 2022].
- [16] "CSS," [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/CSS>. [Accessed 15 03 2022].
- [17] "Python," [Online]. Available: <https://www.python.org/>. [Accessed 15 03 2022].
- [18] "Angular," [Online]. Available: <https://angular.io/>. [Accessed 15 03 2022].
- [19] "node.js," [Online]. Available: <https://nodejs.org/en>. [Accessed 12 01 2023].
- [20] "Flask," [Online]. Available: <https://flask.palletsprojects.com/en/2.0.x/>. [Accessed 15 03 2022].
- [21] "PostgreSQL," [Online]. Available: <https://www.postgresql.org/>. [Accessed 15 03 2022].
- [22] "MongoDB," [Online]. Available: <https://www.mongodb.com/>. [Accessed 2023 02 11].
- [23] "RFC 6749: The OAuth 2.0 Authorization Framework," [Online]. Available: <https://www.rfc-editor.org/rfc/rfc6749>. [Accessed 13 2023].



- [24] "RFC 6750: The OAuth 2.0 Authorization Framework: Bearer Token," [Online]. Available: <https://www.rfc-editor.org/rfc/rfc6750>. [Accessed 01 03 2023].

