INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Web3 Ticket

ANDRÉ FILIPE MONIZ MORAIS
Junho de 2023

POLITÉCNICO
DO PORTO

Web3 Ticket

André Morais

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science, Specialisation Area of Computer Systems

Supervisor: Dr. Jorge Pinto Leite & Co-orientador (Dr. Nuno Silva)

Porto, June 30, 2023

Statement of Integrity

I hereby declare having conducted this academic work with integrity.

I have not plagiarised or applied any form of undue use of information or falsification of results along the process leading to its elaboration.

Therefore the work presented in this document is original and authored by me, having not previously been used for any other end.

I further declare that I have fully acknowledged the Code of Ethical Conduct of P.PORTO.

ISEP, Porto, June 30, 2023

Abstract

The present work aims to make the transition from Web 2.0 to Web 3.0 in the ticket-selling market. For this purpose, the use of Blockchain technology to implement the system in question is justified, based on transparency, security, reduced costs, and traceability, advantages that are taken into account when developing the system.

Non-Fungible Tokens (NFTs) will have a strong role in the task of defining the concept of the project because it is the element that bridges the gap between the traditional and the decentralized market, as they have similarities with tickets, such as uniqueness, for allowing verification of identity and for guaranteeing to belong to an individual.

Research is also made of current knowledge about blockchain, of projects carried out in the artistic area that took advantage of this technology, of the problems that can be encountered in terms of security, and the justification for choosing one blockchain among others. Finally, the added value that the project has in the market in which it fits is demonstrated, through analytical processes to support decision-making, and the planning of experimentation of the developed solution is carried out, following strategies of experiments and tests with the target audiences.

The project can be considered a success, thanks to the successful development and implementation of the key requirements, as well as the positive evaluation provided by the users who tested the application.

Keywords: Blockchain, Non-Fungible Tokens, Tickets, Show, Cardano, Marketplace

Resumo

O presente trabalho tem como objetivo fazer a transição da Web 2.0 em Web 3.0 no mercado da venda de bilhetes. Para o efeito é justificada a utilização da tecnologia Blockchain para implementar o sistema em causa, tendo como base a transparência, a segurança, os custos reduzidos e a rastreabilidade, vantagens que são tidas em conta ao desenvolver o sistema.

Os Non-Fungible Tokens (NFTs) terão um forte paper na definição do conceito do projeto porque é o elemento que faz a ponte entre o mercado tradicional e o descentralizado, por terem semelhanças com bilhetes, passes VIP (very important person), diplomas ou certificados, tais como, a irrepetibilidade, por permitem a verificação de identidade e por garantirem a pertença a um indivíduo. Desta forma, com a utilização de NFTs, pode ser armazenada informação relevante, como o assento a que o bilhete corresponde, é possível garantir que o bilhete pertence a uma pessoa específica e é praticamente impossível burlar os compradores, salvo contacto direto com as private keys (PK) que geram o bilhete ou negligência do comprador.

É também feito um levantamento do conhecimento atual sobre blockchain, dos projetos realizados na área artística que tiram proveito desta tecnologia, dos problemas que podem ser encontrados a nível de segurança e a justificação de escolha de uma blockchain entre as outras. Por fim, é demonstrado o valor acrescentado que o projeto tem no mercado em que se enquadra, através de processos analíticos de apoio à tomada de decisão, e faz-se um planeamento de experimentação da solução desenvolvida, seguindo estratégias de questionários e testes com o público alvo.

O projeto pode ser considerado um sucesso, graças ao sucesso no desenvolvimento e implementação dos principais requisitos, bem como à avaliação positiva dos utilizadores que testaram a aplicação.

Contents

List of Figures **vii**

List of Tables **ix**

List of Symbols **xi**

List of Acronyms **xii**

1 Introduction **1**
1.1 Problem Statement . . . 1
1.1.1 Objectives . . . 2
1.1.2 Hypothesis . . . 2
1.2 Plan . . . 3
1.3 Research Methodology & Questions . . . 3

2 Background **5**
2.1 What's a decentralized system? . . . 5
2.2 What's the blockchain role? . . . 6
2.3 Unspent transaction output VS Account-Based blockchains . . . 8
2.3.1 Unspent transaction output . . . 8
2.3.2 Account-based . . . 10
2.4 Real capabilities of NFTs . . . 10

3 State of Art **12**
3.1 Part I . . . 12
3.1.1 Intellectual Property . . . 12
3.1.2 Art Marketplace . . . 14
3.1.3 Non-Fungible Tokens as Event Tickets . . . 15
3.1.4 Security Issues . . . 16
3.2 Part II . . . 17
3.2.1 Most used blockchains . . . 17
3.2.2 Cardano-powered ticket marketplace . . . 19

4 Value Analysis **22**
4.1 New Concept Development . . . 22
4.2 Opportunity identification & analysis . . . 24
4.2.1 Competitors & Market Segment . . . 25
4.3 The idea behind the concept . . . 26
4.3.1 Idea Generation . . . 26
4.3.2 Ideal Selection . . . 26
Decision Making . . . 27

4.3.3 Concept Definition . . . 34
4.4 Value . . . 35
4.4.1 Value Proposition . . . 37
Business Model Canvas . . . 38
4.4.2 Quality Function Development . . . 38

5 Development Process and Implementation 42
5.1 Development Methodology . . . 42
5.2 System Design . . . 43
5.2.1 Database architecture . . . 44
5.2.2 Ticket generation . . . 46
5.2.3 Marketplace architecture . . . 47
Marketplace - Sell a ticket . . . 47
Marketplace - Buy a ticket . . . 49
5.3 User Interface Design . . . 50
5.4 Implementation Details . . . 52
5.4.1 Create & Restore native wallet . . . 52
5.4.2 Consult Balance . . . 53
5.4.3 See and Copy wallet address . . . 53
5.4.4 Create Shows . . . 54
5.4.5 Accept & Refuse Shows . . . 57
5.4.6 Show Details & Redirect to map . . . 58
5.4.7 Buy a ticket in primary market . . . 59
5.4.8 Get all the wallet tickets . . . 65
5.4.9 Sell Ticket - Add to Marketplace . . . 67
5.4.10 Buy Ticket from Marketplace . . . 69
5.4.11 See ticket's qr code . . . 70
5.4.12 Scan & Validate user's ticket . . . 71
5.4.13 Logout . . . 72
5.5 Limitations . . . 73

6 Experimentation and Evaluation 74
6.1 Use Cases . . . 74
6.1.1 Artist . . . 75
Create shows . . . 75
Follow ticket sales . . . 75
Ticket distribution system . . . 75
Get the list of created shows . . . 75
6.1.2 Fan . . . 75
Get all available shows . . . 75
Buy a ticket . . . 76
Get owned tickets . . . 76
Validate a ticket at the entrance . . . 76
6.1.3 Organizer . . . 76
Validate a ticket at the entrance . . . 76
Follow the place's stats . . . 76
Accept/refuse shows for a certain day . . . 76
6.2 Methodology and Evaluation . . . 76
6.2.1 Quantitative Evaluation Framework . . . 79

6.3 The Results 82
6.4 System Usability Scale 87

7 Future Work **90**

8 Conclusion **91**

A Database Diagram **96**

List of Figures

1.1 Web3 Ticket MindMap . . . 3

2.1 Blockchain blocks diagram . . . 5
2.2 Blockchain UTXOs Diagram . . . 9

4.1 Innovation Phases . . . 23
4.2 New Concept Development Diagram . . . 23
4.3 SWOT analysis: Strengths, Weaknesses, Opportunity, Threats . . . 24
4.4 The Hierarchy for Evaluating Blockchain Selection: this identifies the decision problem and defines the objectives and alternatives. . . . 27
4.5 Relative Importance: table matching the importance to its value . . . 29
4.6 Random Index Table . . . 31
4.7 Process of buying a ticket directly to the artist and using it in the show. This is the core process of this project. . . . 35
4.8 Customer Value: benefits and sacrifices . . . 37
4.9 Value Proposition . . . 37
4.10 Canvas Model . . . 39
4.11 House of Quality . . . 40

5.1 Software Architecture . . . 43
5.2 Overview of the database diagram . . . 44
5.3 Relations between Shows entity and its Foreign Keys . . . 44
5.4 Relations between User entity and the entities related to it . . . 45
5.5 Relations between Place entity and the entities related to it . . . 45
5.6 Relations between Ticket entity and the entities related to it and its Foreign Keys . . . 45
5.7 Diagram of ticket minting/creation . . . 46
5.8 Marketplace building . . . 47
5.9 Secondary market sale workflow . . . 48
5.10 Secondary market buy workflow . . . 49
5.11 Web3Ticket logo . . . 50
5.12 Switch to artist view . . . 51
5.13 Switch to Place Responsible . . . 51
5.14 Seed phrase being shown to the user . . . 52
5.15 Wallet details . . . 53
5.16 Screen where the artist can create the show . . . 55
5.17 Workflow of accepting/refusing a request from an artist . . . 57
5.18 Details of an event are shown to the user . . . 58
5.19 Buy directly to the artist . . . 59
5.20 Wallet screen showing the balance, address and assets . . . 66
5.21 Sell an asset from wallet . . . 67
5.22 Buy an asset from marketplace . . . 69

5.23 Generate a QR Code to be validated . . . 70
5.24 Logout button . . . 72

6.1 Business Use Cases Diagram . . . 74
6.2 How easy was the process of creating/restoring a wallet? . . . 82
6.3 After creating/restoring your wallet you were asked to get the address of this wallet. Were you able to get your address? . . . 82
6.4 How easy was to get the address of your wallet? . . . 82
6.5 How easy is to understand where the catalog is located? . . . 82
6.6 Could you see the show details? . . . 83
6.7 Try to know where the Place is located. Were you able of opening the google maps to see the place's location? . . . 83
6.8 Try to buy a ticket from the primary sale. How easy was to find the button? 83
6.9 Could you choose a seat (or more) and complete the purchase? . . . 83
6.10 Can you see, on the wallet page, the ticket you purchased? . . . 83
6.11 Try to sell the ticket. How easy it was to find the ticket sale feature? . . . 83
6.12 Did you managed to sell the ticket? Please remember that it can take some time to the blockchain to validate the transaction? . . . 83
6.13 If the ticket disappeared from your wallet, we are ready to move on. Try to find the marketplace where your ticket is listed. How easy was the process of finding your ticket in the marketplace? . . . 83
6.14 Was you able to find the available tickets to buy from second market? . . . 84
6.15 How easy was to complete the process of buying the ticket from second market? . . . 84
6.16 Wait a few moments for the blockchain to validate your transaction. Is the ticket in your wallet? . . . 84
6.17 Choose a ticket to validate. Follow the steps on the screen to activate the ticket. Were you able to complete this process? Once again, the transaction needs to be validated by the blockchain, so it can take some minutes. . . . 84
6.18 Try to logout from your account. Did you managed to do it? . . . 84
6.19 (Artist) Try to create a show, how easy was to find the functionality and to create the show? . . . 84
6.20 (Artist) After creating a show, were you able to consult the created ones? 84
6.21 (Artist) How easy is it to search for shows in the gallery? . . . 84
6.22 (Artist) Try to logout from your account. Did you managed to do it? . . . 85
6.23 (Place) In the homepage can you see the requests for scheduling? . . . 85
6.24 (Place) How easy it is to accept/refuse shows? . . . 85
6.25 (Place) Can you see the show details? . . . 85
6.26 (Place) Its now time to scan a ticket at the entrance. In another device, open the qrcode of one ticket and try to scan it. How easy was to access the QR code reader? . . . 85
6.27 (Place) How easy it is to scan & validate the user's ticket? . . . 85
6.28 (General) Do you think all the messages are easy to understand? . . . 85
6.29 (General) Do you think the app provides enough information to the users to start managing their tickets? . . . 85

A.1 Overview of the database diagram . . . 96

List of Tables

4.1 Pairwise comparison table 28
4.2 Pairwise comparison table: Columns sum 28
4.3 Pairwise comparison table: normalization 29
4.4 Pairwise comparison table: Criteria Weights calculation 29
4.5 Pairwise comparison table: initial pairwise table multiplied by criteria weights 30
4.6 Pairwise comparison table: ratio calculation 30
4.7 Comparison table between criteria and alternatives 31
4.8 Comparison table between criteria and alternatives: normalization 32
4.9 Comparison table between criteria and alternatives: applying the weights . 33
4.10 Comparison table between criteria and alternatives: calculation of the Ideal Best and Ideal Worst 33
4.11 Comparison table between criteria and alternatives: calculation of Euclidean Distance 33
4.12 Comparison table between criteria and alternatives: Performance Score and Rank 34

6.1 Functional requirements table: matching the actor to its desire 78
6.2 Functional requirements table: core components matching hypothesis . . . 79
6.3 QEF table: core functional requirements matching weight and completeness level 80
6.4 QEF table: core adaptability requirements matching weight and completeness level 81
6.5 QEF table: core usability requirements matching weight and completeness level 81
6.6 QEF table: core adaptability requirements matching SUS and SUS Score . 87
6.7 QEF table: core functional requirements matching SUS and SUS Score . . 88
6.8 QEF table: core usability requirements matching SUS and SUS Score . . . 89

List of Codes

5.1 Function to store user's information in device's storage 53
5.2 Return from Blockfrost's endpoint 53
5.3 Function to gather and format the user's wallet address 54
5.4 Code used to copy the address 54
5.5 Server side code to upload image into the IPFS network 56
5.6 Server side code to create policy keys 56
5.7 Server side code to return the show list 58
5.8 Code to open google maps on the Show Details Page 59
5.9 Function that gathers all the available seats 60
5.10 Server side code to filter out the reserved seats 60
5.11 Code that controls the way that selected seats are presented in the app . . 61
5.12 Server side code to check user's balance 61
5.13 Server side code to load the required keys 62
5.14 Server side code to load the policy 62
5.15 Server side code to define the NFTs being minted 62
5.16 Server side function to generate the metadata for the ticket 63
5.17 Server side code to construct the transaction 63
5.18 Server side code to set all the outputs of the transaction 64
5.19 Server side code to create the list of witnesses 65
5.20 Server side code to submit the transaction to the blockchain 65
5.21 Function to get all the user's tickets 66
5.22 Recursive function to gather all the tickets' details 67
5.23 Server side code to generate the selling wallet keys 68
5.24 Server side code to generate the metadata for the '_info' nfts 68
5.25 Server side code to define the objects that represent the burn of the nft in the blockchain 69
5.26 Code that allows showing the qrcode to the user 71
5.27 Server side code to check if the ticket was burned and if the owner is the user 71
5.28 Representation of the returning value of Blockfrost endpoint 72

List of Symbols

CR	Consistency Ratio
CI	Consistency Index
RI	Random Index
λ_{MAX}	average of the ratios
P_i	Performance Score
V_j^+	positive ideal value
V_j^-	negative ideal value
S_i^+	best Euclidean Distance
S_i^-	worst Euclidean Distance

List of Acronyms

ACG	ArtChain Global.
ACGT	ArtChain Global Token.
AES	Advanced Encryption Standard.
AHP	Analytic Hierarchy Process.
AIEEE	Advances in Information, Electronic and Electrical Engineering.
BFT	Byzantine Fault Tolerance.
BSC	Binance Smart Chain.
BTC	Bitcoin.
CBOR	Concise Binary Object Representation.
CC	Creative Commons.
CCL	Cardano Computing Layer.
CMOA	Carnegie Museum of Art.
CSL	Cardano Settlement Layer.
dApps	Decentralized Applications.
DBMS	Database Management System.
DEI	Departamento Engenharia Informática.
DPoS	Delegated Proof of Stake.
ETH	Ether.
eUTXO	Extended Unspent Transaction Output.
GETH	Go-Ethereum.
HDWallet	Hierarchical Deterministic Wallet.
HOQ	House of Quality.
HTML	Hypertext Markup Language.
ICEEOT	International Conference on Electrical, Electronics, and Optimization Techniques.
IoT	Internet of Things.
IPFS	InterPlanetary File System.
IPP	Instituto Politécnico do Porto.
ISEP	Instituto Superior de Engenharia do Porto.
MCDM	Multiple Criteria Decision Making.
NCD	New Concept Development.

NFT	Non-Fungible Token.
NPD	New Product Development.
O/I	Outputs/Inputs.
PoA	Proof-of-Authority.
PoAc	Proof of Activity.
PoAu	Proof of Authority.
PoB	Proof of Burn.
PoC	Proof of Capacity.
PoET	Proof of Elapsed Time.
PoI	Proof of Importance.
PoW	Proof of Stake.
PROMETHEE	Preference Ranking Organization Method for Enrichment of Evaluations.
PWA	Progressive Web Apps.
QEF	Quantitative Evaluation Framework.
QFD	Quality Function Development.
RD	Research and Development.
RPCA	Ripple Protocol Consensus Algorithm.
SAW	Simple Additive Weighting.
SPOOL	Secure Public Online Ownership Ledger.
SPV	Simplified Payment Verification.
TOPSIS	Technique for Order of Preference by Similarity to Ideal Solution.
TSG	Technological Stage Gate.
UI	User Interface.
UNL	Unique Node List.
UTXO	Unspent Transaction Output.
UX	User Experience.
VIP	Very Important Person.
VOC	Voice of the Customer.

Chapter 1

Introduction

Nowadays the current business model for ticket selling is based on many entities apart from the artist, the fan, and the place where the artist will perform. Despite that, the artist has a decreased decision power and many ways of losing money in the process: renting the place, paying the manager to schedule the show, and losing the fee to the company that sells the tickets. In addition to that, although it is the fans who buy the ticket with their own money, they are not allowed to sell it because the law has several constraints about it. There is another problem associated with selling tickets in the secondary market which is the fact that, for very famous artists, there are people taking advantage of it, forging tickets that don't give access to the show.

The root of the problem is the inadequate technology behind this system and this work proposes to migrate into a decentralized version of the current system, redistributing the work between 3 entities: the artist, the fan, and the place organization, where only they define their own limits.

1.1 Problem Statement

Normally artists are accompanied by an agent that charges a fee to take care of the scheduling and it can be admitted in an empirical form that showrooms charge a high amount of money so the artists can welcome their audience. When added the fee that the ticket distribution company charges, the success of artists is even more hampered.

Adding to this problem, and relating to the secondary sales, it is impossible to identify counterfeit ticket selling, control scalping, and price gouging, and combat the lack of transparency, since neither the companies responsible for emitting the tickets, the showrooms, nor the artists can control the publication of classified selling posts for ticket reselling. As stated by The Treasury of The Australian Government (2017), "It is important to know who the official ticket seller is for an event because it ensures that consumers can obtain tickets from an official ticket seller that are guaranteed to be valid for entry into the event." [32]

Counterfeit Tickets are one of the biggest problems in the ticketing industry because some people may try to sell fake tickets as genuine, on unregulated markets, leading to complaints from fans. Scalping and price gouging refer to the practice of reselling tickets at inflated prices, often well above their face value. This can make it difficult for genuine fans to purchase tickets at a reasonable price. Lack of Transparency is the problem associated with the challenges in verifying the authenticity of tickets or determining the true market value.

Regarding validation, the ticket ownership should be verified, and in the current models, the verification is easily bypassed.

1.1.1 Objectives

The objective of this project is to design and develop, in the modern technologies of Web3, a solution that allows the artist, the fan, and the showroom responsible to take full advantage of blockchain capabilities, including the possibility of tracing the tickets to ensure the ownership and the validity of the ticket.

And more than that, to create an abstraction between the normal user and the blockchain world, full of concepts. The goal is to serve both blockchain daily users and people that never used the blockchain in a way that inexperienced people don't feel confused about using the software.

The artist will be able to get the statistic over his/her ticket sales, create/schedule shows, distribute the tickets, and consult the created shows and the different dates; the fan (normal user) will be capable of seeing all the available shows, buy and sell a ticket, see the wallet and its assets, and use the app to validate the ticket and entering the event; the showroom responsible will be capable of using the app to validate the tickets at the entrance, get statistics about the enclosure/room depending on being an open space or a room, and accept and refuse requests from artists to perform on a certain day/hour.

1.1.2 Hypothesis

The viability of implementing a solution for ticket selling, based on NFTs and their usage as daily normal is the hypothesis for this project. The idea behind it is the usage of blockchain ledger[1], which is the name given to a distributed database that is shared over all the nodes (devices) in a computer network and that relies on transaction bases.

It will be needed to decentralize the decision power from an individual to a group of them in a dispersed network. Artists are the engine of the system, all the power is given to them and the ones who pay for the extra features are the fans.

In this project, there is almost no computational power required, since all the processing required is in the blockchain. Thus, the hypothesis is a new business model completely autonomous, without the need for a big infrastructure, and where the artist has much more control, with no need for an agent, a process characteristic of the current models.

The feature that most use the blockchain is the one that would be charged to the user: selling tickets in the secondary market with price control to avoid problems related to inflated prices.

It is important to note that when the fan chooses to sell a ticket it can be priced much higher than the initial price. Hence, there are two possibilities: it can be trusted in a self-regulated market or define a price policy. Also, it is to be noted that the purpose of this work is to focus on a community-driven platform, not on a regulatory system, and that tickets are not a basic necessity whereby it is not a priority to adjust the price of the tickets. Despite all this, it will be presented a way of influencing the prices charges in the second market:

- Tickets sold below the price paid for them don't pay any fees
- Tickets cannot be sold 10 times higher than the floor price or the price defined initially by the artist, depending on which one is the lowest.

[1] A book/list of transactions that, in the context of blockchain, have a similar concept as an accounting book of debits and credits

- Tickets cannot be sold 2 times higher than the mean o the listed prices if the floor price is higher than the initial price defined by the artist.

The first measure encourages the sellers to sell below the price of buying, which means that helps to lower the floor price. The second measure creates an interval of possible price points. The third helps to set a lower price if it is being inflated. The second measure is required to limit the average price, i.e, the third measure limits the price to 2 times the average price but if the average is more than 10 times the initial price, the second point forbids it.

1.2 Plan

The first step in planning the development of a new project is brainstorming, for which was created a mind map that serves as a visual representation of information, ideas, and their inter-relation. This involves the use of keywords to represent different ideas and how they are connected.

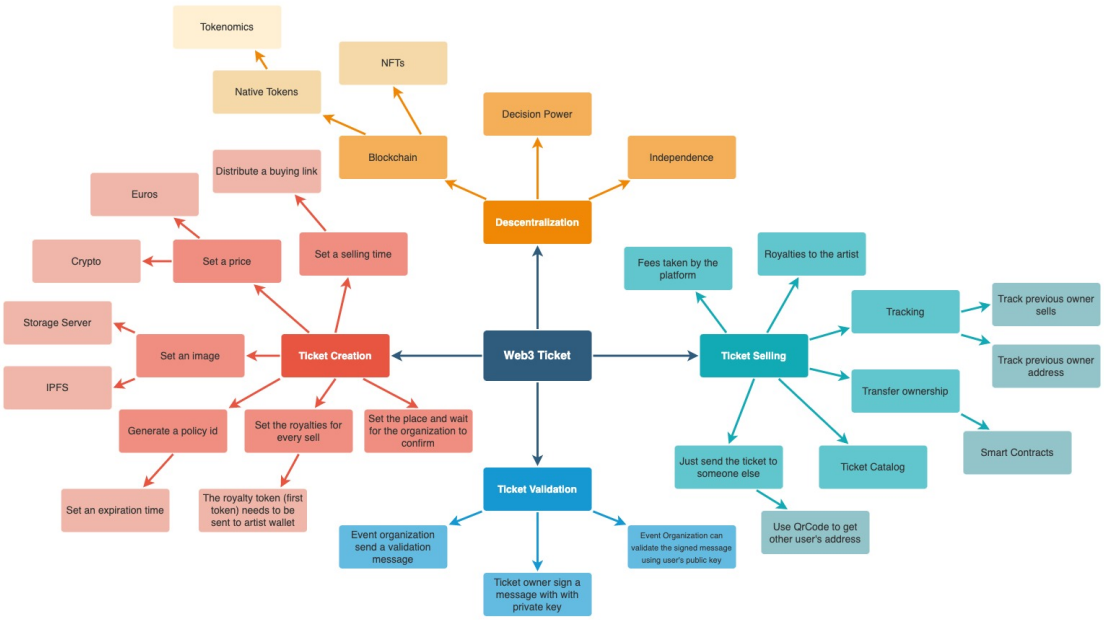


Figure 1.1: Web3 Ticket MindMap

In the center, it is represented the proposal: Web3 Ticket. It branches into four subthemes, each one supporting a different use case like ticket creation, distribution, and selling. For each subtheme are placed concepts that can relate to it. This is useful to break down complex information and organize it in a logical and meaningful way.

1.3 Research Methodology & Questions

The motivation comes from anticipating the future where decentralized apps will rule the market and when trying to answer questions, like "What areas benefit from being decentralized?", "Taking into account that ticket selling can be decentralized, how can it be done?", and "How could NFTs be implemented to decentralize the ticket market?", the idea appeared.

The first step, as shown at the beginning of the chapter, was to create the mind map, so all concepts could be brought together. Then, one by one, the concepts were dissected. The main keywords extracted from the previous task are:

- blockchain
- tickets
- art
- decentralization
- decision power
- copyrights
- selling
- tracking

Prior research was made to list all the possible blockchains that supported NFTs, and for each alternative were investigated each of the valuable characteristics (reliability, transaction cost, environmental sustainability, and implementation simplicity). Platforms like ResearchGate[2] and IEEE Xplore[3] were used with the identified keywords to find good-quality papers about the subject.

The way the papers and online articles were explored was by following 'The 3-Pass Approach' where, after a good paper is found, first its title, publication date and abstract, and introduction must be read. If the theme is relevant, then the research continues to the next step, or else a new paper must be found. The second step is to understand what the paper is telling. If some of the concepts are not familiar, they must be examined. Lastly, if the paper has enough content to be used, then taking notes is a good way of getting the required knowledge [39].

[2] "Find and share research", ResearchGate, https://www.researchgate.net/
[3] IEEE Xplore, https://ieeexplore.ieee.org/

Chapter 2

Background

It is important to address some concepts before properly starting the contextualization. This chapter will describe some technologies to clarify the following chapters regarding the already established projects in this area, beginning with the decentralized systems, how blockchain is implemented in the real world and the idea behind Non-Fungible Tokens.

2.1 What's a decentralized system?

For starters, decentralized systems and distributed systems are concepts that need to be differentiated. Andrew S. Tanenbaum sets a definition to distributed systems, as a "collection of independent computers that appear to the users of the system as a single computer" [70]. Decentralized systems don't specify exactly the same working method because they describe how the decision-making power is divided in the network. Decentralized systems are an abstraction applied to a chain of computers that doesn't depend on a unique machine, as it can be applied to any computer network in which each node doesn't have most of the authority required by the system to keep it alive. Those kinds of systems work with distributed control, and each component (computer), using its local resources, is equally responsible for ensuring that the global system works properly. [60]

A blockchain is a ledger that encrypts the information about the transactions recorded in it. Through a validation procedure, the blockchain's network comes to a consensus regarding transactions and seals the blocks where the data is kept. This validation procedure requires the majority of the network to agree [27] unless it is a centralized system that is not relevant to the work being developed. The following image illustrated how transactions are stored in a blockchain:

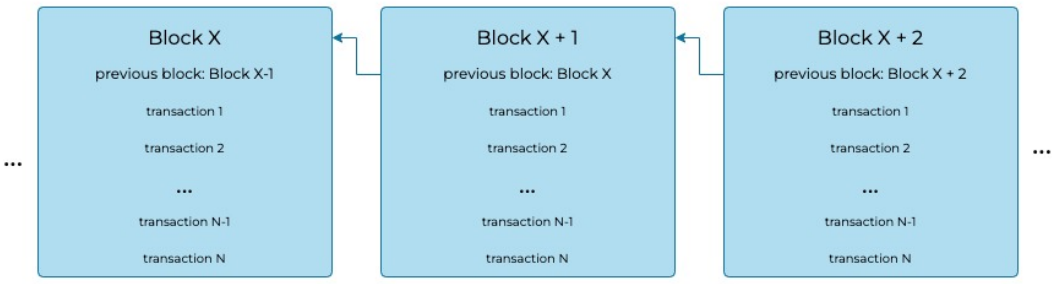


Figure 2.1: Blockchain blocks diagram

It is to be noted that each node of a decentralized blockchain has a full copy of these transactional chains.

Also, the network is shared by the members; the data is added only with more than 50% of group consensus, and although everyone has the data, no one can control it; the number of computers in the network defines the level of security, since the larger the number of computers, the more reliable the network. In this context, the concept of decentralized authority is also involved, in which each computer has power over the final decision but no one can overrule the majority because, in most decentralized systems, more than 50% of the computers need to approve an operation. Among the numerous benefits of using those systems are the trustless environment, data agreement, lack of weakness, and resource distribution. [60]

- **Trustless environment** - None of the participants in a decentralized network need to know or trust one another. A distributed ledger that contains identical data is shared by every member of the network. The majority of the network's members will reject any member whose data has been changed or corrupted in any way.
- **Data reconciliation** - Data exchange between businesses and their partners is commonly stored in silos that transform and store this data, which is then only brought to light when it has to be transferred further down the pipeline. Every time the data is transferred, there is a chance that some of the data will be lost or that some inaccurate data will enter the workstream. Every entity gets access to a real-time, shared view of the data thanks to a decentralized data store.
- **Reduces points of weakness** - When a system relies too much on a few key members, decentralization might lessen its weak spots. These vulnerabilities could result in systemic failures, such as the inability to deliver on promises or inefficient service as a result of resource exhaustion, recurrent outages, bottlenecks, a lack of enough incentives for effective service, or corruption.
- **Resource distribution** - Decentralization can also aid in resource allocation optimization, improving the performance and consistency of promised services while lowering the risk of catastrophic failure.

2.2 What's the blockchain role?

The various technological and industrial fields can leverage blockchain technology. The top IT firms are embracing Blockchain technology to enhance the reliability and efficiency of their systems [11]. Blockchain systems can be used in systems like economic services, electronic voting, authorship, diamonds, medicine, or even in supply chains because of its advantages [31]:

- **Proof of identity** - By disclosing pertinent elements of their identities to the service providers, each user in this type of blockchain proves their identity and membership. The ability to refuse consent for access gives the individual complete control over their own data.
- **Secure and transparent** - In any blockchain, there is a mix of secure and transparent transactions. This is because every transaction is available to the public, where it shows the sender and recipient addresses, the value sent, and fees applied, and there is no way to steal an account without the victim signing a malicious transaction or having the victim's private keys.

- **Authorship and authenticity** - The authenticity of a message can be granted by having the author sign it with the proper signing keys (will differ between blockchains). In the same way, authorship can be acknowledged and royalties can be set for each sale.
- **Tracking** - Companies like Walmart Canada are already using blockchain to track the trucks that transport their inventory. This way, the blockchain enables the synchronization of logistic data, shipment tracking, and the automation of payments [29].

In the same way that blockchain has its advantages also has some disadvantages [73]:

- **High implementation costs** - Unfortunately, blockchains also entail substantial implementation costs for already running businesses, which prevents their widespread adoption and implementation. It can be an advantage for small businesses because they can use the already implemented structure of blockchains.
- **Inefficiency** - Since only one machine/user will profit from the mining process (either by work power or staking), it is inefficient to have multiple computers validating the same operations. This process, and the fact that numerous people are carrying out the same action, also suggests a significant energy loss and uses environmentally unfriendly technologies, although it prevents cybersecurity attacks.
- **Private keys** - Excessive security can also be a weakness because once the private keys are lost, it is nearly impossible to retrieve them. When the price of Bitcoin skyrocketed and individuals scrambled to find the missing wallets, this became an issue [23]. There is still another problem regarding the private keys. There are two ways of "having" a crypto wallet: by having the private keys on a pen drive or on a hard drive, and by having a seed phrase that generates the private keys. Those private keys are just hashes (with different lengths depending on the blockchain structure) that can be stolen by anyone who has access to them.
- **Storage** - As the number of users increases, so will the number of activities that must be integrated into the blocks to be stored. As a result, the amount of space needed within the computers used by the miners will eventually exceed the hard disk storage capacity.
- **Unemployment** - This is not a nowadays problem because some of the following processes are already computerized, but since there will no longer be a need for intermediaries when Blockchain technology is embraced and put into use, all of these intermediation sectors for the validation of payments and procedures will inevitably be reduced to the point of disappearing, along with the jobs necessary for them.

As explained before, each machine that runs a blockchain node needs to have a copy of the entire blockchain. This is because the network needs to achieve a state of a decision that all participants agree with. The objective of having a consensus mechanism is to reject a change in the network that isn't coherent with more than half of the chain stored in the computer's nodes. To achieve this consensus several criteria can be taken into account:

- Proof of Work (PoW) - miners compete with one another to approve the following transaction block and get rewards. Although it requires a lot of energy, this consensus mechanism fosters a high level of confidence [59].
- Proof of Stake (PoS) - is a consensus process in which new blocks are validated by the people who hold most of the network's money. Transactions are made possible

faster and more affordably. It encourages continued involvement by rewarding those who have the greatest stake in the network [68].

- Delegated Proof of Stake (DPoS) - users who stake their currencies can decide how many delegates should be allowed to create new blocks [26].
- Proof of Importance (PoI) - gives people a score for importance, which eventually encourages them to become block harvesters [34].
- Proof of Capacity (PoC) - relies on the computer's hard disk storage space for its decentralized block generation and verification system [18].
- Proof of Elapsed Time (PoET) - assigns the block verification to a miner at random using a random timer that runs separately at each node [49].
- Proof of Activity (PoAc) - A mix of proof-of-stake and proof-of-work. Aims to maximize the benefits of both architectures [59].
- Proof of Authority (PoAu) - The majority of its users are private businesses or organizations, which rely on blocks made by verified sources with unique access rights to the network. Instead of using the general consensus as with other procedures, assurances are based on reputation and authority [10].
- Proof of Burn (PoB) - miners occasionally burn coins to permanently delete or remove that particular coin from circulation, which drives consensus. This avoids inflation while validating new transactions [54].
- Byzantine Fault Tolerance (BFT) - the system is designed to remain functional even if one of the nodes fails in maintaining constant communication between nodes [34].

This is not an important subject in the project, because it doesn't include the creation of a blockchain from scratch. So, this won't be deeply addressed.

2.3 Unspent transaction output VS Account-Based blockchains

A blockchain network uses two different record-keeping techniques to track the origins of its currencies and how cryptocurrency balances are calculated: unspent transaction output (UTXO) and account-based.

UTXO chains include Cardano, Bitcoin and several of its derivatives, including Bitcoin Cash, Zcash, Litecoin, Dogecoin, Dash, and others. While account-based networks are, for example, Ethereum, EOS, Tron, and Ethereum Classic [3]. To this information, it is also true that Binance Smart Chain (BSC) is also an account-based blockchain since BSC is a hard fork of the Go Ethereum[1] [2].

2.3.1 Unspent transaction output

There are no accounts or wallets in a UTXO-based ledger, at least not at the protocol level. The coins are instead kept as a list of unutilized transaction outputs, and transactions are made up of a series of inputs and outputs (UTXOs).

[1]The Go version of Ethereum, known as Geth (go-ethereum), serves as a portal to the decentralized web. Geth was one of the first Ethereum implementations of an execution client, which means it manages transactions, smart contract deployment and execution, and has an Ethereum Virtual Machine embedded in it. A machine becomes an Ethereum node when Geth is run in conjunction with a consensus client [30].

On a UTXO chain, transactions are created by substituting new UTXOs for existing UTXOs. A UTXO value cannot be divided but can be added to the value of other UTXOs to make up a transaction's required amount. [2]

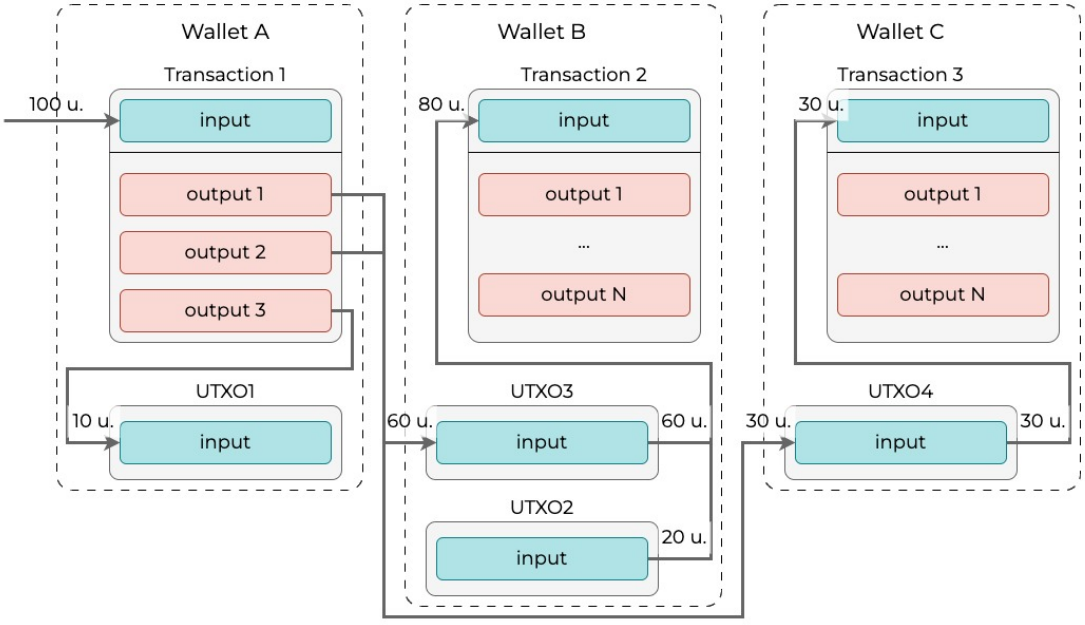


Figure 2.2: Blockchain UTXOs Diagram

In the example above it is illustrated how UTXOs work. To simplify, the unity is the dollar and those are simple money transactions. Let's suppose that Wallet A has 100$ to spend in a transaction. The following occurs:

- Wallet A wants to send 60$ to Wallet B and 30$ to Wallet C. That amount of money sent to each wallet is stored in an unspent transaction output (UTXO) that belongs only to the recipient. This UTXO can be used as input to another transaction, but only the owner of UTXO can use it to fund a transaction.
- Since Wallet A has 100$ to spend but only sends 90$, the remaining 10$ is stored as a UTXO in Wallet A. Further, Wallet A can use this UTXO in combination with another (or not) to send a transaction.
- Wallet B already had a UTXO with 20$ (UTXO2 from Wallet B) and combine its value with UTXO3 to send 80$ to someone (that is not illustrated).
- Wallet C received 30$ which is used to create a transaction in which the whole amount of dollars is used.

This example is important to show three different cases of transactions with unspent transaction outputs. In the first case, there is a remaining, similar to the shopping change, that returns to the sender. In the second case, Wallet B needs to combine two UTXOs in one transaction to send the required amount, like when a person uses two different notes to buy something. The third case, the simplest, uses the whole amount present in the UTXO to create a transaction.

2.3.2 Account-based

Account-based blockchains, like Ethereum, EOS, Tron, and Ethereum Classic, are frequently geared toward smart contracts. A balance management system that functions similarly to a conventional bank account is the account-based model.

Value in an account-based balance can be partially spent, unlike UTXOs. If a user has 75 ETH, for instance, this user can send someone 5 ETH right away from his/her account. The user remains with 70 ETH and the other person has 5 ETH more as a consequence of this transaction. It's not necessary to send the whole 75 ETH and get 70 ETH back as a change, as it would work on a UTXO chain. Instead, on the account-based chain, a transaction request causes nodes to make a withdrawal from the sender's account balance and a deposit to the receiver's account balance.

The account-based technique is more effective because it just has to be verified that the sending account has sufficient funds to cover each transaction[2] [2].

2.4 Real capabilities of NFTs

Non-Fungible Token (NFT) is a type of token, such as cryptocurrencies, that can be traded in a blockchain like Cardano, Ethereum, Solana, Binance Chain, and much more. The difference between NFTs and the common native tokens (like ADA in Cardano, ETH in Ethereum, SOL in Solana, and BNB in Binance Chain) is that each chain's native token can have N instances of it. For example, one person can have 10 ADA tokens (from Cardano). However, each NFT is unique, which means that each person can only have 1 NFT "A" and nobody else can have that same NFT "A". Nowadays, because of the previously explained uniqueness inherent to any NFT, it is commonly used to store art, which confers two characteristics to NFTs: authorship and storage provision. The disproof of this explanation is the duplication of NFTs, that is, two NFTs with the same image/sound/file embedded can be sold separately as different assets. In part, this is true; however, it is impossible for two NFTs to have the same asset ID. Also, there is a collection id (called 'policy id' in Cardano) in every NFT that identifies which collection the NFT belongs to. With this said, there are two cloning possibilities:

- **Two NFTs are originals from the same artist and have the same embedded image** - normally, it is not in an artist's interest to create two similar NFTs because it reduces the rarity of the NFTs. The NFTs have fields, like in an object database, and the rarer the presence of a field in an NFT, the more valuable it becomes. Even on this possibility, although having the same policy id (by being part of the same collection), the two NFTs will have different asset IDs.
- **Two NFTs have the same image associated with them, but they were created by different people** - this is a case where the second person to create the NFT can fool other people. However, there is a security measure that every customer must have before buying an NFT, which is to verify the policy ID. The policy ID is a string

[2] Although it is theoretically more efficient, because there is no need to verify older UTXOs, in practice, a common user will have made many transactions, which originate, in a UTXO blockchain, many UTXOs. The advantage of having multiple UTXOs, instead of one unique balance is that, if the user has UTXO1, UTXO2, UTXO3, and UTXO4 and only uses UTXO1 and UTXO2 to make a transaction, the user doesn't need to wait for the validators to validate the transaction and can make another transaction at the same time, using UTXO3 and UTXO4. In an account-based blockchain, the transaction needs to be validated before making a new one.

generated by a signing key (similar to a private key) so, unless the swindler has direct access to it, the two NFTs will have different policy IDs. All the NFT marketplaces have means of verifying a collection, so it is easier for clients to know if a collection is trustworthy and if they are buying an NFT from the correct collection or not. As this depends on the user, this topic won't be deeply addressed.

As explained before, the NFTs have authority, uniqueness, tracking, and storage properties that give them the ability to store the content for a show ticket, order package tracking, database content, and more, redistributing power and money all over the owners of the NFTs.

Chapter 3

State of Art

In the following sections, there will be a characterization of the work developed to date regarding the use of blockchain in the context of the arts, starting with the knowledge about intellectual property, a blockchain created specifically to implement a marketplace for artworks, and the security issues inherent to the use of blockchain. Then it will be presented the justification for choosing a specific blockchain for the development of the project. This blockchain will be chosen taking into account its usage, reliability, support, development, and maintenance.

3.1 Part I

In this first part of the State of Art will be presented the most common use cases of blockchains and NFTs since the beginning of NFTs' generation, in 2014, called Quantum, created over the Namecoin blockchain (a fork from Bitcoin) [72].

3.1.1 Intellectual Property

Visibility is seen as the capability of linking useful information to assets and letting it flow to those who need it in a network of individuals. This concept emerged from a very common problem between the creator and the consumer of an online file: the rights associated with a work may not be fully known to the consumer, and once the work has been posted online, the creator may not be fully aware of who, where, when, and how their work is being used. The blockchain creates a distributed ledger of transactions that can be read and verified by anyone and is replicated across many computers, providing (the previously explained) transaction visibility [47].

Nowadays, questions regarding copyrighting rights are still up because, while owners of the rights want to refer to copyright violations as "stealing", in legal terms, they are violations of an artificial monopoly on copying rather than theft [56] [15]. As part of general beliefs in a socio-cultural setting, the manner and processes of transactions to purchase and transfer ownership are easily known in any physical store where the products are plainly the store owner's property: It seems obvious that removing a physical commodity from the store owner is to their disadvantage. The setting in the digital world is different, however, due to the simplicity of digital copying and the fact that the rights-holder does not lose anything after the copy is made. In another opinion, given by Glenn Parry [55], it is said that a change is needed to protect property rights in art while still allowing the use of that artwork for creative purposes.

With this said, it is observable that copyright laws are ineffective in practice because of how simple it is to distribute and copy digital artworks on the internet. If the processes for granting and transferring ownership are not offered, customers will continue to behave indifferently in the digital domain, which is inconsistent with how they behave elsewhere.

In the real world, a piece of art is created by an artist who gives it to a gallery to sell or sells it to a known person who likes the art. The person who buys the art, either directly from the artist or through a gallery, is called a collector. When a purchase is made for the first time, a provenance document is created for that artwork with all of the art's history (biographic details, proof of authenticity, attribution, and with space for all future buyers, including each transfer of ownership and information regarding the work's locations, dates, and means of transference). This is important to keep track of since provenance creates a chain of ownership and determines a work's value; it is crucial for demonstrating the legitimacy of an artwork.

Since the moment that provenance was seen as important (about 200 years ago) [53] that a textual document has been available to the public. This document provides a succinct, semi-structured description of a complicated network of relationships between people, things, places, and events. Although it contains a remarkable amount of information about the object's past, understanding it heavily depends on the reader's level of expertise. To express the recorded knowledge of provenance in a machine-readable way, many implicit statements, events, and entities would need to be extracted from the data. [53]

There is also an international standard, fully established in 2006, after its standardization began in 2000, that is accepted as the ISO 21127:2006 standard: Information and documentation, A reference ontology for the interchange of cultural heritage information.

> "ISO 21127:2006 establishes guidelines for the exchange of information between cultural heritage institutions. In simple terms, this can be defined as the curated knowledge of museums" [64] - *International Organization for Standardization*.

Quote 1: International Organization for Standardization regarding cultural heritage

BigchainDB[1] is an organization that created an open-source project called "ascribe", whose mission was to register digital property in the form of videos, screensavers, software, audio, images, and more. The main purpose of *ascribing* was to give *visibility* to digital work. Ascribe uses the Bitcoin blockchain to store data about artworks and the ownership chain of transactions. In that project, creators could register a work, create editions, transfer editions, or consign a work for sale in the secondary market. This was a step forward for artwork visibility as BigchainDB worked with creators and lawyers in several countries to ensure compliance with terms of service and to allow creators to register a work under the Creative Commons[2] (CC) license.

Since the blockchains, until that date, didn't offer sufficient means to meet the needs, ascribe developed a new protocol called Secure Public Online Ownership Ledger (SPOOL). The method behind it is simple: the creator sends his/her own work with the help of blockchain and keeps it "sealed" until the moment (if necessary) of a dispute over authorship. This way

[1] "The blockchain database.", BigchainDB, https://www.bigchaindb.com/

[2] "When we share, everyone wins", Creative Commons, https://creativecommons.org/

the artist can prove that he/she had the work all this time (with the timestamp associated with the creation).

As of the moment of writing this document, ascribe is no longer active, having ceased in 2013. However, since it used the Bitcoin blockchain to store the data, ownership provenance is still recorded.

3.1.2 Art Marketplace

In 2019, the Blockchain Innovation Center presented a way to track and record trading in the art industry with the help of blockchain [75]. The team identified some challenges in the art market:

- lack of ownership history (provenance) and price transparency, as well as insufficient transaction data control because of data asymmetry;
- it can be challenging to determine the authenticity and value of expensive pieces of art;
- lack of transparency in secondary auction market trade and the main art market's lack of value for artworks;
- lack of respect, awareness, and consideration for a wide number of artists;
- it is challenging for artists to obtain royalties from secondary market sales.

The given solution has four main entities: the artist, who can publish his/her artwork; the art gallery, which can track and locate artworks; the auction houses, which can open new channels for a greater audience; and the collector, whose physical artwork is synchronized with online assets, proving its ownership. This way, blockchain helps this system by providing privacy protection, traceability (and provenance to reduce the feeling of suspicion about stolen or fake art), irreversibility, and transparency. Another important point about digital art is the royalty payout, which was difficult at the moment of the project's creation because it is difficult to track the sales of physical art. ArtChain proposes a 5% royalty [75] payable to the artists on commercial sales.

Since the project refers to the creation of a blockchain, the rewards would be based on ACG (ArtChain Global) - for security - and ACGT (ArtChain Global Token) - for utility. The first ones are related to the essential technical features of digital currencies, including free trading, immunity to double-spending attacks, and traceable transaction histories. The ACTG token would be used to facilitate payment in art trading, with a less volatile value being collateralized with a fiat currency[3], in a 1/1 relationship, like the commonly called "stable coins" (which mimic the value of the fiat currency).

Although their project was canceled, most likely because all the concepts presented were already applied to the most commonly used blockchains, the team was able to prove that a system for artwork like the one they created could provide a complete blockchain-based solution with all the benefits for the art industry as previously explained.

[3]Government-produced currency that is not backed by a tangible good like gold or silver, but rather by the government that created it. Instead of being backed by the value of a commodity, fiat money derives its value from the interplay between supply and demand as well as the stability of the government that issues it. The majority of contemporary paper currencies, including the U.S. dollar, the euro, and other significant world currencies, are fiat currencies. [13]

3.1.3 Non-Fungible Tokens as Event Tickets

In a paper called "Non-Fungible Tokens as Core Component of a Blockchain-based Event Ticketing Application", Ferdinand Regner, André Schweizer, and Nils Urbach explain how they designed and developed a blockchain-based system for event ticketing, using NFTs as the core component [57].

To do so they started by identifying the problems of the ticketing industry, such as the Lack of Trust, the lack of control over secondary market prices, the dependence on intermediaries, the fact that there is no immediate validation, and the lack of transparency.

After identifying the issues, some objectives were defined, as follows:

- Digitization: all data should be stored in a digitalized way
- Secondary market control: the event organizer should be paid for each transaction among clients and the prices should be capped
- Autonomy: intermediaries shouldn't exist between the organizer and the client
- Security: the environment should have the resources accessible, the data authentic, and should prevent illegitimate users from accessing the data
- Security: the environment should have the resources accessible, the data authentic, and should prevent illegitimate users from accessing the data
- Validation: the ownership of a ticket should be easily verifiable
- Transparency: the transaction history should be transparent for every ticket. Includes the current ownership and transactions between clients
- Automation: the system shouldn't have any manual interaction from the organizer
- Cost Efficiency: event organizers should see the costs as economical

The resulting Ethereum-based system eliminates the need for a middleman by making the only two entities communicate through a smart contract that allows the creation of a ticket as well as its purchase and sale. For this communication to work it is needed the both entities to have native tokens from the Ethereum Blockchain (Ether).

The whole process can be described in 3 phases: The Setup phase where the event organizers deploy the smart contract for the event; the Primary market where the clients can buy the tickets, by sending the Ether to the payable function, if there is any remaining supply; and the Secondary market, where the ticket owners can offer their tickets for sale and define the price they want for the ticket, if it doesn't exceed the price price cap set by the event organizer.

The work developed by the researchers does not provide a bridge between the real world and the blockchain, i.e. only experienced blockchain users are allowed to use the system, making it hard for the common user to take advantage of it.

In the end, as the system follows the ERC-721 standard [24], it enables the users to use Ethereum-compatible marketplaces like OpenSea to make transactions between the attendees easier.

3.1.4 Security Issues

As reported in "A Survey of Blockchain Security Issues and Challenges" written by Iuon-Chang Lin and Tzu-Chun Liao [46], there are six main challenges faced by blockchains, with more focus on the Bitcoin blockchain, which is based on a proof-of-work consensus. And at the end, it will be presented with one more, commonly associated with proof-of-stake blockchains.

- The most common attack is the 51% attack, called **The Majority Attack**. Iuon-Chang Lin and Tzu-Chun Liao refer to this kind of attack as occurring when, at least 51% of the mining power is controlled by one entity, and this entity doesn't need to be a person, because a group of people can join a group of miners, called a mining pool, and get 51% of mining power. This kind of power lets the entity control which blocks enter the network, modifies transaction data (and causes a double-spending attack), and orders a miner to stop mining a block. Despite this being explained for blockchains based on proof-of-work consensus, this is also applicable when the proof-of-stake mechanism is used, as it will be explained later.
- Blockchains are a relatively new technology, and they imply many technological updates. Therefore, all the nodes connected to the network must run the same version; however, it isn't always possible because the network can't stop so all the nodes can upgrade. This process is incremental, and some nodes will have version A (let's suppose it is the newer version) and other nodes will have version B (the older version of the node software). The issue can occur when an A node only trusts A nodes; when a B node only trusts B nodes; when a B node accepts an A node transaction; or when an A node accepts an A node transaction.
- As new blocks are minted in blockchains, more data is required for nodes to store. This brings an issue because Bitcoin has a new block of transactions every 10 minutes (on average), and every node in the chain needs to clone that block. This is computationally expensive. So, there is a technology used to verify transactions that is lighter compared to "Full Payment Verification", which is called Simplified Payment Verification (SPV) and it was first explained in the Bitcoin whitepaper. SPV refers to a process where the receiver of a transaction can demonstrate that the sender owns the transaction's source funds without having to download the full blockchain history. The SPV just downloads the block headers and requests the blockchain's Merkle Tree[4] to reduce the amount of storage needed [45].
- Bitcoin has an average validation time of 1 to 1.5 hours. It is not perfect, and for this, there is the research presented by Thaddeus Dryja and Joseph Poon [37] which would become the Lightning Network, a project that allows peer-to-peer transactions without telling the main blockchain. For that, it is needed to create a channel where the payer needs to lock a certain amount of bitcoin into the network. The recipients can invoice quantities of Bitcoin as they see fit once it is locked in [17].
- The laws and regulations are not equal all over the world. Until 2022 the Portugal government was not taxing cryptocurrencies, but since 2023, all revenues from cryptocurrencies will be taxed at 28% over the gains, but only if they are stored for less than one year, otherwise, there is a tax exemption [65]. Short-term capital gains are

[4] Merkle Tree is a data structure used for data synchronization and verification. It is mostly used in distributed systems where the same information exists in multiple machines because it helps to find inconsistencies.

subject to a tax of up to 37% in the USA, whereas long-term capital gains are subject to a tax of 0% to 20% [19].

- The costs associated with the transition to a system fully based on blockchain are also a problem, especially when a company already has a complex system to deal with. The easier it is to make the conversion, the lower the costs.
- The Majority attack on the proof-of-stake consensus mechanism. The difference between proof-of-stake and proof-of-work relies on the computational power required to perform validation. In blockchains with proof-of-work consensus, validators just need to run software on a computer to start validating blocks of transactions, and the quicker it gets to validate a transaction, the higher the probability of getting rewarded. In proof-of-stake, there is no competition to validate a block of transactions, since the mechanism settles on the number of tokens staked. Stake tokens mean letting a validator prove it is trusted. Anyone can stake tokens to get rewards because the validator's pool redistributes its rewards to the people who stake their tokens in that pool. This means that the higher the number of people staking tokens in a pool, and the higher the number of tokens being staked in that pool, the higher the confidence in that pool. With this explained, and even though this is hard to happen because of the pool's limits if a pool (or a group of pools controlled by the same entity) has 51% of the staked tokens, then fraudulent transactions can be validated.

3.2 Part II

3.2.1 Most used blockchains

According to Coinmarketcap [16], with data dated 11/01/2023, the most traded tokens and consequently the most used blockchains are, by descending order, Bitcoin, Ethereum, BNB, XRP, and Cardano, where Cardano has the least amount of transactions. This ranking is measured by comparing the market capitalization of each of the blockchain's native tokens. Although are many other tokens listed between these five, those other tokens are built on top of the others. For example, Tether, and USD Coin are tokens built on top of Ethereum and some other blockchains. Tether, USD Coin and Binance USD are examples of the previously explained stablecoins.

Bitcoin

Bitcoin was the first cryptocurrency, launched in 2009 to overtake the normal electronic banking system [50]. Since Bitcoin doesn't have a centralized system behind it, no one can control it entirely. Bitcoin as a token (BTC) wasn't directly an exchangeable currency; it started as a reward for anyone who was validating the transactions in the network. With the quick expansion of this token, it quickly became a financial asset that barely everyone could store in a virtual wallet.

Ethereum

Because both Bitcoin and Ethereum use blockchain technology, there are parallels between them. However, when compared to each other's designs and intended uses, they are radically dissimilar. Bitcoin's main use case is as a payment currency, while the Ethereum blockchain is built to allow for a lot more uses that could be beneficial to the corporate world [50]. Ethereum has built a global computer network that connects users to a marketplace of

decentralized applications (dApps) offering unprecedented efficiency, security, and user control, building on its predecessor's vision of a decentralized payments system. While Bitcoin's innovative decentralized network and cryptocurrency were a ground-breaking achievement, Ethereum has expanded on it. Ethereum is utilized for a wide range of cutting-edge applications in finance, web browsing, gaming, advertising, identity management, and supply chain management thanks to its revolutionary combination of features, including smart contracts. The ether (ETH) coin that powers the Ethereum blockchain enables developers to build new kinds of ETH-based tokens that leverage smart contracts to power dApps. The ERC-20 token standard is the foundation for the majority of ETH-based cryptocurrencies. A fundamental feature in Ethereum and blockchain are smart contracts, which are self-executing agreements that enable, confirm, and enforce transactions. In addition, Ethereum is a permissionless blockchain, which enables the construction and development of apps without the need for oversight from a central authority. On Ethereum, there have been developed thousands of dApps, which generated millions of users, and many developers could earn a large amount of money [67].

Binance Smart Chain / BNB Chain

This is a blockchain completely independent from Binance Exchange (which started out as a centralized blockchain). The characteristics and operations of the Binance Smart Chain (actual BNB Chain) are comparable to those of Ethereum because it was designed to be interoperable with Ethereum. The two blockchains do differ in a few ways, though [1][5]:

- For Ethereum, staking a validator requires a minimum of 32 ETH, whereas Binance Smart Chain demand a minimum staking of 10,000 BNB.
- While BSC is branded with the firm name Exchange Binance, Ethereum is not branded under the name of its corporation.
- The Binance smart chain has more centralization when it comes to validator numbers, whereas Ethereum has less.

Although the proof-of-authority (PoA) protocol is frequently criticized for not being as decentralized as the proof-of-work (PoW) protocol, Binance Smart Chain uses PoA as its consensus mechanism, and its validators, or the nodes that produce blocks alternately, hold all the power and are therefore vulnerable to fraud and other security threats. To solve this, it's used a combination of Proof-of-Authority and Proof-of-Stake [5]:

- Blocks are generated by a small number of validators.
- Similar to Ethereum's Clique consensus design, validators construct blocks in a PoA manner by alternating between them.
- A staking-based governance is used to elect the validator set both within and outside.

Ripple's XRP

With its own cryptocurrency, XRP, Ripple is a blockchain-based digital payment network and protocol. Similar to the SWIFT system for international money and security transfers, which is used by banks and financial middlemen trading across currencies, Ripple's primary

[5] The author of this article defends that one of the drawbacks of the Binance Smart Chain is its centralization. However, as stated in Binance's blog, the Binance Smart Chain is decentralized and created for the developers to make dApps (decentralized applications) [4]

operation is a payment settlement asset exchange and remittance system [28]. Some of the potential use cases for XRP include [9]:

- International money transfers: XRP can facilitate cross-border transactions by allowing for the quick and efficient transfer of funds between different currencies.
- Banking and financial services: XRP can be used by financial institutions to settle transactions and reduce the need for intermediaries.
- Online marketplaces: XRP can be used as a means of payment on online marketplaces, enabling fast and secure transactions.
- Micropayments: XRP can be used to facilitate small, frequent payments, such as those made for online content or IoT (Internet of Things) services.

Cardano

Cardano is a blockchain platform that is designed for a wide range of use cases. Some of the potential use cases for Cardano include:

- Smart Contracts: Cardano supports the development of smart contracts, which are self-executing contracts with the terms of the agreement between buyer and seller being directly written into lines of code.
- Decentralized Applications (dApps): Cardano can support a wide range of dApps, including financial services, digital identity, and supply chain management.
- Tokenization: Cardano can be used to create and manage digital assets, enabling the tokenization of traditional assets such as real estate and art, are opening up new financial opportunities.
- Microfinance: Cardano's blockchain can be used to create a decentralized microfinance platform, which could help to increase financial inclusion in under banked-regions.

3.2.2 Cardano-powered ticket marketplace

When creating a marketplace for concert and theater tickets, expect a blockchain capable of tracking the buyers and creating a secure way of granting ownership of a ticket. As explained before, NFTs can take most of those responsibilities adding the royalties redistribution to the list. The following items are the core unique features of Cardano that make it the final decision for the development of a ticket distribution system [6]:

- The minting of NFTs doesn't require smart contracts - In Cardano, the creation of crypto tokens like NFTs is possible without the use of a smart contract because it is a fundamental feature of the Cardano blockchain. This reduces the possibility of NFT minting mistakes and bugs throughout a token's lifetime. The extended-UTxO (eUTXO) concept of the Cardano blockchain, as explained before, enables the native transfer of many assets alongside ADA, which gives rise to the expression "multi-asset network".
- Batch transactions - Cardano transactions allow for the bundling of several token transfers to numerous wallet addresses in a single transaction. As a result, the owner of an NFT collection can swiftly transfer their NFTs to a large number of recipients while only incurring a single small transaction charge. This can be a big advantage when minting hundreds or thousands of NFTs at once for many customers.

- Metadata is part of the transaction - The NFT asset is permanently connected to this data that is stored in the Cardano blockchain. Since everyone has access to the same data, this makes Cardano NFTs easier to validate and secure. There is no need to rely on third-party storage options that are prone to destroying, altering, or even incorrectly labeling an NFT's metadata[6].
- Low and deterministic fees for fast NFT transactions - The number of NFTs and the collection's metadata requirements determine the transaction to mint an NFT on Cardano. All subsequent transfers of the tokens simply cost the network's basic transfer fee after the minting fee has been paid. The fundamental transaction fee on the Cardano blockchain is flat and applies to all simple transactions, in contrast to smart contract-based NFTs, where fees can be extraordinarily high owing to blockchain network congestion.
- Expanding the NFT market - Despite the current state of the market[7], the Cardano NFT market is expanding as more NFT collections are created on the platform and new infrastructure initiatives are established. With its distinctive advantages as an environmentally sustainable and decentralized smart contract platform that provides users with quick transaction times and affordable fees, Cardano is a hub for NFT collections, with many artists, developers, and collectors looking to migrate to the Cardano blockchain for NFTs. In contrast to previous networks, this has produced an environment that is conducive to innovation and growth.

Although many of these characteristics are common to other smart contract blockchains, there are some points in which Cardano beats the strategies adopted by other blockchains.

Cardano vs Ethereum & Binance Smart Chain Cardano uses a more advanced consensus mechanism called Ouroboros, which is designed to be more secure, fair, and energy-efficient than the Proof-of-Stake consensus mechanism used by Ethereum and BSC (Binance Smart Chain).

Ouroboros, the Proof-of-Stake (PoS) consensus algorithm used in the Cardano blockchain, has several distinguishing security features when compared to other PoS protocols.

It is provably secure, insofar as is the first PoS protocol that has been mathematically proven to be secure; Ouroboros divides time into epochs and slots, with each slot representing a fixed time interval, which in terms of security means that allows an efficient block production and consensus; Ouroboros uses a VRF (Verifiable Random Functions) based leader selection process to determine the slot leaders for each epoch which ensures randomness and unpredictability in the selection of slot leaders, preventing manipulation and enabling secure block creation. [41]

Also, the team has a strong focus on research and development, with a team of scientists and academics working on the project. This allows for a more robust and secure blockchain protocol.

The founder of Cardano, Charles Hoskinson, regularly responds to users' questions through lives treams and explains openly the algorithms chosen for Cardano. This blockchain has

[6] Cardano has a size limit of 16KB of data per transaction [7] while Ethereum has a limit based on the gas price paid [52]. This reduces the capability of Cardano to send many transactions at once. However, the costs of sending transactions through the Ethereum blockchain are much higher.

[7] recession market by January 2023.

a layered architecture that separates the Cardano Settlement Layer (CSL) and Cardano Computing Layer (CCL), which allows for more flexibility and scalability [12].

Cardano's smart contract platform, Plutus, is built on a functional programming language, Haskell, which is considered to be more secure than Solidity, the language used by Ethereum and Binance, because Haskell is a functional language, which turns out to be much easier to fully check for errors [71].

As explained before, Binance Smart Chain is a fork from the core blockchain of Ethereum, so both blockchains will have many similarities and share the same disadvantages. Binance Smart Chain was investigated by the community in 2021 because of suspicions that it was a semi-centralized blockchain, as stated by Wilson Withiam [48].

Cardano vs XRP Cardano is built on a proof-of-stake (PoS) consensus algorithm, which is considered to be less energy-efficient but more secure than the Ripple Protocol Consensus Algorithm (RPCA) used by Ripple. Also, anyone can sign up to be an XRP validator, but to win the trust of the network and be used by other participants, validators must be included in Ripple's unique node list (UNL), which designates them as reliable Ripple validators, the reason why PoS is more secure. These centralized validators are essential for avoiding double spending and transaction censorship. There are only 35 active XRP validators, of which Ripple operates six. Ripple transactions are cheaper, faster, and use less energy than Bitcoin or Ethereum (Classic) transactions because they do not require the energy-intensive mining associated with Proof of Work. This relies on the voting power of the validators. However, some contend that the centralized infrastructure of XRP renders the network less secure, censorship-resistant, and permissionless than an open-source blockchain network like Cardano, which ultimately accounts for this openness and, simultaneously, security [66].

Cardano vs Solana The biggest difference between both blockchains is the amount of information that NFTs in Solana can store which is around 1.2KB [22] compared to the 16KB from Cardano [7] and the fact that despite the fact Solana community considers it a reliable blockchain, it has many downtimes during the year of 2022, as reported by Solana [21], which is not good for running businesses. The only advantage of Solana is that its transaction fees are much lower than Cardano's. However, because of the down times, Cardano is still preferable.

There are many other blockchains but those stated are the most used ones and with the purpose that a marketplace app needs. Further, these blockchains will be compared with each other based on their reliability, decentralization, simplicity, sustainability, and operational cost[8].

[8] In chapter 4, the criteria are presented in more detail

Chapter 4

Value Analysis

The value of a product is a term commonly confused even between teams in the same company because each team tries to give value to the product/service in its way, i.e, the selling team will see the 'value' as the price of the product, the design team will see it as the aesthetics, the manufacturers as the efficiency and the administrator as the excellence. Regarding the investigation, there are 4 types of value considered to study: 'cost value' which is the monetary value spent creating the product, 'exchange value' seen as the product's qualities that makes other people may want to trade the product for something else, 'use value' that is the price paid by the buyer or the manufacturer to ensure that the product works properly, and 'esteem value' which is the attractiveness that increases the sales power by creating the desire of possession [62].

4.1 New Concept Development

Innovation can be a business term used in many cases to describe a new kind of value created for the customer. The companies exist to create value for the customer and this can be made by investing in Research and Development (R&D) and New Product Development (NPD) activities to aid in this direction. However, this is a vague description of what needs to be done to generate new ideas because it needs to take into account the organization's structure. [44]

Traditional Product Development techniques are incomplete approaches from the perspective of innovation. This is because of being disciplined with a project plan and a high degree of certainty, budgeted, predictable, oriented to milestone achievement, and with a multi-function product/process development team. NPD is not a good strategy to create the previously mentioned innovation because it is just a part of the process of generating new ideas. [42]

> "Innovation can be seen as "a new or changed entity realizing or redistributing value" [61]
> - International Organization for Standardization.

Quote 2: International Organization for Standardization regarding innovation

This is why the Fuzzy Front End (FFE) is important to exist. It is the starting point of the innovation process and was perfected to be generic to any company and to specify what terms correspond to each definition. Because it can be the foundation of innovation initiatives, the quality of the starter work impacts the level of innovation and the success of innovation in the later phases. It guarantees, for instance, that no opportunities are lost. [25]

In addition, Fuzzy Front End differs by being experimental and chaotic. There are many "Eureka" moments and there is no other goal to achieve than finding new ideas. Since it is unpredictable and uncertain, the funding is hard to get and many projects get bootlegged or need funding to proceed. During this phase, the team can join to discuss ideas or individually the collaborators conduct research to minimize the risk and optimize the project's potential. The way the team can measure progress is by analyzing the generated concepts and the stronger the concepts, the greater the phase has progressed. [42]

Taking the previous premises into account, the entire innovation process may be divided into three phases, starting with the fuzzy front end, going through new product development, and ending with commercialization, as shown in the following image:

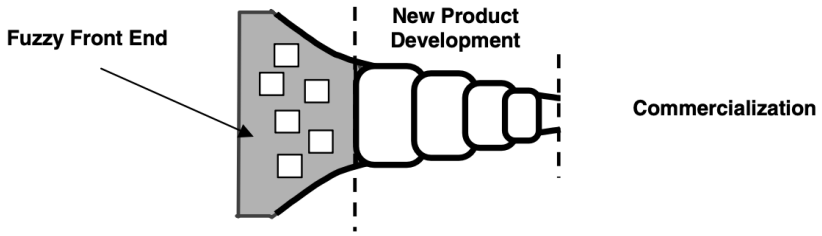


Figure 4.1: Innovation Phases

The main outcome of the investigation done by Peter A. Koen and his co-authors is the new model called **New Concept Development** (NCD). This model is composed of three main components: the "**engine**" which is the leadership, culture, and business strategy; the **controllable activities** which can be defined as opportunity identification, opportunity analysis, idea generation/enrichment, idea selection, and concept definition; and the **influencing factors** that consist of organizational capabilities, the outside world and the technologies that can be involved. [42]

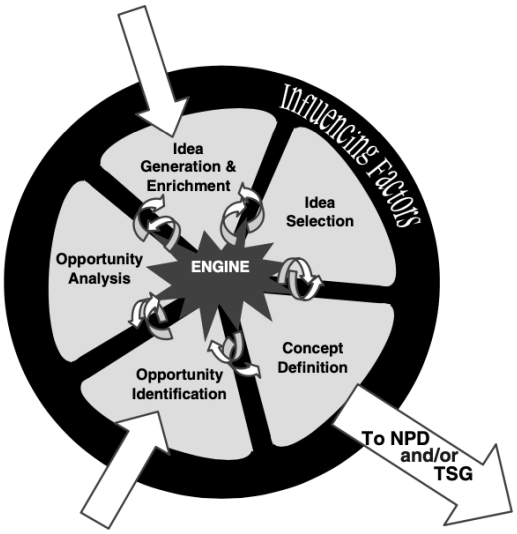


Figure 4.2: New Concept Development Diagram

The NCD model gives definitions and a standard vocabulary for the essential parts of the fuzzy front end. In the center, the five activities of the NCD model are propelled by the engine, which stands for support from the business leadership. The influencing factors are

positioned beneath the engine and the five components of the NCD model. The NCD model's circular diagram shows that new ideas and concepts move from activity to activity, i.e. from each one of the five elements without a specified order. The starting points for projects are represented by the arrows pointing into the diagram, and they are idea generation and enrichment or opportunity identification. Concepts exit the model and enter the new product development (NPD) or technological stage gate (TSG) process as shown by the exiting arrow. [42]

4.2 Opportunity identification & analysis

An opportunity can be seen as a business or technological breach that one recognizes between the present situation and the desired future to gain a competitive edge, address a threat, find a solution to a problem, or lessen a challenge. [42]

In the present work, many gaps are being explored creating value for the users that experience some difficulties in the process of buying/selling show tickets. The first problem is addressed to the artists who need to pay a big amount of operational costs when selling tickets, as well as to rent the space where they are going to perform. In this case, the selling costs could be annulled and something else could be added to benefit the artist and help with the renting costs.

On the other hand, the normal user, the one who buys the ticket, with the current models cannot perform the selling of the ticket at a higher price than it was initially purchased. Nowadays users already use the tickets from the most famous shows as an investment, since there is a lot of demand. The problem with this is that selling at a higher price cannot be done. This is an issue that can be solved with the usage of blockchain and nfts through a dedicated marketplace.

To assess the previous subject, internal and external factors SWOT analysis needs to be done. This is a technique that helps the management to build strong foundations, minimize weak points, pick good opportunities, and mitigate threats since this allows the management to predict what can go wrong [8].

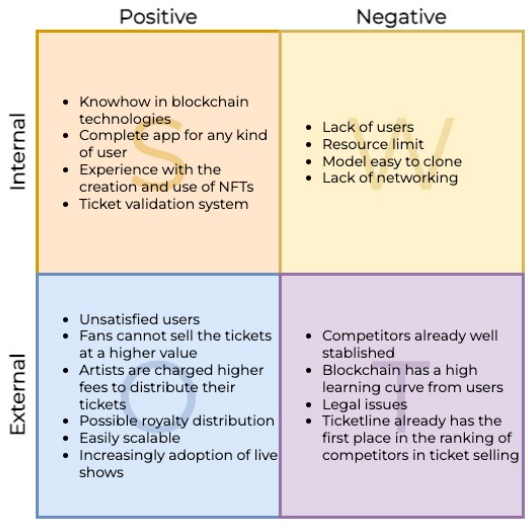


Figure 4.3: SWOT analysis: Strengths, Weaknesses, Opportunity, Threats

The previous image allows the understanding of the state of the project and where there are opportunities for growth or fixing any faults in its operation. The main topics of SWOT analysis can be described as follows:

- **Strengths**: This point tries to aim at what the business does well and what is its competitive advantage. In this project, the know-how in blockchain technologies and in the creation of Non-Fungible Tokens, and the fact of being a complete system with its own validation method based on blockchain signed messages, constitute the strengths.
- **Weaknesses**: This aspect enumerates the weaknesses of the system and processes. In this case, there is a lack of users, a limitation on available resources and contact networking, and the model is easy to clone.
- **Opportunity**: Taking into account the current market trends, what can be used to the project's advantage is the subject of this point. Web3 Ticket will be dealing with unsatisfied customers, that can't either sell the tickets at a higher price or save money on ticket redistribution (because of the current ticket stores[1]). There is also a space for royalty distribution and it is easily scalable. In addition, there is an increase in the adoption of live show audiences[2].
- **Threats**: This matter refers to the external factors that can put the business at risk. Regarding this, there are competitors, like Ticketline in Portugal and Ticketmaster abroad, that already have a big amount of market share. Also, blockchain is a concept with a high learning curve and it can bring legal issues because of the invoicing.

4.2.1 Competitors & Market Segment

Before explaining the project's market segment the main features need to be addressed:

- The artist can create tickets for the show, specify the date, the price, and the place where it will be.
- The fan can buy the ticket directly from the artist (if there is any available) or at the public marketplace where other users can sell their tickets for a show at any price they want [3]
- The artist can receive royalties for each marketplace sale.
- The place's organization can validate the tickets by asking the user (fan) to sign a nonce with his private key.

Given the past features, some competitors can be picked:

- JPG.Store - is a marketplace for Cardano blockchain nfts and has 99.8% of the market share[4]. This marketplace is not centered on tickets but can easily be used to sell them. Also, anyone can create collections of NFTs using JPG.Store, since it has its own NFT

[1] This information can be obtained from the Terms & Conditions of some ticket stores.

[2] From 2015 to 2019 there was a big increase in the number of shows and audiences. In the year of the pandemic, these numbers dropped a little, showing a recovery since then. Data were taken from PORDATA: https://www.pordata.pt/db/portugal/ambiente+de+consulta/tabela

[3] From the previous study, this is not completely true because there is a minimum amount of tokens that the transaction need to carry to make the transfer between two wallets. This value doesn't correspond to the transaction fee that is also added.

[4] This data was taken on 09/02/2023 at the OpenCNFT analytics platform.

creation platform. JPG.Store is the biggest competitor in the Cardano blockchain macro-system since it allows the users to buy and sell NFTs, returning royalties to the collection creator, like proposed in this project.

- Ticketline - isn't the only ticket store in the market, but it is at the top 1 in the ranking of ticket-selling websites[5]. This is an online store with a huge visibility among the artistic world, although there are other stores like BlueTicket and Fnac doing the same exact job,

In addition, some users/customers targets can also be picked:

- Artists - want to create tickets for each show. These artists need to be raised taking into account that they are performing artists. Those artists are also the main decoy for this project because they are the ones who will call people (the fans) to use the application.
- Fans - are the most common user, with more than 3.5 million spectators in Portugal in 2021[6]
- Place's organization - needs to be contacted directly, because not anyone can register in the app as a "place" and its organization needs to be trained to use the platform as a way of validating people and to accept live shows to occur on a certain date.

4.3 The idea behind the concept

An idea is the earliest stage of a new good or service. It frequently comprises a broad overview of the proposed solution to the issue raised by the opportunity. [42]

4.3.1 Idea Generation

Since the opportunities as described and are worth taking this path, the next step is to brainstorm the ideas that will turn the opportunity into a solution that takes advantage of the project's strengths and market opportunities. Previous investigations uncovered the reality of selling tickets in the current models and NFTs made by musical and visual artists.

While the NFTs can be commercialized only online and with advantages for both fans and creators, the tickets are currently sold only using fiduciary exchanges and don't give much freedom either to artists or fans. Then two solutions came up to simplify the processes, get track of activities, and benefit the entities:

- Try to update the current system to get track of the tickets, delete the fees taken by ticket stores, give royalties back to artists, and let fans sell the tickets at the price they want;
- Create a solution based on blockchain, in which its capabilities already solve most of the problems.

4.3.2 Ideal Selection

As already explained in the previous chapters, blockchains that support NFTs have a common implementation behind them and solve many of the problems listed before. On one hand,

[5]This data was taken on 09/02/2023 from the SimilarWeb analytics platform.
[6]Data were taken from PORDATA.

blockchain has increased transparency, improved security, lower implementation costs, and better traceability since every transaction is indefinitely stored in blockchain. On other hand, the advantages o NFTs in this context are their uniqueness, the proof of ownership, and the identity verification. All these advantages are inherent to the use of NFTs and blockchain technologies, also they are required for a system like this and don't need an extra implementation of those features.

The current model only benefits the stores that charge a high commission for each sale. With this in mind, the chosen idea is the one that has more advantages for the artist and the fan and can get high profitability if well adopted. However, many blockchains can be used for development and although it has been explained before in the contextualization, this decision lacks a data analysis. The next chapter will explain in detail how Cardano was chosen to be the selected blockchain.

Decision Making

Multiple Criteria Decision Making (MCDM) is a framework that allows the selection of the best choice among several alternatives. For that, the method used must compare the conflicting criteria. Although there are many other methods to choose from, like Simple Additive Weighting (SAW) and Preference Ranking Organization Method for Enrichment of Evaluations (PROMETHEE), it will be used the Analytic Hierarchy Process (AHP) and the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) [76].

AHP is a pair-wise method constructed over a hierarchical structure composed of the Objective, the Criteria, and the Alternatives. It assesses the relative importance between criteria, then the alternatives are compared for each criterion and the overall ranking is determined. This has a particularity in the final result because AHP has a mechanism for checking the consistency of the weights given to the criteria. The second method, TOPSIS, is based on the shortest distance between the alternative and the ideal solution, using the Euclidean distance to determine the relative proximity. This takes into account two variables: the positive ideal solution, is calculated as the maximum of all the returned values for each criterion, and the negative ideal solution is calculated as the minimum of all the values [69].

Since the TOPSIS can't evaluate the criteria based on their relative weights, the first phase of the analysis will be done resorting to AHP, which the outcome will be the criteria weights to be used in the middle of the TOPSIS analysis, without which the assessment would have been negligent.

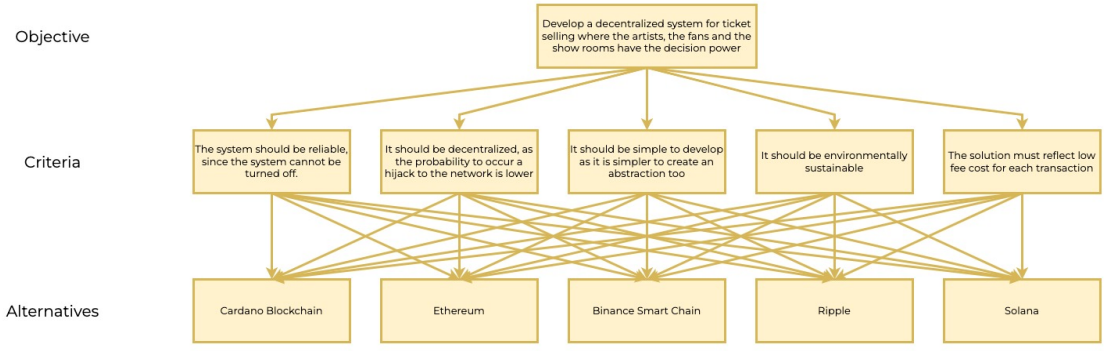


Figure 4.4: The Hierarchy for Evaluating Blockchain Selection: this identifies the decision problem and defines the objectives and alternatives.

As shown in Figure 4.4 AHP lay in a set of criteria and alternatives, with the objective at the top of the tree being the development of a decentralized system for ticket selling where the artists, the fans, and the showrooms have the decision power. The criteria that will be used are the **reliability**, i.e, the probability of the network having flaws; **Decentralization** which consists of the probability of the network being controlled by a single entity[7]; **Simplicity** describes how simple is to create decentralized apps (dApps) on it; **Sustainability** is referred to the impact that heavy usage of blockchain has on the environment; **Cost** is not about the implementation cost, because it is low using any of the alternatives, but instead it refers to the transaction cost to the user in which each blockchain has a different strategy. Ultimately, between the alternatives are the 5 most used blockchains by the date of 2022/2023: Cardano, Ethereum, Binance Smart Chain, Ripple, and Solana.

Starting with the AHP, the following table (4.1) is the second step of AHP since the first one was the development of the hierarchical structure. In this step, it is determined the relative importance of different criteria in relation to each other by making pairwise comparisons.

Table 4.1: Pairwise comparison table

	Reliability	**Decentralized**	**Simplicity**	**Sustainability**	**Cost**
Reliability	1	5	7	4	5
Decentralized	1/5	1	3	1/3	1
Simplicity	1/7	1/3	1	1/7	1/6
Sustainability	1/4	3	7	1	1/2
Cost	1/5	1	6	2	1

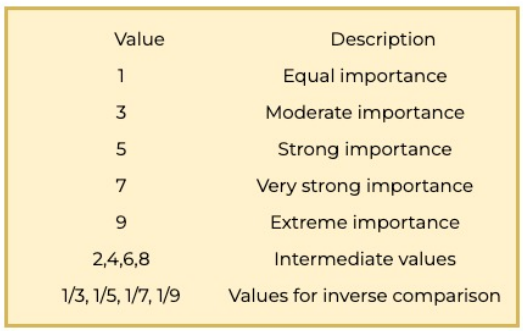
The pair-wise matrix is created with the assistance of a scale of relative importance (Figure 4.5) where 1 is applied for similar elements, 3 is placed when the criterion has moderate importance, 5 when it has strong importance, 7 for elements with very strong relative importance and 9 when the element has extreme importance. In between, there are intermediate values (2, 4, 6, and 8). These values are related to the corresponding element on the row, i.e, Reliability has a strong importance relative to Decentralization so the value is placed on the cell where the Reliability row is and on the Decentralized column. In the same way, the inverse comparison needs to be done by placing 1/5 in the cell where the row is corresponding to Decentralization and the column to Reliability. This process repeats as shown in the table.

Note: For all the cells where the row's identifier is the same as the column's, the value placed must be '1'.

Table 4.2: Pairwise comparison table: Columns sum

	Reliability	**Decentralized**	**Simplicity**	**Sustainability**	**Cost**
Reliability	1	5	7	4	5
Decentralized	0,2	1	3	0,33	1
Simplicity	0,14	0,33	1	0,14	0,17
Sustainability	0,25	3	7	1	0,5
Cost	0,2	1	6	2	1
Sum	1,79	10,33	24	7,47	7,67

[7] As previously seen, there are blockchains that, although they have a decentralized voting system, are restricted to a small number of nodes, which leads to a more centralized blockchain.



Value	Description
1	Equal importance
3	Moderate importance
5	Strong importance
7	Very strong importance
9	Extreme importance
2,4,6,8	Intermediate values
1/3, 1/5, 1/7, 1/9	Values for inverse comparison

Figure 4.5: Relative Importance: table matching the importance to its value

Table 4.2 has the values from the first table but with the decimal format to easily understand it. Also, it shows the sum of the values of the cells from each column. This is an important calculation for the next step.

Table 4.3: Pairwise comparison table: normalization

	Reliability	**Decentralized**	**Simplicity**	**Sustainability**	**Cost**
Reliability	0,56	0,48	0,29	0,54	0,65
Decentralized	0,11	0,1	0,13	0,04	0,13
Simplicity	0,08	0,03	0,04	0,02	0,02
Sustainability	0,14	0,29	0,29	0,13	0,07
Cost	0,11	0,1	0,25	0,27	0,13

The current step corresponds to the matrix normalization. It's important to ensure that the values in the table are comparable and can be meaningfully compared across rows and columns. To do this the values of each column are divided by the sum of that column, calculated in the earlier step.

Table 4.4: Pairwise comparison table: Criteria Weights calculation

	Reliability	**Decentralized**	**Simplicity**	**Sustainability**	**Cost**	**Criteria Weights**
Reliability	0,56	0,48	0,29	0,54	0,65	0,5
Decentralized	0,11	0,1	0,13	0,04	0,13	0,1
Simplicity	0,08	0,03	0,04	0,02	0,02	0,04
Sustainability	0,14	0,29	0,29	0,13	0,07	0,18
Cost	0,11	0,1	0,25	0,27	0,13	0,17

Now it's time to get the criteria weights. This can be done by determining the average of each row. The result accurately reflects the relative importance of the criteria. The outcome from table 4.4 is

- Reliability - 0,5
- Decentralization - 0,1
- Simplicity - 0,04
- Sustainability - 0,18

- Cost - 0,17

Table 4.5: Pairwise comparison table: initial pairwise table multiplied by criteria weights

Criteria Weight	0,5	0,1	0,04	0,18	0,17
	Reliability	**Decentralized**	**Simplicity**	**Sustainability**	**Cost**
Reliability	0,504	0,508	0,270	0,737	0,857
Decentralized	0,101	0,102	0,116	0,061	0,171
Simplicity	0,071	0,034	0,039	0,026	0,029
Sustainability	0,126	0,305	0,270	0,184	0,086
Cost	0,101	0,102	0,231	0,368	0,171

The initial table is now multiplied by the weights so in the next step the ratio can be determined.

These assessments can result in inconsistent judgments due to various factors such as cognitive biases, subjectivity, and misunderstanding of the criteria or alternatives so extra calculations need to be done to ensure this consistency.

Table 4.6: Pairwise comparison table: ratio calculation

	Weighted Sum Value	**Criteria Weights**	**Ratio**
Reliability	2,88	0,5	5,70
Decentralized	0,55	0,10	5,42
Simplicity	0,20	0,04	5,13
Sustainability	0,97	0,18	5,27
Cost	0,97	0,17	5,68

It is calculated the total score for each alternative, i.e, the sum of the weighted performance scores of each alternative's row across all the criteria (Weight Sum Value). The ratio is the division of the Weighted Sum Value by the criteria weight.

It needs to be determined the λ_{MAX}, the Consistency Index, and the Consistency Ratio. Ultimately the Consistency Ratio needs to be less than 0,1. The consistency ratio is calculated using the following formula:

$$CR = \frac{CI}{RI}$$

Where:

- CR = Consistency Ratio
- CI = Consistency Index
- RI = Random Index

The Random index is constant and can be obtained in the following table, where 'n' is the number of criteria, in the case of this project, $n = 5$, so the $RI = 1.12$:

The Consistency Index can be resolved by the following formula:

n	1	2	3	4	5	6	7	8	9	10
RI	0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49

Figure 4.6: Random Index Table

$$CI = \frac{\lambda_{MAX} - n}{n - 1}$$

Finally, λ_{MAX} is the average of the ratios obtained in the last table. Then,

$$\lambda_{MAX} = \frac{5,702 + 5,415 + 5,126 + 5,270 + 5,682}{5} = 5,439$$

$$CI = \frac{5,439 - 5}{4} = 0,110$$

$$CR = \frac{0,110}{1,12} = 0,098$$

Since $CR = 0,098$ which is < 0.1, the result can be considered consistent and it can be applied to TOPSIS to the alternatives based on their proximity to an ideal solution.

Table 4.7: Comparison table between criteria and alternatives

	Reliability	**Decentralized**	**Simplicity**	**Sustainability**	**Cost**
Cardano	5	5	5	5	3
Ethereum	5	5	3	5	1
Binance Smart Chain	5	3	3	5	2
Solana	3	5	3	5	4
Ripple	5	3	4	1	5
$\sqrt{\sum_{j=1}^{n} x_{ij}^2}$	109	93	77	101	55

The relation between the criteria and the 5 alternatives is made on a scale from 1 to 5, not exactly in terms of greatness but in terms of being more suitable. For example, for Reliability, Decentralization, Simplicity, and Sustainability, the scale between 1 to 5 corresponds, respectively, to less Reliable/Decentralized/Simple/Sustainable and to more Reliable/Decentralized/Simple/Sustainable. However, for the Cost comparison, the scale of 1-5 represents the best option, i.e, '1' corresponds to more expensive, and '5' to cheaper. The justification for these assessments was given in chapter 3 while comparing each blockchain with Cardano. So, in this chapter, it will only be justified the values that affect negatively the evaluation.

The criterion **Cost** has a different value for each alternative. This is because of the transaction cost. On average, Cardano has a transaction cost of 0,17ADA; Ethereum of 0.00045ETH; Binance Smart Chain (BSC) of 0.0007BNB; Solana of 0.000011SOL; and

Ripple of 0.00016XRP[8]. Each of these values is represented in the native token of the respective network and the current value depends on the current price of the token when converted to Dollars, even the transaction cost depends on the network congestion (the more congestion, the higher the costs). That's why the values are the average ones at the time of writing the present document (Fev 2023). Looking at the historical prices of the native tokens of these blockchains (ADA, ETH, BNB, SOL, and XRP) they tend to climb and fall in the same proportion, which means that, converting the token prices to Dollars, it is expected to get the same comparison result in different time frames [9]. The only way to correctly order the cost of sending a transaction in each of the networks is by converting it to a common unit, which can be the dollar. Below are the prices converted to dollars:

- (Cardano) 0,17ADA -> $0.07
- (Ethereum) 0.00045ETH -> $0.75
- (Binance Smart Chain) 0.0007BNB -> $0.22
- (Solana) 0.000011SOL -> $0.00026
- (Ripple) 0.00016XRP -> $0.000064

Ordering the blockchains by its transaction fee would be Ethereum (1) > BSC (2) > Cardano (3) > Solana (4) > Ripple (5). This way, the higher the price, the greater the evaluation.

Regarding the evaluation of Ethereum, Solana, and BSC about **Simplicity**, it is needed to note that the process of generating a new ticket in the Web3 Ticket platform will take advantage of the functionality of minting an NFT and this process is more complex in these blockchains because they require the programming of a new Smart Contract [51] [74] [40], while the other two just deal with common transactions [20] [38].

Table 4.8: Comparison table between criteria and alternatives: normalization

	Reliability	**Decentralized**	**Simplicity**	**Sustainability**	**Cost**
Cardano	0,479	0,518	0,606	0,498	0,405
Ethereum	0,479	0,518	0,364	0,498	0,135
Binance Smart Chain	0,479	0,311	0,364	0,498	0,270
Solana	0,287	0,518	0,364	0,498	0,539
Ripple	0,479	0,311	0,485	0,100	0,674

Just like in the previous AHP analysis, a normalization is needed using the sum value from each column. This makes the values comparable. In the next step, these values will be multiplied by the weights given to each criterion.

[8] These values were captured by making transactions in each of the blockchains at the same hour of the day, 3 times (morning, midday, and night)

[9] The price history can be consulted at coinmarketcap.com, as well as the current price of the tokens

Table 4.9: Comparison table between criteria and alternatives: applying the weights

Weightage	0,504	0,102	0,039	0,184	0,171
	Reliability	**Decentralized**	**Simplicity**	**Sustainability**	**Cost**
Cardano	0,241	0,053	0,024	0,092	0,069
Ethereum	0,241	0,053	0,014	0,092	0,023
Binance Smart Chain	0,241	0,032	0,014	0,092	0,046
Solana	0,145	0,053	0,014	0,092	0,092
Ripple	0,241	0,032	0,019	0,018	0,115

In this step, to balance the assessments to the correct weight, it will be used the weights from the AHP analysis that had their consistency verified and have the correct importance between them.

Table 4.10: Comparison table between criteria and alternatives: calculation of the Ideal Best and Ideal Worst

	Reliability	**Decentralized**	**Simplicity**	**Sustainability**	**Cost**
Cardano	0,241	0,053	0,024	0,092	0,069
Ethereum	0,241	0,053	0,014	0,092	0,023
Binance Smart Chain	0,241	0,032	0,014	0,092	0,046
Solana	0,145	0,053	0,014	0,092	0,092
Ripple	0,241	0,032	0,019	0,018	0,115
V_j^+	0,241	0,053	0,024	0,092	0,115
V_j^-	0,145	0,032	0,014	0,018	0,023

The ideal (best and word) values are taken for each column. V_j^+ corresponds to the ideal best value and V_j^- to the ideal worst one. These values are essential because they will be used to calculate the Euclidean Distance in the next step.

Table 4.11: Comparison table between criteria and alternatives: calculation of Euclidean Distance

	S_i^+	S_i^-
Cardano	0,046	0,132
Ethereum	0,093	0,123
Binance Smart Chain	0,073	0,123
Solana	0,100	0,103
Ripple	0,076	0,134

The Euclidean Distance for the best and worst distances are calculated by the following formulas:

$$S_i^+ = \sqrt{\sum_{j=1}^{m}(V_{ij} - V_j^+)^2}$$

$$S_i^- = \sqrt{\sum_{j=1}^{m} (V_{ij} - V_j^-)^2}$$

Table 4.12: Comparison table between criteria and alternatives: Performance Score and Rank

	S_i^+	S_i^-	**Sum**	P_i	**Rank**
Cardano	0,046	0,132	0,178	0,741	1
Ethereum	0,093	0,123	0,216	0,570	4
Binance Smart Chain	0,073	0,123	0,196	0,628	3
Solana	0,097 0,120	0,203	0,508	5	
Ripple	0,080 0,119	0,210	0,636	2	

In this last step, the objective is to calculate the Performance Score for each one of the alternatives and it can be done using the following formula:

$$P_i = \frac{S_i^-}{S_i^+ + S_i^-}$$

Lastly, with the values taken from the Performance score, it is visible that, between Ethereum, Binance Smart Chain, Solana, and Ripple, Cardano is analytically the preferable option to develop a dApp, followed by Ripple despite its validation insecurity.

4.3.3 Concept Definition

The concept has a clearly defined format, including written and visual descriptions that highlight its key attributes and consumer benefits as well as a thorough comprehension of the required technology. [42]

Beginning the concept, three important entities correlate with each other: the artist, the fan, and the place's organization. The workflow is simple as the artist creates the show that needs to be accepted by the chosen place's organization. Upon creation, the ticket is scheduled to appear on the user's home page at a certain date. After the user purchases a ticket, it remains in the virtual wallet until it is used. The user can put the purchased ticket for sale, at any price, even if it's a ridiculously high price because it will be the market (people's behavior) to decide if the ticket is worth that amount of money or not. At every second market sale (the sale made by the user) there will be two types of fees: maintenance fee taken by Web3 Ticket to allow the project to proceed and the royalties that the artists need to set upon the creation of the show. Let's suppose the user sells a ticket for 100$, the artist set the royalties to 3% and Web3 Ticket takes 2 more percent, then the seller will only receive 95$ for the sale.

Blockchain is the ideal solution because it natively allows its users to create NFTs that can generate royalty income and since it is ideally distributed by many computers around the world, it has the abstraction of being endlessly scalable. Creating a marketplace dedicated to ticket selling, and providing a way of validating the ownership of a ticket, there is a complete ecosystem based on blockchain.

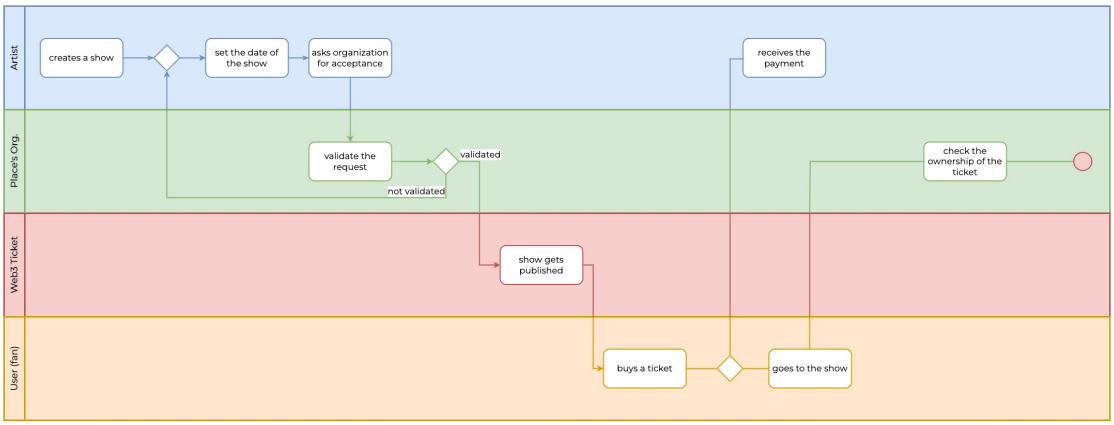


Figure 4.7: Process of buying a ticket directly to the artist and using it in the show. This is the core process of this project.

4.4 Value

Because of the way our society is structured, the word 'value' is commonly associated with money, however, the value for the customer is much more than that, even if it is the factor that defined the final price of the product or service. The consumer value, more than the amount of money that a customer is willing to spend on a product, is measured by what the customer is willing to give up. In his book, Consumer Value: A Framework for Analysis and Research, Morris Holbrook defines consumer value as an "interactive relativistic preference experience", which means that there is an interaction between the customer and the product (**interactive**); there is a moment and context to use the product so the value is transitory (**relativistic**); there is a preference judgment related to a choice, associated with feeling as like/dislike and approach/avoidance, if something is good/bad or favorable/unfavorable, making the customer create opinions about pros and cons or positive/negative (**preference**); and there is the 'proposition that all products provide services in their capacity to create want-satisfying experiences' [35]. Regarding this topic, and to measure the product in the terms of value produced, the previous book (Morris Holbrook, Consumer Value: A Framework for Analysis and Research) identifies the following types of value that will be explained individually:

- Efficiency - this type of value can be measured in terms of inputs and outputs ratio (O/I ratio). For example, when a customer chooses a computer because of its processor, he/she might take into consideration the amount of energy used to complete certain tasks. But the key input in this kind of comparison is the time, constituting the main concern.
- Excellence - is observable when the user likes a product or service for its ability to fulfill a purpose, carry out a task, or for its value as an experience.
- Status - the status is acquired when the consumer adjusts his/her consumption habits to influence other people
- Esteem - intimately related to the status, the esteem-as-value happens when the customer appreciates his/her own consumption or lifestyle without actively influencing other people. It is still a function of status, but where other people are not involved. One example of it is the desire of owning possessions just because of it.

- Play - as the name suggests, this is a type of value characterized by the fun provided by the product/service. Like the following 3 value creation factors, this is a self-oriented experience.
- Aesthetics - this can be seen as the beauty of a product, but a more in-depth description defines that this beauty is related to its intrinsic value, without expecting any other use than the one it was designed for since it is self-justifying, ludic, or autotelic.
- Ethics - this is related to the consequences that the use of the product will take on others and how they will react to it. A product or service that put someone else at risk may keep some customers away.
- Spirituality - it can be seen as a counterpart to ethics and is the admiration of super-natural power. The tarot cards are a good example of this.

For the present work, it will be taken into account the efficiency, excellence, esteem, play, and aesthetics. Efficiency because with fewer steps the client can get the process of buying, selling, or validating the tickets done. Excellence, given that the objective is to fulfill an already existing need. Esteem as is expected by user to feel good using the Web3 Ticket, since it is supposed to solve his/her problem. Play because it will be fun to compete for a limited number of tickets being for sale. And then Aesthetics, where in addition to its well-formed design, the user will use the service for its intrinsic value.

There is another important concept related to perceived value, studied by many authors, among them Zeithaml, that started the explanation about perceived value with 4 premises: "value is low price"; "value is whatever I want in a product"; "value is the quality I get for the price I pay"; and "value is what I get for what I give". Those four assertions led the author to reach a new conclusive one: "perceived value is the consumer's overall assessment of the utility of a product based on perceptions of what is received and what is given" [77].

From this perspective, all the concepts of value can be seen as the relation between the benefits, like the revenue extracted from it, the reduction of risk or the operational benefit, and the sacrifices, which can include the learning curve, the search for the better product, the time spent, the switching cost, the risk associated and the financial cost. All the previous examples are associated with the direct benefits and sacrifices, but there is also the indirect value, where the benefits are the reputation, relationship, or environment-related, and the sacrifices are the relationship side effects, physiological issues, or the loss of 'power' [14]. Figure 4.8 resumes the meaning of customer value, pointing to the benefit and the sacrifices that the user can pull from it:

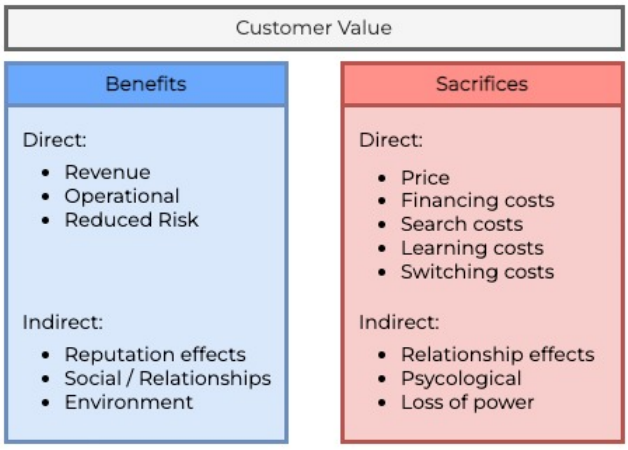


Figure 4.8: Customer Value: benefits and sacrifices

4.4.1 Value Proposition

Developed by Dr. Alexander Osterwalder, the Value Proposition Canvas is a tool that allows the company to understand how the product fits the market. It takes into account the Customer Segments and Value Propositions from the Business Model Canvas and details them in two separate diagrams [36].

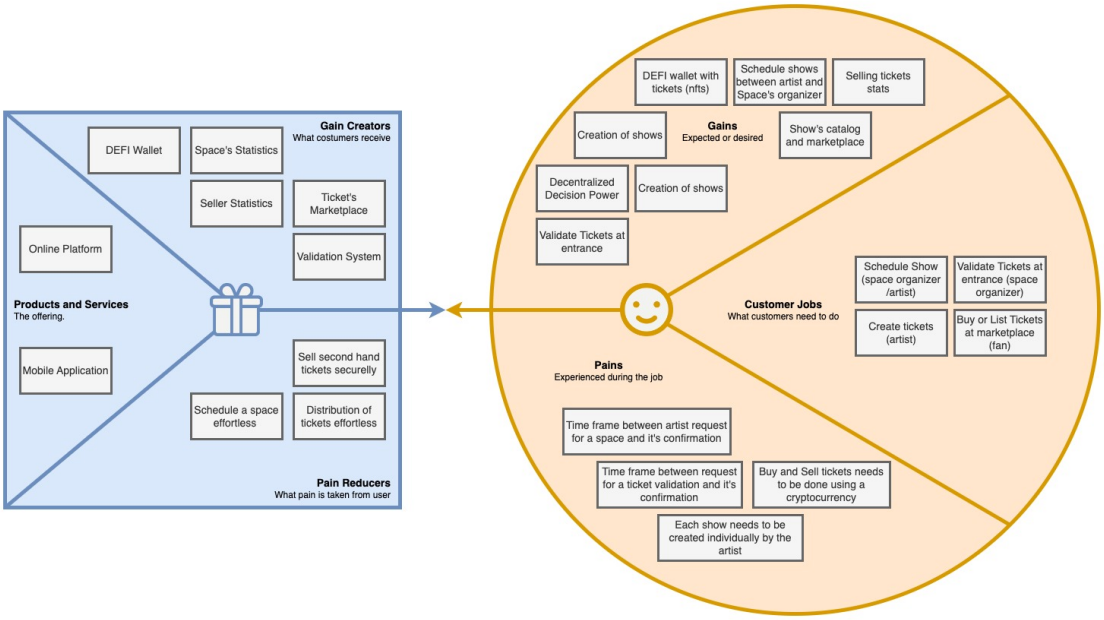


Figure 4.9: Value Proposition

Figure 4.9 demonstrates how the Pain Reducers (on the left) will relieve the pains described in the right section. In this case, since the user cannot sell his/her tickets freely in the current systems, it's pain reliever is the possibility to sell the ticket on a proprietary marketplace; the current process of scheduling a show is painful and slow, which can be solved with real-time requests through the platform; and the ticket distribution is made online using websites like Ticketline and Blueticket, and its promotion needs to be made in each one of the platforms,

what can translate in high costs. The solution is to have only one place to advertise and where all the fans can buy the tickets.

Business Model Canvas

The Business Model Canvas is more complex than the previous diagram of Value Proposition because, in addition to the Customer Segments and Value Propositions, Business Model Canvas, also includes seven other components that help to easily set the business idea and concept [63]:

- Key partners - the entities from where the resources come from. In this project, this represents the artists, coliseums, artist's managers, event organizers, and influencers.
- Key activities - the activities required to create value. For Web3 Ticket, it is required a continuous development of the platform in terms of the distribution system, ticket validation, blockchain technologies, marketplace, creation system decentralized wallet, and the bridge between the artist and the place's organization.
- Key resources - the resources that will be used to achieve the value. The resources selected to make the project happen are the internet provider, equipment to host the web server and the database, a blockchain node that depends on the selected network, an IPFS node to store the images freely, and a DB sync node to save money on requesting information to the blockchain.
- Value proposition - the value that is being delivered to the consumer. The value proposition is brought by the platform that contains the dedicated blockchain wallet, statistics for both artist and place responsible, ticket creation, its scheduling, and distribution, ticket selling at a marketplace, the validation system to prove its ownership, and the decentralized decision power.
- Customer relationships - the kind of relationships that is being kept with the customers. There will be direct contact with the artists since they are the engine of this platform. But there will be also a commitment with the entities responsible for the place because they will have a big learning curve to switch to this completely different system.
- Channels - how the public is going to be reached. The interaction between the public and the platform will be done through an app and a website.
- Customer segments - the public for whom the product is being made. There are taken into account 3 entities which are the artists, the fans, and the show places.
- Cost structure - the most important cost adjacent to the business. The cost structure is composed of the equipment enumerated in the Key Resources, and the marketing-related activities.
- Revenue streams - the sources where the income comes from. There are two sources of income, one from the second market sales, that will be charged in 2%. And the other is from helping artists to appear at the top of the page

4.4.2 Quality Function Development

Quality Function Development (QFD) is a design system, created in Japan, used to match customer demands to functional requirements, taking into account the Voice of the Customer (VOC) to get effective responses to customer needs in return. [33] Quality expresses

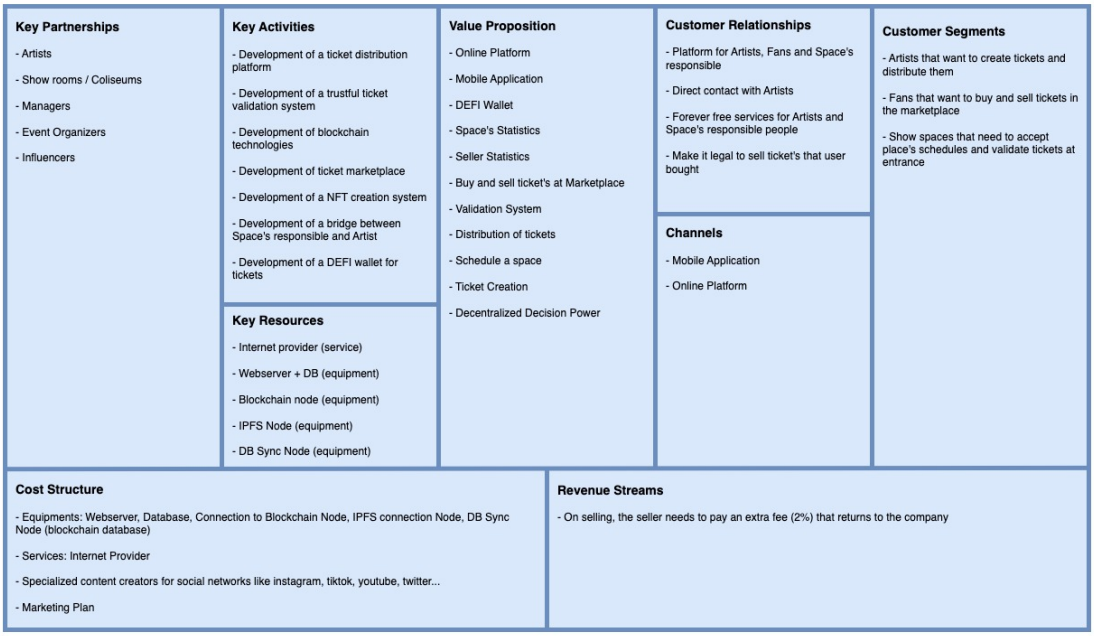


Figure 4.10: Canvas Model

the meeting of customer requirements; Function consists in what needs to be done, with a focus on consumer's needs; and Deployment sets when and who will do the activities needed to meet the consumer's needs [58].

QFD has three purposes in its implementation, which are the increase of speed and cost reduction on creating new products for the market, developing a product with customer-oriented design, and since it provides a development tracking system, future designs, and process improvements get easier [58].

Using this technique is expected to get a better understanding of consumer needs, improve the organization on development projects, mitigate production problems, prepare the product to have fewer conceptual changes, get a reputation for quality assurance, get an overall better success rate, and to provide the whole team a good documentation about the ongoing work [58].

The House of Quality (HOQ), also known as the product planning matrix, is an important stage of the QFD:

The presented HOQ is characterized by the Demanded Quality, also known as Consumer Requirements and "Whats", and by the Quality Characteristics, also known as Functional Requirements and "Hows". The list below enumerates all the demanded quality, as well as their importance:

- Ticket Decentralization (6)
- Tickets on Pocket (7)
- Royalties Redistribution (9)
- Ticket Selling (9)
- Fees Reduction (9)

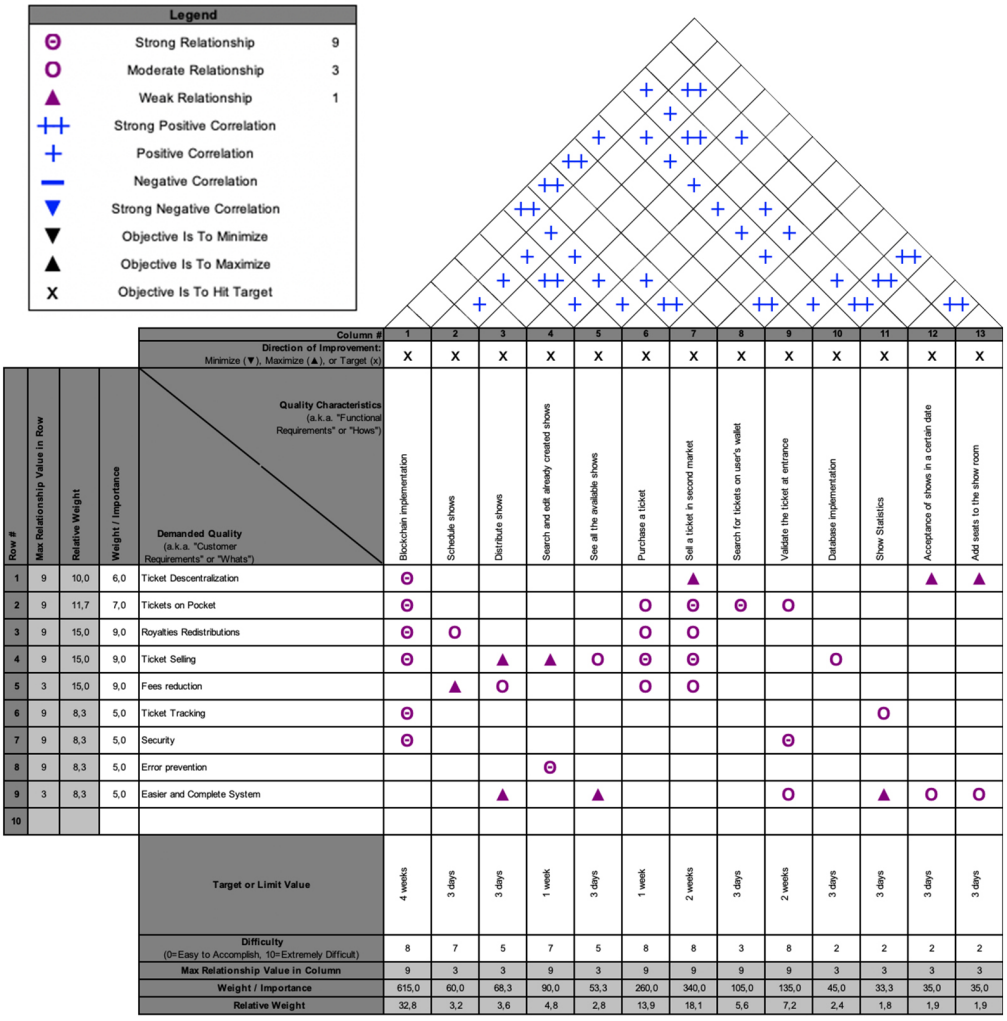


Figure 4.11: House of Quality

- Ticket Tracking (5)
- Security (5)
- Error Prevention (5)
- Easier and complete system (5)

To match the consumer requirements there are the functional requirements, that explain how the expected additional value will be created:

- Blockchain implementation
- Allowing artists to schedule shows
- Automatically distributing shows among users (fans)
- Allowing artists to search and edit the already-created shows
- Showing all the available shows
- Purchase a ticket

- Sell a ticket in the second market (resell)
- Search for tickets on the user's wallet
- Validate the ticket at the entrance
- Database implementation
- Showing statistics about the shows and places
- Allowing the space's organization to accept shows on a certain date
- Allowing the space's organization

Chapter 5

Development Process and Implementation

The development of the project marks the phase in the realization of a practical solution aimed in the previous chapters. This chapter's goal is to provide a detailed exploration of the development process attempted to create the app, highlighting the key design decisions, development methodology, and implementation details. Thus, it provides a comprehensive understanding of the steps involved in transforming the initial concept into a functional application that tries to create an abstraction between the process of holding a ticket for a show and the blockchain.

The main goal of Web3Ticket is to provide users with a user-friendly platform to search and look at show details, purchase, and even sell tickets to other users, for different events. As such, special attention was given to the system architecture, user interface / experience, and to the main functionalities to ensure an intuitive and meaningful user experience. This chapter will focus on the technologies used, and the challenges faced throughout the development process.

Overall, this chapter serves as a bridge between the conceptualization of Web3Ticket and its tangible realization. It illustrates the transformation of ideas into a practical solution, heading the way for a deeper exploration of the blockchain features, impact, and future potential in the next internet era.

5.1 Development Methodology

The waterfall methodology was chosen as the development approach for the Web3Ticket app. The waterfall methodology is the most suitable approach because it is a linear and sequential process that follows a predefined set of phases, including requirements gathering, design, implementation, testing, and deployment, not exactly in this order.

The first phase of the waterfall methodology involved the design of the user experience because it proved to be a very effective method of gathering all the features of the software and hence the requirements. In this case, Figma was used to design the UI/UX (User Interface/User Experience) and then each requirement was listed in Trello which played a vital role in facilitating the development process, mainly in task management, and progress tracking.

Once the requirements were defined, the design phase commenced. In this phase, the app's architecture, and data models were created. The design decisions were based on the gathered requirements and aimed to ensure that the app was efficient, the server responded

quickly, and all blockchain capabilities were explored. This is also meant to test the viability of the decisions. A small project was created using Python as the main server programming language and web technologies such as HTML (Hypertext Markup Language) and JavaScript to simulate a marketplace resorting to the Cardano blockchain network [1].

After the design phase, the implementation phase began, where the actual development of the Web3Ticket app took place, although most of the code that makes the connection to the blockchain was already done in the small test project. This phase required adherence to coding standards and best practices to ensure maintainability and scalability.

Once the implementation was completed, the focus shifted to the deployment phase (instead of the testing phase). The app was deployed to a production environment, making it accessible to users. The deployment involved configuring a Raspberry server, setting up the database, connecting a dedicated IPFS (InterPlanetary File System) node, and ensuring the app's compatibility with the PWA (Progressive Web Apps) format.

After completing the deployment phase, the focus shifted to the testing phase. The app underwent rigorous testing to check the defects or issues and to ensure that the concept was viable in the real world. User acceptance testing, along with QEF, was employed to ensure the app met the specified requirements and performed as expected.

5.2 System Design

The system combines Python with the Django framework for server-side development, along with React Native for mobile application development. The Django server serves as the core component, providing a robust RESTful API that seamlessly connects with an SQLite database for reliable data storage and retrieval. The integration of the IPFS network ensures decentralized and efficient file storage, while the utilization of the Cardano blockchain enables secure transactions, ticket ownership verification, and authenticity validation. This cohesive architecture facilitates smooth interactions between the React Native application, the Django server, the SQLite database, the IPFS network, and the Cardano blockchain, trying to ensure a seamless and secure ticket selling experience.

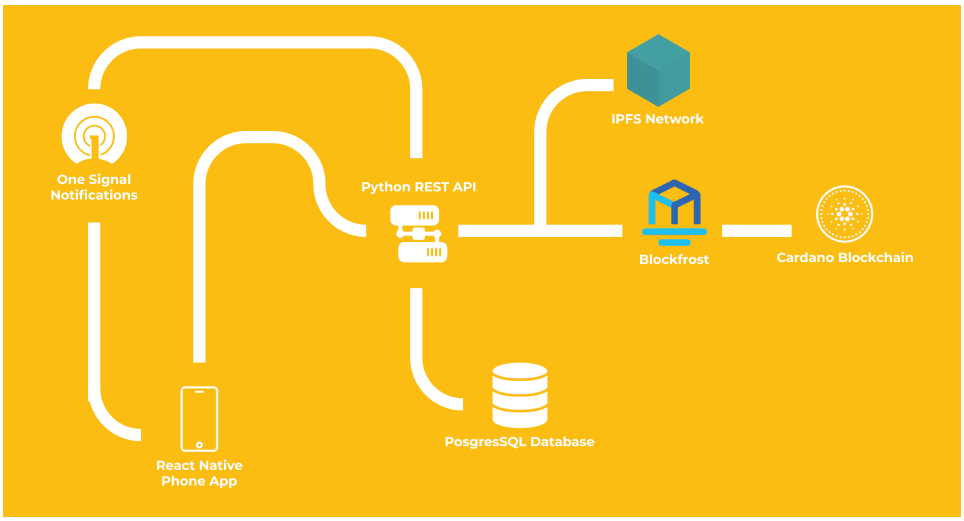


Figure 5.1: Software Architecture

[1] https://github.com/moraisandreua/cardano-marketplace

The front end of the Web3Ticket application was developed using React Native, a powerful framework, which in this project is powered by JavaScript, although it could be powered by Typescript. React Native enables the creation of a dynamic and user-friendly interface that is optimized and responsive for both iOS and Android platforms. The Figure 5.1 envisages the use of OneSignal, for example, to the artist to receive notifications every time the Place Responsible accepted or refused his/her show, and to the fan to receive a notification when his/her tickets were sold, however, the plans changed as the time was running out.

5.2.1 Database architecture

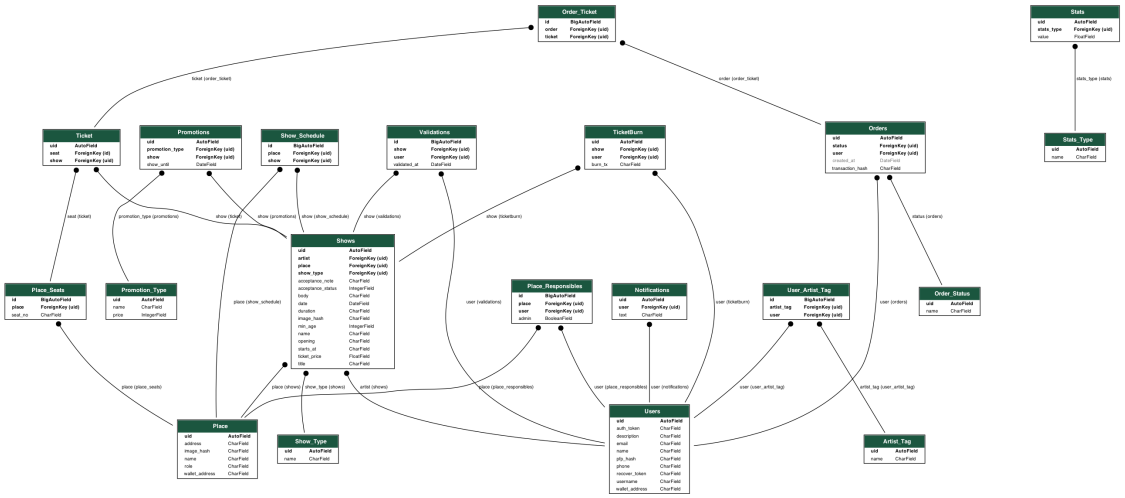


Figure 5.2: Overview of the database diagram

In this project, SQLite was chosen as the database management system (DBMS) due to its lightweight nature and seamless integration with the Django framework. Figure 5.2[2] shows a generic view of the developed database, which will be broken down over the next few paragraphs.

The 'Shows' model represents a table/entity that stores information about various shows/events. The 'artist' field is a foreign key referencing the 'Users' model, representing the artist associated with the show (any user can be an 'artist'). The field named as 'place' is a foreign key referencing the 'Place' model, representing the location of the show, which can be a showroom or a venue. Since there can be different types of shows, 'show_type' is a foreign key referencing the 'Show_Type' model, representing the type or category of the show. As illustrated in the Figure 5.3 the present relationships are One-to-Many where one 'User', 'Show_Type' or 'Place' can have many 'Shows' associated to it.

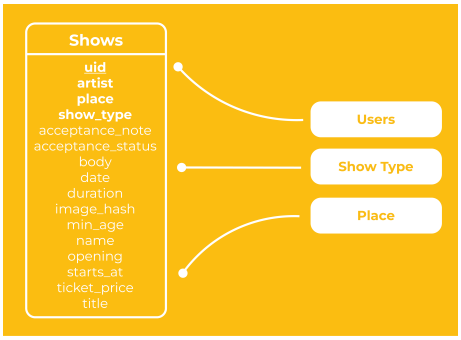


Figure 5.3: Relations between Shows entity and its Foreign Keys

[2]The Appendix A has a bigger representation of the same image

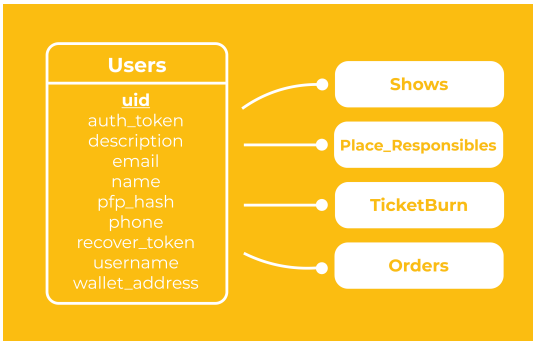


Figure 5.4: Relations between User entity and the entities related to it

The 'Users' model represents a table/entity that stores information about the users. As in the 'Show' entity (image_hash), the field 'pfp_hash' corresponds to an IPFS image hash that identifies the image in the IPFS network. The field 'wallet_address' stores the Cardano native's wallet address where the user has its funds. It doesn't give any permissions to whoever holds its value but is a way of identifying a wallet. The user's authentication token is stored in the field 'auth_token' which is random and unique for every user. This model relates to other entities such as the 'Shows', 'Place_Responsibles', 'TicketBurn', and 'Orders' in a way as the 'artist' of the 'Show' is a 'User'; the 'User' can be responsible for a 'Place' and, when validating a ticket, the database stores the request for a ticket 'burn'.

The 'Place' model represents a table/entity that stores information about the places or venues associated with shows. Since the 'Place' also needs to receive the fees from the renting of its location, the field 'wallet_address' stores the wallet address where the money goes. Just like the other models, the field 'image_hash' is used to store the IPFS hash received after uploading the image to the IPFS node. The 'Place_Seats' model makes a relation between the seat identifier and the 'Place', where the seat identifier is defined by the 'Place_Responsible', which created a relation between a User and the Place.

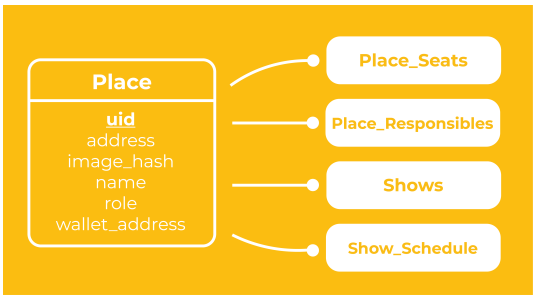


Figure 5.5: Relations between Place entity and the entities related to it

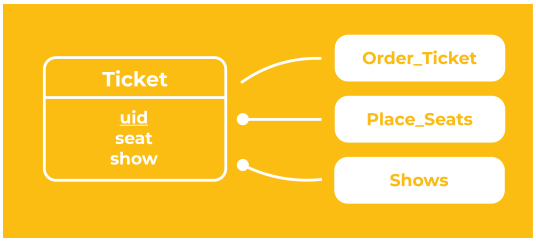


Figure 5.6: Relations between Ticket entity and the entities related to it and its Foreign Keys

The 'Ticket' model represents a table/entity that stores information about the tickets associated with shows. The field 'show' is a foreign key referencing the 'Shows' model, representing the show to which the ticket belongs, while the field 'seat' is a foreign key referencing the 'Place_Seats' model, representing the specific seat associated with the ticket. The relations with the 'Place_Seats' and 'Shows' models are Many-to-One where there can have many tickets for different shows in the same place and seat, and one show can have multiple tickets. Regarding the model 'Order_Ticket' it is an auxiliary model to allow the creation of a Many-To-Many relationship where many 'Orders' can have many tickets associated with it.

5.2.2 Ticket generation

The creation of a ticket, which is represented as an NFT, involves three entities: the user, the server, and Blockfrost, a service that facilitates connections to the Cardano blockchain. As illustrated in Figure 5.7, when a user wants to purchase a ticket, they initiate the process by making a request, telling the server who he/she is and the desirable show and seat(s).

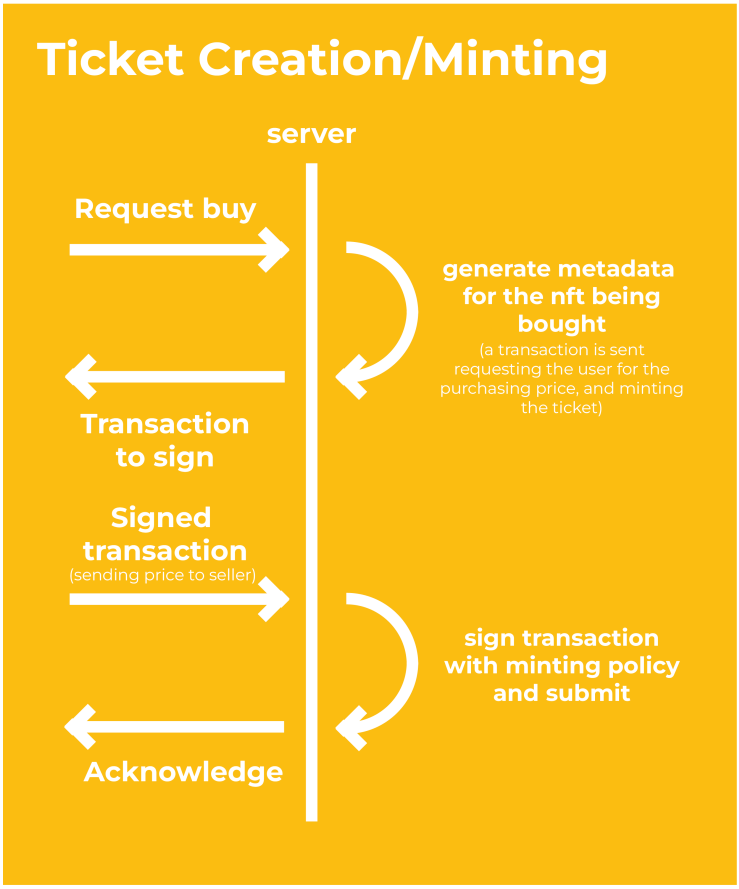


Figure 5.7: Diagram of ticket minting/creation

To create the ticket, the server generates the necessary metadata and constructs a transaction that involves two important outputs. Firstly, it charges the user the cost of the ticket, which goes from the user to the artist. Secondly, the transaction includes the ticket itself. To interact with the Cardano blockchain, the server makes use of the services of Blockfrost.

Once the transaction is constructed, the server sends it to the user. The user manipulates their private key to sign the transaction, ensuring its authenticity and security. This step confirms that the user approves the payment for the ticket and is ready to proceed.

Additionally, the server also needs to sign the same transaction but with a private key related to the show. This dual signing process is necessary because both the payment needs to be made to the artist and the ticket needs to be sent to the user within the same transaction. By signing the transaction with the show's private key, the server affirms its authorization to complete the transaction on behalf of the show. All the tickets from a certain show are signed with the same private key related to the show.

Once the transaction is fully signed by both the user and the server, the server requests Blockfrost to submit it. Upon successful submission, the user receives an acknowledgment,

confirming that the transaction has been successfully submitted, which doesn't mean that the transaction will be validated.

The limitations of this process will be discussed at the end of this chapter.

5.2.3 Marketplace architecture

The architecture of the marketplace is based on an interaction of a buyer and a seller, making use of native wallets to create a layer of trust between the two entities, as illustrated in the Figure 5.8. When a seller wants to sell a ticket, the ticket is sent to a temporary wallet. In addition, a corresponding non-fungible token (NFT) called '_info' is minted to the same temporary wallet. This '_info' NFT contains metadata that includes information about the asset being sold (the ticket), the associated show, the seller, and the seat of the ticket.

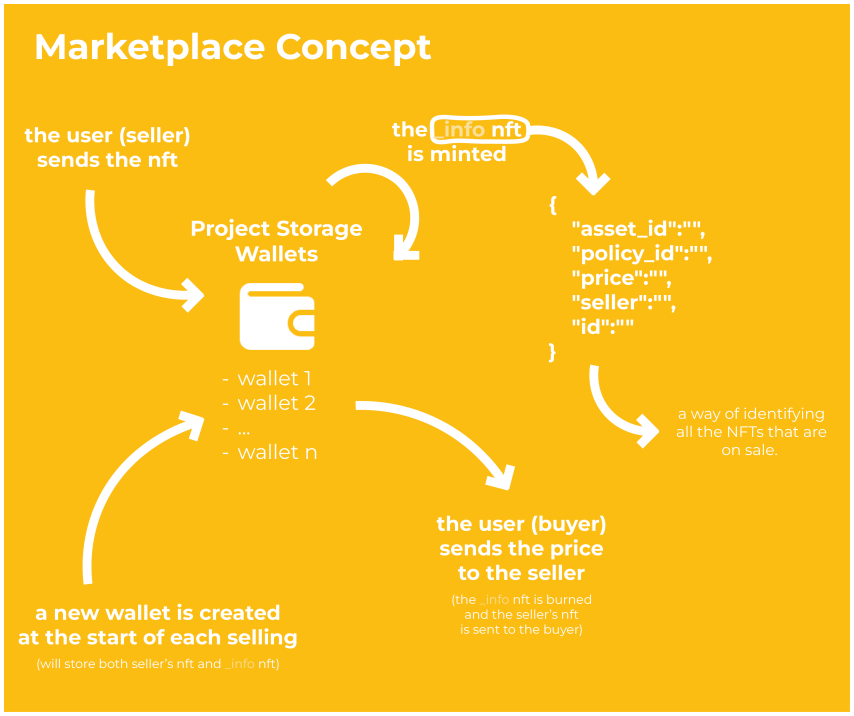


Figure 5.8: Marketplace building

The marketplace page displays the information from the '_info' NFTs that are already stored on the blockchain. Each ticket available for sale is associated with a unique temporary wallet. Thus, the number of temporary wallets increases with the number of tickets available on the secondary market.

When a buyer purchases a ticket, the payment is sent to the seller that is identified in the '_info' nft metadata. Simultaneously, the ticket is transferred from the seller to the buyer. Finally, the '_info' NFT corresponding to the ticket is burned, effectively removing it from the blockchain.

Marketplace - Sell a ticket

The sale of a ticket depicted in Figure 5.9 involves both client-side and server-side actions. The process begins when the seller, on the client side, specifies the asset he/she wishes to sell and sets the desired price. The server then takes the seller's address for further use.

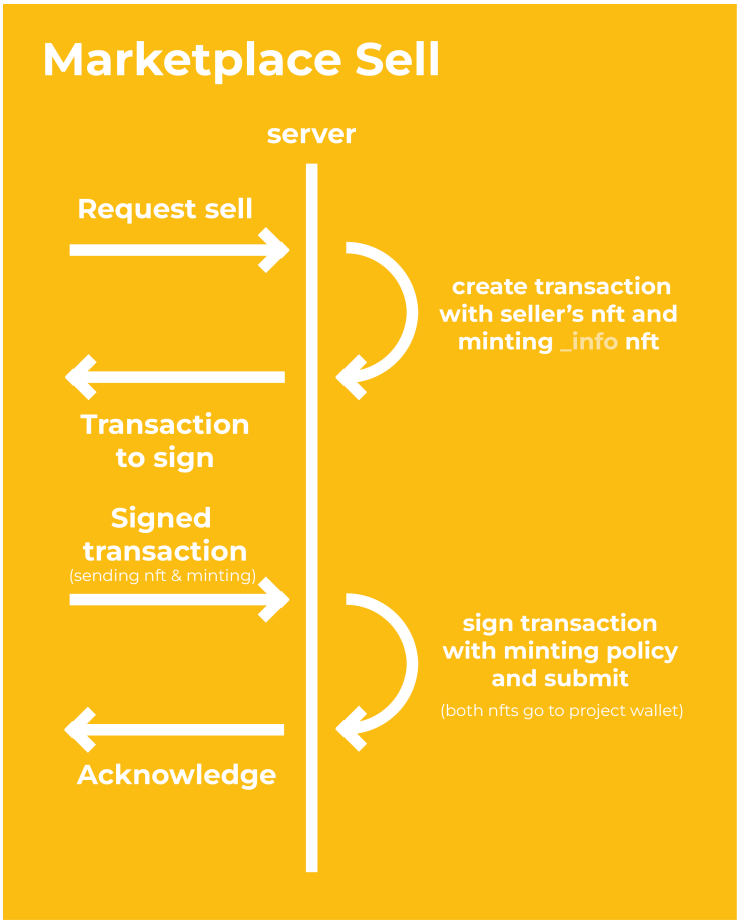


Figure 5.9: Secondary market sale workflow

Using the information provided by the user, the server proceeds to create metadata for a new NFT called '_info'. This '_info' NFT shares the same "asset name" as the ticket being sold but has a different "policy ID". Both the '_info' NFT and the ticket being sold are transferred to a temporary wallet.

The server returns the constructed transaction to the user, who is required to sign it using his/her wallet's private key. This signature serves as the user's consent to transfer the ticket to the temporary wallet. Upon receiving the signed transaction from the user, the server adds its own signature to the transaction. The server's signature is required to validate the '_info' NFT policy.

Both signatures—the user's and the server's—are necessary for the transaction to proceed. The user's signature indicates their agreement to transfer the ticket to the temporary wallet, while the server's signature verifies the authenticity of the '_info' NFT policy.

The server then attempts to submit the transaction through the Blockfrost endpoint. If the submission is successful, the server acknowledges the user about the transaction, ensuring they are informed of its progress.

Marketplace - Buy a ticket

Figure 5.10 showcases the interaction between the user and the server, leading to a ticket purchase. When a user shows interest in purchasing a ticket from the secondary market, he/she initiates the process by requesting the desired ticket from the server.

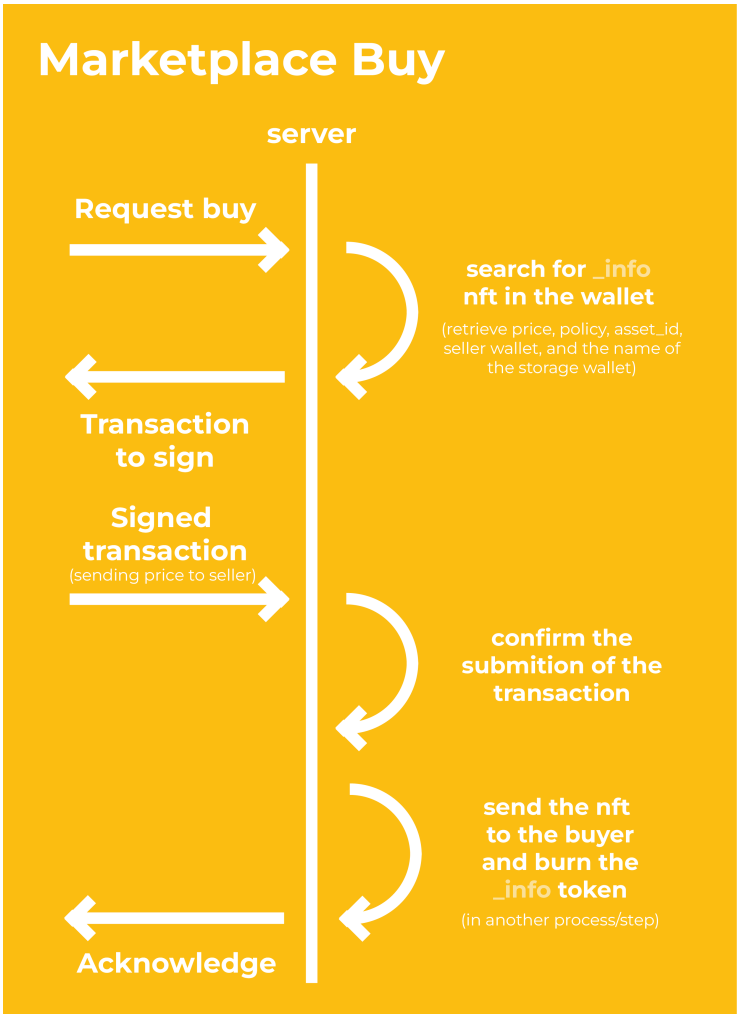


Figure 5.10: Secondary market buy workflow

To proceed with the purchase, the server searches the blockchain for the corresponding '_info' NFT, which holds vital metadata about the ticket, such as the show details, price, and seller information.

With the retrieved '_info' NFT metadata in hand, the server proceeds to build the transaction. The transaction specifies that the seller will receive a predefined amount of money, previously defined in the '_info' NFT metadata, while the buyer will obtain the requested ticket.

Once the transaction is constructed, the server sends it to the user for their signature. The user's signature serves as a confirmation of his/her agreement to the transaction terms, including the price and the ticket being purchased. The signed transaction is then returned to the server.

Next, the server takes the user-signed transaction and submits it to the blockchain using the services of Blockfrost.

Additionally, as part of the transaction, the '_info' token associated with the ticket is burned, meaning it ceases to exist. This allows the ticket to be listed multiple times in the marketplace, enhancing its availability and promoting a dynamic secondary market.

Finally, upon successful submission of the transaction, the buyer is acknowledged about the status of the transaction.

5.3 User Interface Design

Any software application's success is greatly influenced by the user interface (UI) design, which has a direct impact on how users interact with the system and their experiences using it. For users to be able to navigate through the different features of the Web3Ticket app while maintaining an abstraction from the blockchain, the user interface must be well-designed and simple to use. One of the hypothesis points that the UI design tries to implement is the bridge between the real world and blockchain, because some features are common to a blockchain user, but that a normal user is not used to. Eliminating that learning curve was therefore one of the objectives.

Figure 5.11: Web3Ticket logo

The color scheme and logo design play a major role in establishing the visual identity and recognition of the this project, including this report. The primary color, yellow ('#FFBC00'), combined with the light yellow was chosen to represent vibrancy, energy, and enthusiasm, symbolizing the dynamic nature of the ticket-selling ecosystem. Serving as the dominant hue throughout the UI, it creates a sense of prominence and draws attention to key elements. The logo design further reinforces the brand identity, incorporating the primary color in a meaningful way. The logo features a stylized combination of a "W", a "3", and a "T" to represent Web3Ticket. Additionally, the logo incorporates an appealing hexagonal shape, reminiscent of blockchain technology.

The fusion of three different apps into one cohesive platform was one of the major difficulties encountered during the creation of the application. Each of these apps targeted to particular user needs while serving a variety of functions. To enable users to access all functionalities from a single entry point without feeling disjointed or overloaded, it was challenging to effortlessly transition between the many aspects of each app.

The three apps, namely Fan's, Artist's, and Place's possessed their unique features and functionalities. The Fan's app focused on buying, selling, and using the tickets, Artist's app specialized in creating shows and sales management, and Place's app provided a way of managing the event scheduling and available seats. In their standalone forms, these apps offered valuable services to users, but integrating them effectively allows the user to act like someone responsible for a place and/or like an artist.

Within the Web3Ticket application, users have the flexibility to seamlessly transition between two distinct perspectives: the Fan view and the Artist view. This feature is facilitated through an intuitive option presented on the wallet page, labeled "Switch to artist." Whether users want to explore events, discover new artists, and purchase tickets as fans, or showcase their own performances, manage ticket sales, and interact with their audience as artists, the "Switch to artist" option serves as a gateway to effortlessly transition between these two perspectives.

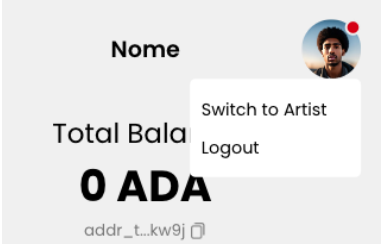


Figure 5.12: Switch to artist view

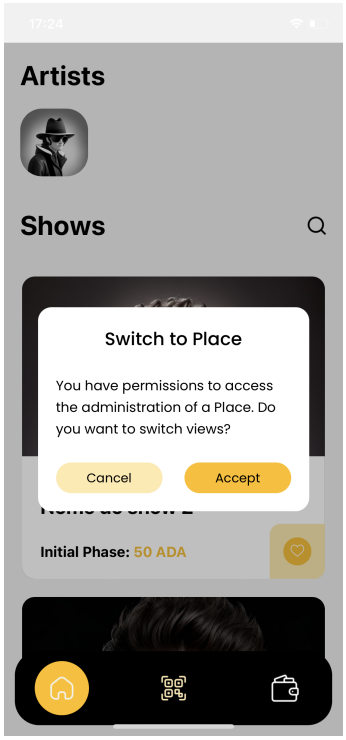


Figure 5.13: Switch to Place Responsible

The Web3Ticket application offers a seamless transition between the Fan view and the Place's Responsible view too. When a user with administrative permissions associated with a specific place accesses the home page, an option to switch to the place's responsible view becomes available. This option is presented through a convenient pop-up format, ensuring clear and intuitive navigation. By clicking on the "Accept" option, users are granted access to an interface that empowers them with administrative functionalities and tools to manage the operations and activities related to their respective places.

5.4 Implementation Details

This section explores the technical aspects of the project's implementation. It provides an overview of the underlying technologies, frameworks, and architecture employed to bring the application to life.

5.4.1 Create & Restore native wallet

To ensure that the user's funds are secured, a robust approach was implemented, needing the generation of a wallet that is securely stored on the client side. The wallets are kept locally to reduce the dangers associated with centralized storage and to provide users with complete control over their money. These wallets are created using a mnemonic, a seed phrase made up of 24 words that act as the master key for controlling the wallet and gaining access to the signing and verification keys. This seed phrase is unique to each user and must be securely stored as it grants complete control and ownership of the wallet.

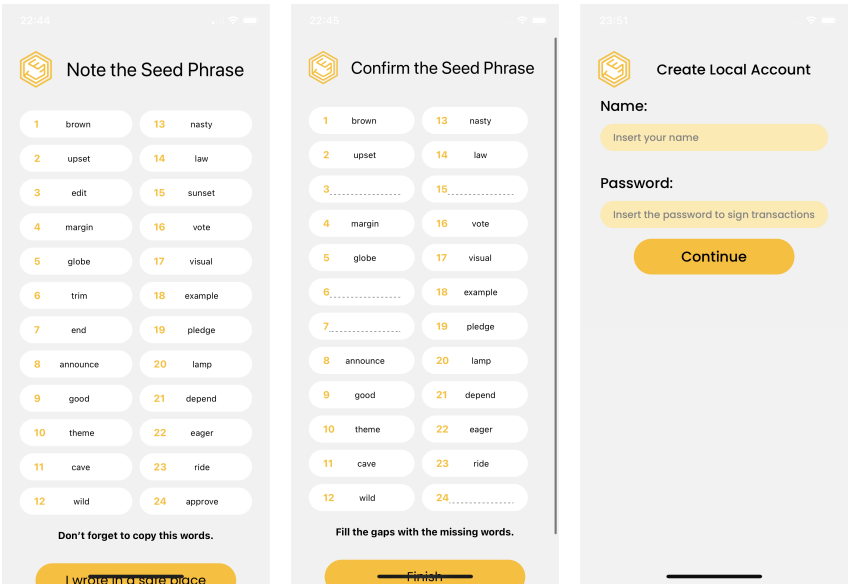


Figure 5.14: Seed phrase being shown to the user

Figure 5.14 showcases an arrangement of the mnemonic in two columns, so the user can take note of them. By having users physically write down the seed phrase, the application minimizes the risk of digital theft or loss. This offline storage method adds an extra layer of protection against potential vulnerabilities in the digital realm. This seed phrase is generated in the server because of one of the limitations explained further in this document and is returned to the user along with his/her authentication token (for further validations) and with the wallet's address.

After taking note of the seed phrase, the user is asked to confirm the seed phrase is securely stored by filling the gaps (which are selected randomly) as illustrated in Figure 5.14. And then, it is required for the user to choose a name and a password. In addition to enabling the identification of the user, the name is also used for seed phrase encryption.

The authentication token, the wallet's address, the name, and the wallet's seed phrase are stored in the device's storage through a library called 'AsyncStorage'[3] as shown in the

[3]https://www.npmjs.com/package/@react-native-async-storage/async-storage

Code 5.1. This allows to store key-value pairs in a React Native app with support for iOS, Android, Web, MacOS and Windows. The seed phrase is stored encrypted using AES (Advanced Encryption Standard) using the name chosen by the user as the 'salt' as shown in the Code 5.1 on the line 1:

```
1  const encrypted_mnemonic = encrypt(JSON.stringify(words), nameField +
       passwordField);
2  AsyncStorage.setItem( 'name', nameField ).then(()=> {
3    AsyncStorage.setItem( 'mnemonic', encrypted_mnemonic ).then(()=>{
4      AsyncStorage.setItem( 'address', address ).then(()=> {
5        AsyncStorage.setItem( 'auth_token', auth_token ).then(()=> {
6          navigation.navigate('FanHome')
7        })
8      })
9    })
10 });
```

Code 5.1: Function to store user's information in device's storage

5.4.2 Consult Balance

Each of the three entities of the app (Fan, Artist, and Place) have an area specific for managing the funds, it's the Wallet page. The user's balance is retrieved by making a request to Blockfrost API to the endpoint presented in the line 1 of Code 5.2:

```
1  // https://cardano-preprod.blockfrost.io/api/v0/addresses/{address}
2
3  {
4      "address": "addr1qxqs5...y6pz",
5      "amount": [
6          {
7              "unit": "lovelace",
8              "quantity": "42000000"
9          }
10     ],
11     "stake_address": "stake1ux3g2n...gt7",
12     "type": "shelley",
13     "script": false
14 }
```

Code 5.2: Return from Blockfrost's endpoint

The amount of money (ADA) that will be presented to the user is a fraction of the Lovelace since the Lovelace is the basic unit of ADA. The amount of Lovelace will be divided by 1000000 and the resultant number is the amount of ADA.

5.4.3 See and Copy wallet address

On the Wallet page users can access and copy their Cardano wallet address with just a few simple steps. To achieve this, the library '@react-native-async-storage/async-storage' is first used combined with State Hook to automatically update the UI once the address is retrieved, because 'async-storage' returns a Promise, which triggers a result of an asynchronous operation.

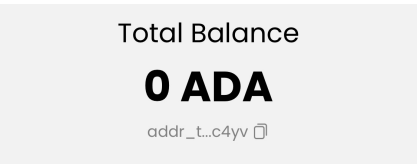


Figure 5.15: Wallet details

The Code 5.3 shows the process of obtaining and placing the address (from which only the beginning and the end are selected):

```
const [ address, setAddress ] = useState("addr1...");

AsyncStorage.getItem("address").then((data)=>{
    setAddress(data);
})

...

<Text style={{textAlign:'center', fontFamily:'Poppins', fontSize:14,
    color:'#828282'}}>{address.substring(0,6)}...{address.substring(
    address.length-4,address.length)}</Text>
```

Code 5.3: Function to gather and format the user's wallet address

The application uses the '@react-native-clipboard/clipboard' library to make it simple to copy the wallet address to the device's clipboard after it has been fetched. In addition, the copy icon present in the Figure 5.15 is also updated when clicked, i.e, when the user clicks on the address, it is copied and the icon turns into a checkmark for one second and then back to normal. Code 5.4 shows that behaviour:

```
const [ copied, setCopied ] = useState(false);

const copyAddress = () => {
    AsyncStorage.getItem("address").then((address)=>{
      setCopied(true);
      Clipboard.setString(address)

      setTimeout(()=>setCopied(false), 1000);
    })
}

...

<Image source={(copied) ? checkIcon : copyIcon} style={{width:14, height
    :14, marginLeft:5}}/>
```

Code 5.4: Code used to copy the address

5.4.4 Create Shows

The artists are asked for a variety of information, including a cover image, a title, the date and time of the event, and its duration. Additionally, artists specify the venue where the show will take place, indicate the minimum age requirement for attendees, price details, and the type of the event, whether it is indoor or outdoor. Lastly, artists provide a concise description.

The Figure 5.16 represents the screen where the artist can create the show. As it will be explained next, the artist can't have his/her show instantly approved, so the Place Responsible needs to approve/refuse it.

When the user selected an image to be the cover, this is converted into a base-64 string to be easily transferred and managed by the server. Along with the user's authentication token, all the previously mentioned parameters are sent to the server to insert the show into the database.

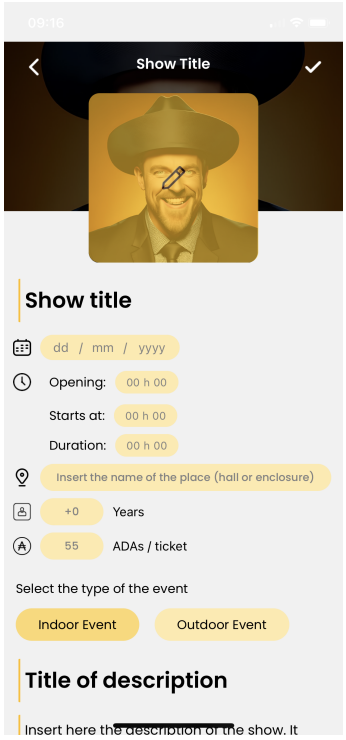


Figure 5.16: Screen where the artist can create the show

A fundamental part of the creation of a show and hence the decentralization of the application is the project storage. The use of IPFS networks allows the project to be scaled indefinitely because as the number of cover images increases, the storage space required could be a problem. This problem is mitigated, by sending the image to a pre-configured IPFS node, hosted in a Raspberry and connected to more than 300 other nodes. Code 5.5 shows how the server connects to the IPFS node to first upload the image and then pin the image to the server [4]:

Note: 'IPFS_NODE' and 'IPFS_PORT' are two environment variables defined in the '.env' file that store the address of the IPFS node. In the production version, the running server is on the same machine as the IPFS node. The response from this request is a string that corresponds to the hash that identifies the image in the network. When using this hash as the sufix in 'https://ipfs.io/ipfs/{sufix}' the IPFS public gateway return the uploaded image.

[4] Pinning a file means forcing its persistence. If a file isn't visited in a certain interval of time set by the node owner, the file is automatically removed from the network, with pinning the file persists in the chosen service which can be the local server or another associated service like pinata.cloud

```python
def pinImageToIPFS(image_hash):
    # ask for the node to pin the file locally
    response = requests.post('http://'+IPFS_NODE+':'+IPFS_PORT+'/api/v0/pin/add?recursive=true&stream=true&arg='+image_hash)


def uploadImageToIPFS(image_base64):
    image_base64_arr = re.split(r':|;|,', image_base64) # image_base64 = ...
    imgdata = base64.b64decode(image_base64_arr[3])

    # send image to ipfs server
    response = requests.post('http://'+IPFS_NODE+':'+IPFS_PORT+'/api/v0/add?stream-channels=true&pin=true&wrap-with-directory=false&progress=false', files={
        'file': ("show image", imgdata, image_base64_arr[1]),
        'Content-Disposition': 'form-data; name="file"; filename="show image"',
        'Content-Type': image_base64_arr[1]
    })

    # retrive the hash from uploaded image
    response_arr = response.text.split("\n")
    image_hash=json.loads(response_arr[0])["Hash"]

    # ask the node to pin the image
    pinImageToIPFS(image_hash)

    return image_hash
```

Code 5.5: Server side code to upload image into the IPFS network

Along with that, signing and verification keys are generated for each new show. This called 'policy' keys are required to generate the tickets in an nft format. Another file 'expiration.block' is saved for later calculation of policy id (a string that is commonly used to verify the authenticity of an nft).

```python
def createPolicyKeys(show_uid, expiration):
    PROJECT_ROOT = os.path.dirname(os.path.abspath(__file__))
    SHOW_DIRECTORY = "shows_pk/" + str(show_uid)

    basePath = PROJECT_ROOT + "/" + SHOW_DIRECTORY + "/"

    os.makedirs(basePath)

    # create policy keys
    policy_skey_path = f"{basePath}policy.skey"
    policy_vkey_path = f"{basePath}policy.vkey"

    policy_key_pair = PaymentKeyPair.generate()
    policy_key_pair.signing_key.save(str(policy_skey_path))
    policy_key_pair.verification_key.save(str(policy_vkey_path))

    # create the file with expiration block
    f=open(f"{basePath}expiration.block", "w")
    f.write(str(expiration))
```

Code 5.6: Server side code to create policy keys

Code 5.6 shows that the keys will be stored in a directory called 'shows_pk/{show uid}' with the names being 'policy.skey' for the signing key and 'policy.vkey' for the verification key. The 'expiration.block' file stores the date of the show in Unix time[5] format. The expiration block is the last block of transactions that can be used to generate a ticket. After this date/block the ticket are blocked from being generated.

5.4.5 Accept & Refuse Shows

Within the application, the users responsible for the place are presented with an overview of all show requests received from various artists. Also allows the user to navigate through the requests and access detailed information such as the proposed date, time, price, and other relevant parameters. The Place Responsible can review each request, considering various factors such as venue availability and alignment with their programming goals. Upon evaluation, the Place Responsible can either click "Confirm" to accept the show request for scheduling or "Refuse" to decline it. If the user opts to refuse a request, a popup window appears, asking the user to provide specific reasons for the decision. This mechanism removes any intermediary that could exist in the communication between the Place Responsible and the artist.

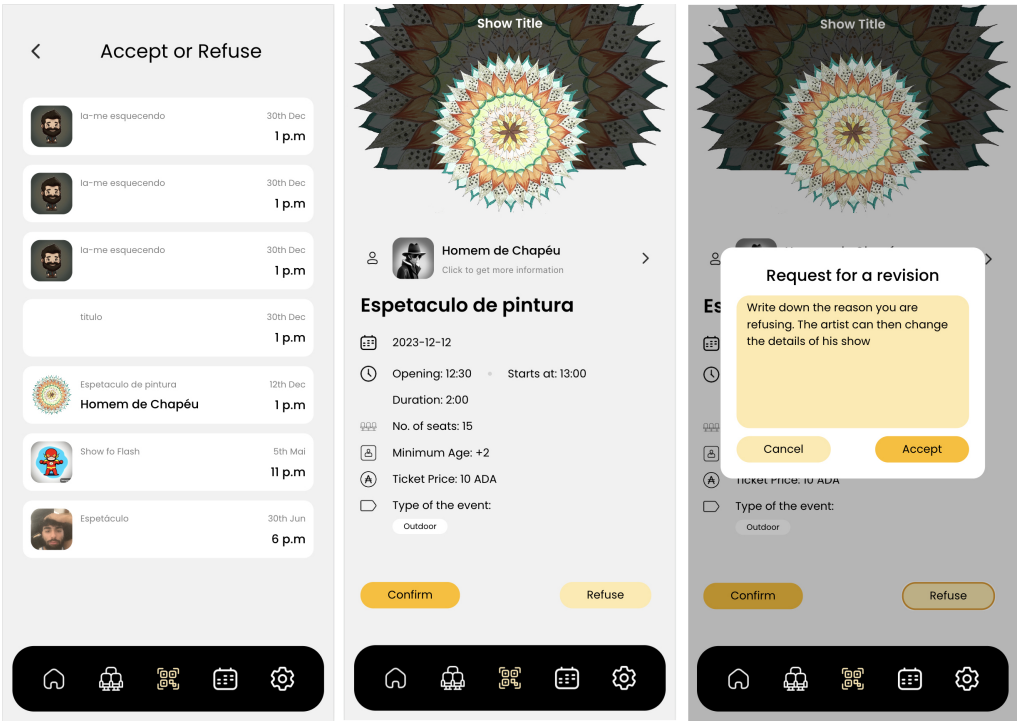


Figure 5.17: Workflow of accepting/refusing a request from an artist

Code 5.7 illustrates how the server selects the shows that haven't been reviewed yet, i.e, which status is "Requested", with value 1, as indicated in the line 2:

[5] it's a time format corresponding to the number of seconds that have elapsed since January 1st 1970, 00:00:00

```
class ShowStatus(Enum):
    REQUESTED = 1
    REVISION = 2
    PUBLISHED = 3
    REFUSED = 4

...

requests_list = Shows.objects.filter(place=place, acceptance_status=
    ShowStatus.REQUESTED.value)

shows_list=[ {
    "uid":show.uid,
    "image":"https://ipfs.io/ipfs/"+show.image_hash,
    "showname":show.name,
    "artistName":show.artist.name,
    "day":show.date,
    "hour":show.starts_at
    } for show in requests_list if show.acceptance_status==ShowStatus.
    REQUESTED.value ]
```

Code 5.7: Server side code to return the show list

When the artist creates a Show, it receives a status with the value '1' ("Requested") by default. Once the user responsible for the place affirmatively reviews the show, its status takes on another value, which is '3' ("Published"). The value '2' ("Revision") is given to the Show that has already been changed by its creator after the note left by the Place Responsible. In the last instance, the Show's status can take on the value '4' ("Refused") permanently if it is marked as spam.

5.4.6 Show Details & Redirect to map

The same way as the 'Requested' Shows appear to the Place Responsible, the 'Published' Shows are shown to the regular user (Fan).

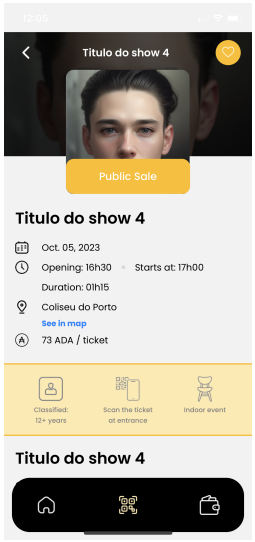


Figure 5.18: Details of an event are shown to the user

The show details screen (Figure 5.18) offers fans a view of the event they are interested in. The screen presents information such as the date, time, place, price, minimum age requirement, event type (indoor or outdoor), and the description of the show provided by the artist. Fans have the option to directly purchase tickets from the artist through the "Public Sale" button. Beyond that, the screen includes a dedicated area for fans to buy from the secondary market.

Another useful feature is the button for the maps navigation. When the user clicks on "See in map" (highlighted in blue) he/she will be redirected to the google maps app pointing to the address of the place. Code 5.8, on line 4, demonstrates how the google maps linking was implemented. The method 'openPlaceAddress' is triggered when the user clicks on "See in map".

```
1 import { Linking } from 'react-native';
2
3 const openPlaceAddress = () => {
4     Linking.openURL("https://www.google.com/maps/search/"+showDetails.
    placeAddress);
5 }
```

Code 5.8: Code to open google maps on the Show Details Page

5.4.7 Buy a ticket in primary market

The process of buying a ticket directly from the artist, also called 'minting an nft', starts by clicking the "Public Sale" button in the show details screen (Figure 1), and the user is directed to a dedicated screen where he/she can select the desired tickets. The screen features a dropdown menu that allows the user to search and choose their preferred seats from the available options, which are the seats that haven't been purchased yet.

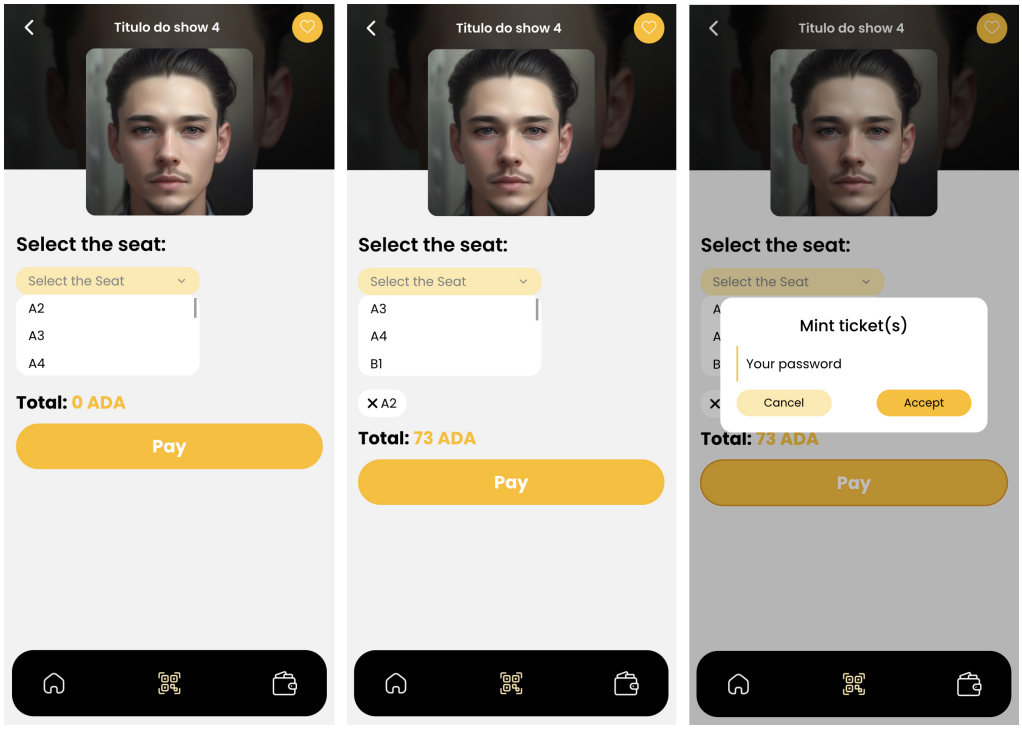


Figure 5.19: Buy directly to the artist

The app starts by gathering all the available seats from the server which subtracts all the reserved seats from the available ones. Both 'availableSeats' and 'filteredAvailableSeats', defined in lines 2 and 3 of Code 5.9, start with the same value because the variable 'availableSeats', although being a variable list, it maintains unchanged from the moment it is initiated, while the variable 'filteredAvailableSeats' is just filtering from the 'availableSeats' and the content from that list controls what appears in the UI.

```
1  // code from the frontend
2  const [availableSeats, setAvailableSeats] = useState([]);
3  const [filteredAvailableSeats, setFilteredAvailableSeats] = useState([])
       ;
4
5  const getAllSeats = () => {
6      AsyncStorage.getItem("auth_token").then((auth_token) => {
7          fetch(`${WEB3TICKETS_API_URL}seats/?auth=${auth_token}&place_uid
       =${showDetails.placeUid}&show_uid=${showDetails.id}`)
8              .then((data) => data.json())
9              .then((data) => {
10                 if (data["status"] === "ok") {
11                     setAvailableSeats(data["seats"])
12                     setFilteredAvailableSeats(data["seats"])
13                 }
14             })
15     });
16 }
```

Code 5.9: Function that gathers all the available seats

When the front end asks for the available seats, it needs to send the show's unique identifier, so the server can search in the database for the already reserved seats and filter out the ones that are not on that list. This is possible by applying the list comprehension of line 15 of Code 5.10.

```
1  # code from the server
2
3  def getAllReservedSeats(place, show):
4      tickets=Ticket.objects.filter(seat__place=place, show=show).all()
5      return [ t.seat.seat_no for t in tickets ]
6
7  ...
8
9  place=Place.objects.filter(uid=int(request.GET.get("place_uid"))).first
       ()
10 show=Shows.objects.filter(uid=int(request.GET.get("show_uid"))).first()
11
12 reservedSeats=getAllReservedSeats(place, show)
13 seats_list = Place_Seats.objects.filter(place=place).all()
14
15 return Response({"status":"ok", "seats":[ s.seat_no for s in seats_list
       if s.seat_no not in reservedSeats ]})
```

Code 5.10: Server side code to filter out the reserved seats

Code 5.11 controls the search, selection, and deselection of a seat. The variable 'selectedSeats' is used to control UI, presenting the seats that the user already selected and if the user clicks on the seat item in the list, the function 'deselectSeat' will be triggered, deselecting that seat.

```
const [selectedSeats, setSelectedSeats] = useState([]);
const [availableSeats, setAvailableSeats] = useState([]);
const [filteredAvailableSeats,setFilteredAvailableSeats]=useState([]);

const selectSeat = (seat) => {
    if (!selectedSeats.includes(seat)) {
        var tempSelectedSeats = [];

        selectedSeats.forEach((e) => tempSelectedSeats.push(e.
    toUpperCase()));

        tempSelectedSeats.push(seat.toUpperCase());

        setSelectedSeats(tempSelectedSeats);
        setFilteredAvailableSeats(availableSeats.filter(e => !
    tempSelectedSeats.includes(e.toUpperCase())))
    }
}

const deselectSeat = (s) => {
    var tempSelectedSeats = [];

    selectedSeats.forEach((e) => { if (e !== s) { tempSelectedSeats.push
    (e.toUpperCase()) } });

    setSelectedSeats(tempSelectedSeats);
    setFilteredAvailableSeats(availableSeats.filter(e => !
    tempSelectedSeats.includes(e.toUpperCase())))
}
```

Code 5.11: Code that controls the way that selected seats are presented in the app

Once the selection is made, which can contain more than one seat, and the price automatically updated, the user proceeds by clicking the "Pay" button. It will send a request to the server containing the authentication token, the seat numbers and the show uid. At this stage, the server generates the transaction associated with the ticket purchase and returns it to the user.

Code 5.12 is used first to check if the user has enough balance to proceed with the purchase. Once again the services of Blockfrost are used to gather the amount of 'lovelace' in the wallet of the user.

```
def checkUserHasBalance(show, user, ticket_amount):
    ticket_price = show.ticket_price * 1000000 * ticket_amount
    user_address = user.wallet_address
    wallet_details = BlockFrostApi(project_id=BLOCK_FROST_PROJECT_ID,
    base_url=ApiUrls.preprod.value).address(address=user_address)
    wallet_amount = wallet_details.amount
    amount_of_lovelace = [ namespace.quantity for namespace in
    wallet_amount if namespace.unit=="lovelace" ][0]
    return ticket_price < float(amount_of_lovelace)
```

Code 5.12: Server side code to check user's balance

To properly build the transaction, the server needs to start by loading all the needed addresses and keys, which are the show's policy keys, the artist's address (where the funds will go), and the user address (where the funds come from). For Code 5.13 to work, the module 'pycardano' needs to be installed and imported.

```
from pycardano import Address

# directory with the show keys
SHOW_DIRECTORY = "shows_pk/" + str(show.uid)

# artist wallet keys
artist_wallet_address=show.artist.wallet_address
artist_wallet = Address.from_primitive(artist_wallet_address)

# user wallet key
user_wallet = Address.from_primitive(user_wallet_address)
```

Code 5.13: Server side code to load the required keys

Then the server needs to calculate the policy and for that, the previously stated expiration block is used. The method 'getPolicy', in the line 5 of Code 5.14, returns the selected show's policy by loading the files created upon the show creation, and then the server can obtain the policy id, which is the serialized identifier of the policy.

```
from pycardano import InvalidHereAfter

# set policy
expiration_slot=getExpirationBlock(SHOW_DIRECTORY)
policy=getPolicy(SHOW_DIRECTORY, expiration_slot)
must_before_slot = InvalidHereAfter(expiration_slot)
policy_id = policy.hash()
```

Code 5.14: Server side code to load the policy

The next step is to define the blockchain asset that will be minted and associate it with the policy id. The variable 'my_asset' is used to store the number of tokens that will be minted for each ticket. Since the NFTs are "non-fungible" the number of tickets for each seat will always be one. The name given to the asset is 'TICKET_{show_uid}_{seat_no}' because this way the asset name identifies the show and the seat. The line 13 of Code 5.15 is used to associate the policy with the transaction.

```
from pycardano import Asset, MultiAsset

# define the nft
my_asset = Asset()

for seat in seat_list:
    nft_name="TICKET_"+str(show.uid)+"_"+seat
    nft1 = AssetName(nft_name.encode("utf8"))
    my_asset[nft1] = 1

my_nft = MultiAsset()
my_nft[policy_id] = my_asset
native_scripts = [policy]
```

Code 5.15: Server side code to define the NFTs being minted

Since the nft needs to carry some data (seat and show_uid) the auxiliary data also needs to be defined. The method 'getAuxiliaryData', in the line 3 for Code 5.16, iterate over a list of seats and puts all the ticket assets under the same policy id to be minted all at the same time. The returned value will be an object whose first key in the chain is "721" because of the standard "CIP 25 - Media NFT Metadata Standard"[6].

```
1  from pycardano import AlonzoMetadata, AuxiliaryData, Metadata
2
3  def getAuxiliaryData(policy, show, seat_list):
4      policy_id = policy.hash()
5      policy_hex = policy_id.payload.hex()
6      metadata={721:{policy_hex:{}}}
7
8      for seat in seat_list:
9          nft_name="TICKET_"+str(show.uid)+"_"+seat
10         metadata[721][policy_hex][nft_name]={
11             "name": "Ticket for show "+show.name,
12             "seat":seat,
13             "show_uid":str(show.uid),
14             "image": show.image_hash,
15         }
16     auxiliary_data = AuxiliaryData(AlonzoMetadata(metadata=Metadata(
       metadata)))
17
18     return auxiliary_data
19
20 ...
21
22 auxiliary_data = getAuxiliaryData(policy, show, seat_list)
```

Code 5.16: Server side function to generate the metadata for the ticket

Before beginning with the distribution of money and tokens by the artist and the fan, respectively, there is the need to instantiate the 'TransactionBuilder', present in the line 4 of Code 5.17, that will be helpful with the calculations of inputs and outputs of the transaction. In Code 5.17, the variable 'user_wallet' is appended to the list of inputs because the money will come from this wallet. The time to live is set to the date of the show because no ticket from this show can be minted after the show occurs. To define that the transaction is a minting transaction, the 'MultiAsset' object is appended to the transaction. The parameter 'native_scripts' are also required because every minting transaction needs the policy to validate it. As explained before, the auxiliary data is the data that will be carried with the transaction.

```
1  from pycardano import TransactionBuilder
2
3  # build the transaction
4  builder = TransactionBuilder(chain_context)
5
6  builder.add_input_address(user_wallet)
7  builder.ttl = must_before_slot.after
8  builder.mint = my_nft
9  builder.native_scripts = native_scripts
10 builder.auxiliary_data = auxiliary_data
```

Code 5.17: Server side code to construct the transaction

[6] https://cips.cardano.org/cips/cip25/

In Code 5.18, the final part of the code, it is shown the division of outputs: the user's wallet will receive the ticket(s) and the artist will receive a multiple of the price of the ticket, depending on the number of purchased tickets. For this to be properly done, the server needs to calculate the minimum value that the user will have to pay for the minting, i.e., the bigger the metadata higher is the amount of 'lovelace' that needs to be carried with the asset(s). To make sure that the fan doesn't need to pay more than the price stipulated by the artist, to the price of the ticket, it is subtracted all the fees (the amount of 'lovelace' that is carried with the asset and the network transaction fee).

```
from pycardano import TransactionOutput, min_lovelace, Value

min_val = min_lovelace(
    chain_context, output=TransactionOutput(user_wallet, Value(0, my_nft
    ))
)

# set nft price
price=int(show.ticket_price*1000000*len(seat_list))

# set one of the outputs (nft mint)
builder.add_output(TransactionOutput(user_wallet, Value(min_val, my_nft)
    ))

# calculate the fees
fee=builder._estimate_fee()

# subtract the fee to the amount of money that the ticket creator will
    receive
builder.add_output(TransactionOutput(artist_wallet, Value(price-min_val-
    fee)))

tx_body = builder.build(change_address=user_wallet)
```

Code 5.18: Server side code to set all the outputs of the transaction

Finally, what's returned to the user is a CBOR (Concise Binary Object Representation) string, just after the new order is added to the database.

The user's wallet's signing key is then used to sign the transaction, ensuring the authenticity of the payment process. Because of one of the limitations, that will be explained further, the seed phrase needs to be sent to the server to sign the transaction with the corresponding signing key.

The last part of the purchase is the submission of the transaction, which requires an important step: the signing of the transaction by the user and the policy. So, for the previously created transaction, the server needs to take the mnemonic from the user and the policy and create a list of witnesses.

Used in the method 'witnessWithPolicy', the other method called 'loadKeys' convert the files 'policy.skey' and 'policy.vkey' into two objects with the sign feature to verify the transaction.

Since the user sends the seed phrase, it needs to be converted into a pair of signing and verification keys, and for that there is the method called 'getKeysFromMnemonic', which takes the mnemonic phrase and converts it into an 'HDWallet' object (Hierarchical Deterministic Wallet). This can easily derive the keys.

```
def witnessWithPolicy(data, dir):
    collection_ticket_skey, collection_ticket_vkey = loadKeys(dir, "
    policy")
    signature = collection_ticket_skey.sign(data)
    return VerificationKeyWitness(collection_ticket_skey.
    to_verification_key(), signature)

def getKeysFromMnemonic(mnemonic_words):
    hdwallet = HDWallet.from_mnemonic(mnemonic_words)
    accountKey=hdwallet.derive(1852, hardened=True).derive(1815,
    hardened=True).derive(0, hardened=True)

    s_key = ExtendedSigningKey.from_hdwallet(accountKey)
    v_key = PaymentVerificationKey.from_signing_key(s_key)

    return s_key, v_key

...

witnessSet = TransactionWitnessSet()
witnessSet.vkey_witnesses=[user_witness, witnessWithPolicy(tx_body.hash
    (), "shows_pk/"+str(show.uid))]
witnessSet.native_scripts = [policy]

tx.transaction_witness_set=witnessSet
```

Code 5.19: Server side code to create the list of witnesses

To end the workflow, Code 5.20 shows the simple code used to submit the transaction to the blockchain (using Blockfrost services) and to update the order in the database, setting the new 'transaction_hash' with the submitted transaction's id.

```
from blockfrost import ApiUrls, BlockFrostApi

# submit transaction
context = BlockFrostChainContext( project_id=BLOCK_FROST_PROJECT_ID,
    base_url=ApiUrls.preprod.value )
tx_id = tx.transaction_body.hash().hex()

context.submit_tx(tx.to_cbor())

# update order's transaction id
order.transaction_hash = tx_id
order.save()
```

Code 5.20: Server side code to submit the transaction to the blockchain

5.4.8 Get all the wallet tickets

The wallet tickets are shown on the Wallet page to normal users. This is a process that involves fetching the server first and then fetching the blockchain through Blockfrost. The app needs to fetch the server, so it can filter the assets that belong to a show created in the app, i.e. since Web3Ticket operates within the blockchain, all the assets belonging to the user's wallet would be shown in the Wallet Page, but not every asset will be a ticket, because the user can use his/her wallet outside Web3Ticket.

What the server does is fetch the blockchain requesting for user's assets and then searches in the directory 'shows_pk' for all the policy ids to compare with the assets in the user's wallet.

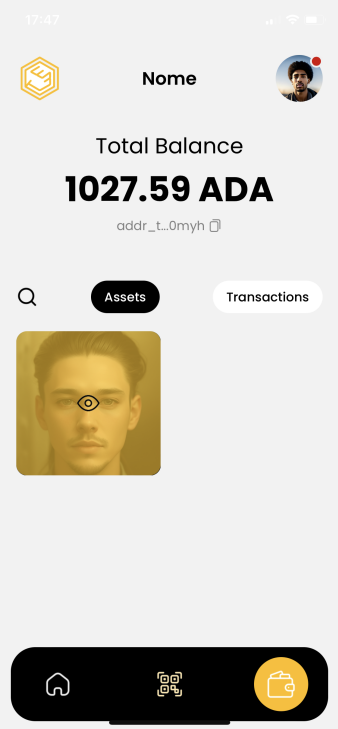


Figure 5.20: Wallet screen showing the balance, address and assets

In Code 5.21, the front end gets the address from the device's storage and fetches the server for the wallet's assets. There are two unrelated methods called 'setLookingForAssets' and 'setRefreshing' used for UI control, which objective is to show and hide the spinning circles characteristic of loading content. After receiving the list of objects with the 'asset name' and the 'show uid', the next method comes to recursively get the details of each ticket.

```
const getWalletTickets = () => {
    AsyncStorage.getItem("address").then((address)=>{
        if(address===null)
            navigation.navigate('BeforeLogin');

        fetch(`${WEB3TICKETS_API_URL}wallet_assets/?address=${address}`)
        .then((assets_response)=>assets_response.json())
        .then((assets_response)=>{
            if(assets_response["status"]==="ok"){
                getAssetDetails(0, assets_response["tickets"]);
            }else{
                setLookingForAssets(false)
                setRefreshing(false)
            }
        })
    })
}
```

Code 5.21: Function to get all the user's tickets

The method 'getAssetDetails', in line 1 of Code 5.22, is useful to iterate over many Promises. For each ticket, the front end does a fetch to Blockfrost API to get its details. When a certain fetch is completed, the function iterates over the next ticket. At the beginning of the function, there is checked if all the tickets have been gathered and then they are shown on the page.

```
1  const getAssetDetails = (rootIndex, assetList, tempAssets=[]) => {
2      if(rootIndex>=assetList.length){
3          setAssets(tempAssets);
4          setLookingForAssets(false);
5          setRefreshing(false)
6          return 0
7      }
8
9      fetch(BLOCKFROST_API_URL+"/assets/"+assetList[rootIndex].assetname,
10         {headers: {"project_id":BLOCKFROST_PROJECT_ID}}
11     )
12     .then((blockfrost_response)=>blockfrost_response.json())
13     .then((blockfrost_response)=>{
14         if(!blockfrost_response["error"]){
15             tempAssets.push( {
16                 "show_uid":assetList[rootIndex].show_uid,
17                 "image":"https://ipfs.io/ipfs/" + blockfrost_response["
    onchain_metadata"]["image"].replace("ipfs://", ""),
18                 "assetname":assetList[rootIndex].assetname
19             } )
20             getAssetDetails(rootIndex+1, assetList, tempAssets)
21         }else{
22             setLookingForAssets(false)
23             setRefreshing(false)
24         }
25     })
26     .catch(()=>{setLookingForAssets(false); setRefreshing(false);})
27 }
```

Code 5.22: Recursive function to gather all the tickets' details

5.4.9 Sell Ticket - Add to Marketplace

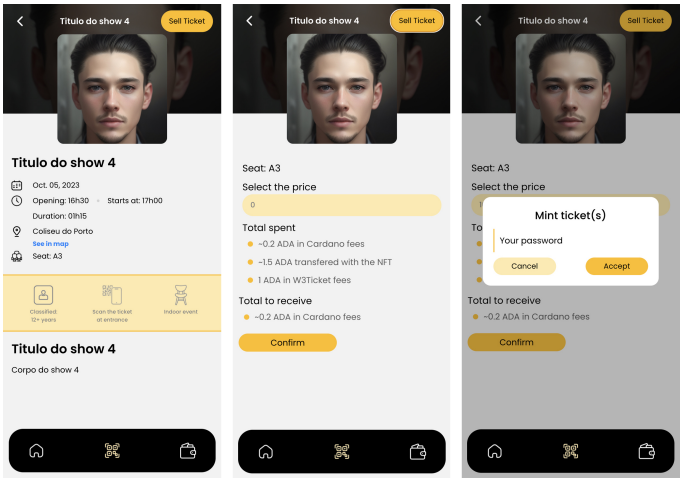


Figure 5.21: Sell an asset from wallet

The background of the process of selling a ticket is very similar to the minting (buying directly from the artist). After building the transaction the user also needs to enter the password for decrypting the seed phrase for later the server signing the transaction. The main points of change in this workflow are the creation of a dedicated wallet and the minting of a new nft. The dedicated wallet serves as a trusted intermediary for where the asset being sold goes until someone buys it.

The method 'createSellWallets', in line 1 of Code 5.23, generates a unique wallet name and creates a corresponding set of keys (public and private keys) for the wallet. The function ensures that the generated wallet name is unique by appending a random number to the base name "storage_" until a unique name is obtained. The chosen name is then returned to be stored in the '_info' nft that will be minted further.

```
1  def createSellWallets():
2      randomNumber = random.randint(0, 99999)
3      wallet_name="storage_"+str(randomNumber)
4      walletCreation = createKeys(wallet_name, "selling_pks")
5
6      while not walletCreation:
7          randomNumber = random.randint(0, 99999)
8          wallet_name="storage_"+str(randomNumber)
9          walletCreation = createKeys(wallet_name, "selling_pks")
10
11     return wallet_name
```

Code 5.23: Server side code to generate the selling wallet keys

```
1  def getInfoNftAuxiliaryData(info_nft_policy, nft_selling_asset_name,
       nft_selling_policy_id, nft_selling_price, seller_address,
       storage_wallet_name):
2      policy_id = info_nft_policy.hash()
3      nft_name=bytes.fromhex(nft_selling_asset_name).decode('utf-8')
4
5      metadata = {
6          721: {
7              policy_id.payload.hex(): {
8                  nft_name: {
9                      "asset_name": split_str_by_num(
       nft_selling_asset_name, 64),
10                     "policy_id": split_str_by_num(nft_selling_policy_id,
        64),
11                     "price": nft_selling_price,
12                     "seller_address":split_str_by_num(seller_address,
       64),
13                     "storage_wallet_name":storage_wallet_name,
14                     "image": "
       QmRhTTbUrPYEw3mJGGhQqQST9k86v1DPBiTTWJGKDJsVFw"
15                 },
16             }
17         }
18     }
19
20     auxiliary_data = AuxiliaryData(AlonzoMetadata(metadata=Metadata(
       metadata)))
21
22     return auxiliary_data
```

Code 5.24: Server side code to generate the metadata for the '_info' nfts

Code 5.24 corresponds to the function 'getInfoNftAuxiliaryData' that illustrates the way that the '_info' nft is built before being minted. The generated auxiliary data includes metadata related to the asset, such as its name, policy ID, price, seller's address, storage wallet name, and image associated with the asset.

5.4.10 Buy Ticket from Marketplace

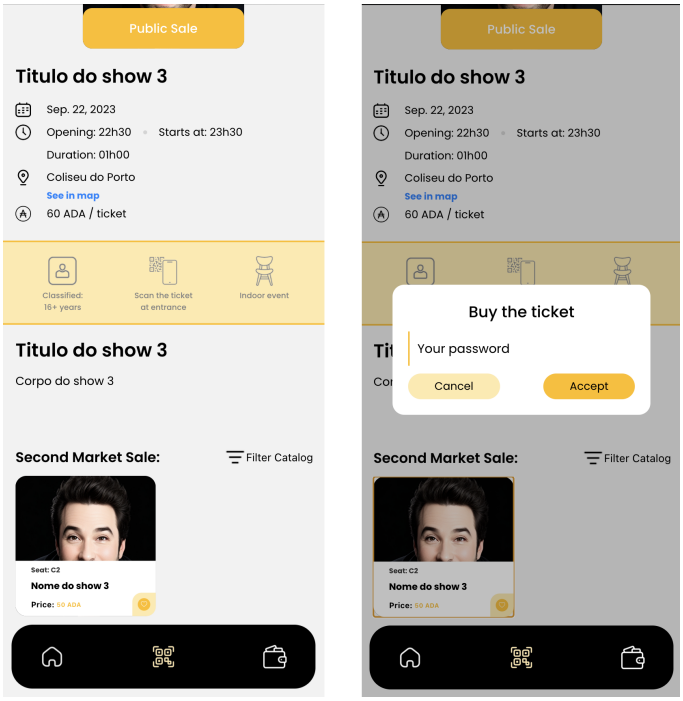


Figure 5.22: Buy an asset from marketplace

Just like the previous methods (Mining and Selling) the feature of buying from the marketplace is very similar, insofar as the transaction building function is essentially the same but changing the inputs and outputs. The major different in this functionality is that the '_info' nft needs to be burned, i.e., permanently removed from the blockchain (doesn't prohibit it from existing again in the next sale).

```
# define the nft being burned; needs to be defined two times:
# one of them, with negative value to burn
# the other with null (0) value to serve as output
info_asset_output = Asset()
info_asset_burn = Asset()
info_nft_name=bytes.fromhex(asset_name_hex).decode("utf-8")
nft1 = AssetName(bytes.fromhex(asset_name_hex))

info_asset_output[nft1] = 0
info_asset_burn[nft1] = -1
info_nft_output = MultiAsset()
info_nft_burn = MultiAsset()
info_nft_output[policy_id] = info_asset_output
info_nft_burn[policy_id] = info_asset_burn

native_scripts = [policy]
```

Code 5.25: Server side code to define the objects that represent the burn of the nft in the blockchain

The major change is represented in Code 5.25. Instead of defining one 'MultiAsset' object, there are needed two of them, one serving as output for the buyer and another serving as minting specification for the blockchain. This makes more sense if considered that the user will receive 0 tokens of the '_info' nft, so the object 'info_asset_output' shall carry zero units of the burning token. And when talking about the blockchain, the logic is similar, the existing nft needs to have 0 tokens of its type in circulation, but there is one. This value needs to be subtracted from the blockchain by indicating that the mint will add '-1' tokens of that token.

5.4.11 See ticket's qr code

The second image from Figure 5.23 shows the step before a ticket becomes green because, by default, all the tickets have a yellow overlay. This overlay indicates whether the ticket is ready to use or not. Shortly, after the user presses the button "Use Ticket", the ticket stops existing (is burned), but as is it registered in the blockchain that the wallet that burned the token is that user's wallet, it is forever associated with the user. The green overlay on top of the ticket means that the ticket has been successfully burned and that the Place Responsible will be able to validate it.

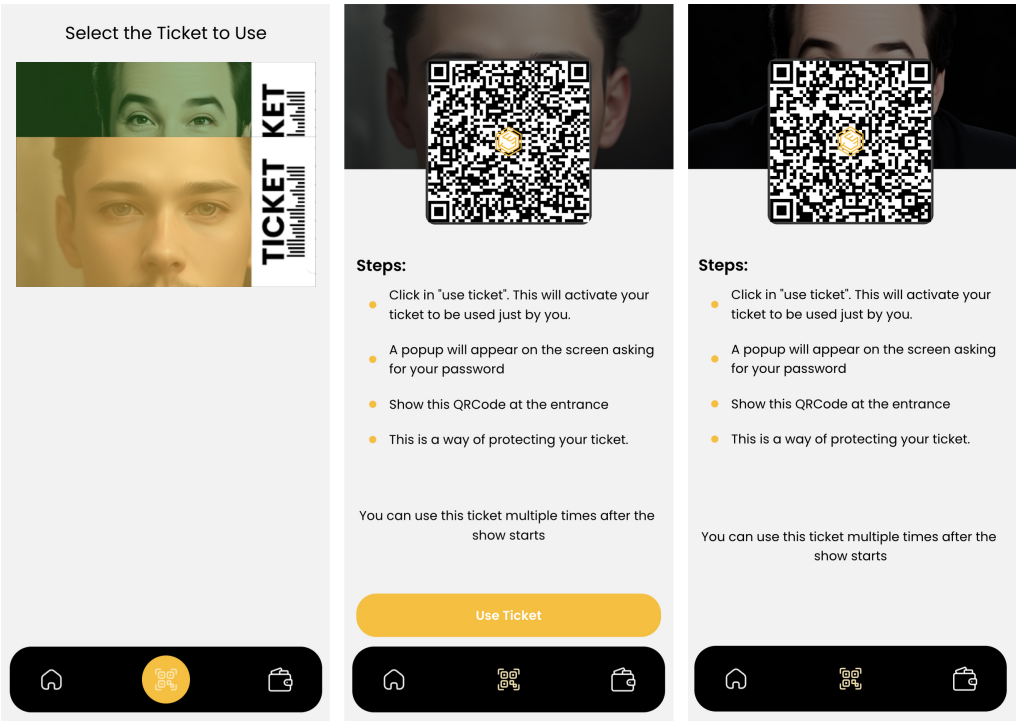


Figure 5.23: Generate a QR Code to be validated

The big advantage of burning the ticket is that the ticket cannot be sold after being burned and the ticket cannot be validated without burning it, so it is not possible to try to sell a ticket after using it to enter into the place.

Code 5.26: utilizes the 'react-native-qrcode-svg' library to generate the QR code based on the provided content, which is composed by the 'asset name' and the authentication token. The component is designed to be animated using the Animated API in React Native, allowing it to increase and decrease in size while maintaining a smooth transition.

Note: The secure issue behind this validation process will be explored further.

```
import QRCode from 'react-native-qrcode-svg';
import { Animated } from 'react-native';

...

var content = JSON.stringify({"assetname":assetname, "auth_token":
    userAuthToken})

...

<Animated.View>
    <TouchableOpacity onPress={openQRCode} style={styles.headerFoto}>
        <QRCode value={content} logo={logo} size={(widthAnim._value -
    6>0) ? widthAnim._value - 6 : windowWidth/2 -6 } color="#000" />
    </TouchableOpacity>
</Animated.View>
```

Code 5.26: Code that allows showing the qrcode to the user

5.4.12 Scan & Validate user's ticket

The feature of scanning and validating the user's ticket is exclusive to Place Responsible entities. On the front end side, it's used the library 'expo-barcode-scanner' for React Native and then on the server side, all the server needs to do is bring the user address and the asset name and search in the blockchain (through Blockfrost) for all the asset's transaction history and try to find a record which key 'action' is equal to 'burned'. This ensures that the ticket was burned but not that the user is the owner. The second fetch to the database is to find the 'inputs' of that transaction. If the user's wallet address and the asset name are present in the transaction inputs (in the same transaction input), then the ownership is proved, as shown in lines 11 to 15 of Code 5.27:

```
# seach for the burning transaction in this asset's history
asset_history = BlockFrostApi(project_id=BLOCK_FROST_PROJECT_ID,
    base_url=ApiUrls.preprod.value).asset_history(asset=asset_name)
burned_transaction=[ x for x in asset_history if x.action=="burned"]

...

# search the user's wallet address in thr burning transaction inputs
burn_tx = burned_transaction[0].tx_hash
transaction_utxos = BlockFrostApi(project_id=BLOCK_FROST_PROJECT_ID,
    base_url=ApiUrls.preprod.value).transaction_utxos(hash=burn_tx)

isValid=False
for i in transaction_utxos.inputs:
    # check if the address that made the transaction is the same as the
    user's and if the asset was added to the inputs of that address (user
    's)
    if i.address==user_wallet and any([ a.unit==asset_name for a in i.
    amount ]):
        isValid=True
```

Code 5.27: Server side code to check if the ticket was burned and if the owner is the user

Its important to note what are the returning values from these fetches explained in Code 5.27:

```
/assets/{asset}/history
[
...
    {
    "tx_hash": "1
    a0570af966fb355a7160e4f82d5a80b8681b7955f5d44bec0dde628516157f0",
    "amount": "1",
    "action": "burned"
    }
]

/txs/{hash}/utxos
{
    "hash": "1e0...477",
    "inputs": [
        {
            "address": "addr1q9...wv",
            "amount": [],
            "tx_hash": "1a0...7f0",
            "output_index": 0,
            "data_hash": "9e4...710",
            "inline_datum": "19a6aa",
            "reference_script_hash": "13a...ad1",
            "collateral": false,
            "reference": false
        }
    ],
    "outputs": [
        {}
    ]
}
```

Code 5.28: Representation of the returning value of Blockfrost endpoint

5.4.13 Logout

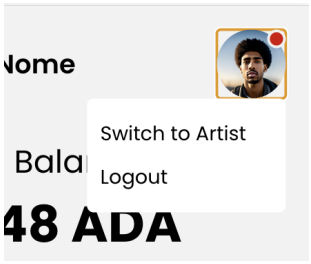


Figure 5.24: Logout button

The logout function is a callback function that handles the logout feature in a React Native application. It removes specific items from the device's storage, which is handled by AsyncStorage to persist data in the application. Once the required items are successfully removed, the function navigates the user to the first screen where the user can choose between recovering or creating a new wallet.

5.5 Limitations

While the Web3Ticket application and its associated server offer many advantages, it is important to note certain limitations intrinsic to the system.

Firstly, using the InterPlanetary File System (IPFS) introduces a delay in making show cover images globally accessible. Due to the decentralized nature of IPFS, the image files must circulate through computer nodes across the globe, resulting in a time lag before they are readily available to users worldwide. This delay may slightly impact the user experience related to image loading times.

Secondly, the blockchain, which serves as the underlying technology for transaction validation, can cause a delay at the moment that the user transactions are processed. In cases where multiple buyers attempt to purchase the same ticket simultaneously, both transactions can be submitted. However, due to the sequential nature of blockchain validation, only one buyer will ultimately receive the ticket (the one whose transaction is validated first). This limitation is easily mitigated by providing an abstraction layer in the database and the server, so the user can receive a notification if the transaction has its validation failed in the blockchain. For example, the server can listen for the blockchain status of a specific transaction and communicate to the user if the transaction has failed.

It is worth noting that React Native and Expo, the frameworks utilized in the development of the app, have compatibility challenges with the '@emurgo/cardano-serialization-lib-nodejs' library. The file system module required by certain methods in the library is not fully supported by React Native and Expo. This limitation doesn't allow the app to sign the transaction after receiving it, and for that, the solution for the present work was to send the seed phrase to the server to sign the transaction with it. Another approach could be to use raw modules/methods that support bip32[7] type of encryption, so the app could sign the transaction itself and prevent security issues.

Still talking about security, there is a security problem when talking about the ticket validation and it is related to shoulder surfing. When a fan has his/her ticket's QRcode opened and ready to use, a malicious user can take a picture of that QRcode and use it as his/her own, and even take the user's authentication token because it is present in the QRcode. There are two solutions for this: one of them involves notifications and the other is message queues.

The one with the notifications consists of the Place Responsible for scanning the QRcode (containing only the asset name) and the server checks for the user who owns that ticket and sends him/her a notification asking for permission for validation which the user can accept or refuse. The solution that involves a message queue consists in starting a communication channel between the user and the server at the moment that the user opens the QR code in the app. Then, when the Place Responsible tries to validate the ticket, the user is automatically requested to accept/refuse. Both approaches can include a message being signed by the user to prove his/her authorization.

By recognizing these limitations, the Web3Ticket application and server may undergo changes and seek potential improvements in future iterations.

[7] https://github.com/bitcoin/bips

Chapter 6

Experimentation and Evaluation

This chapter includes a detailed description of the experiments conducted to validate the research hypothesis enunciated in Chapter 1. To evaluate the application, a combination of Quantitative Evaluation Framework (QEF) and usability tests[1] was employed. The QEF provided a structured approach to measure various aspects of the application's performance, efficiency, and reliability. It involved collecting and analyzing data related to metrics such as functionality, adaptability, and usability.

It will start with the description of the use cases because, without the general vision about them, it wasn't possible to extract all the functional requirements to proceed with the evaluation methodology.

6.1 Use Cases

Business Use Cases, intuitive and self-explanatory, are a powerful tool to gather requirements in the analysis of user requirements. After creating the use-case list, the list of goals is also determined. They are different from the System Use Cases which are more complex and dedicated to describing the architecture of the system [43].

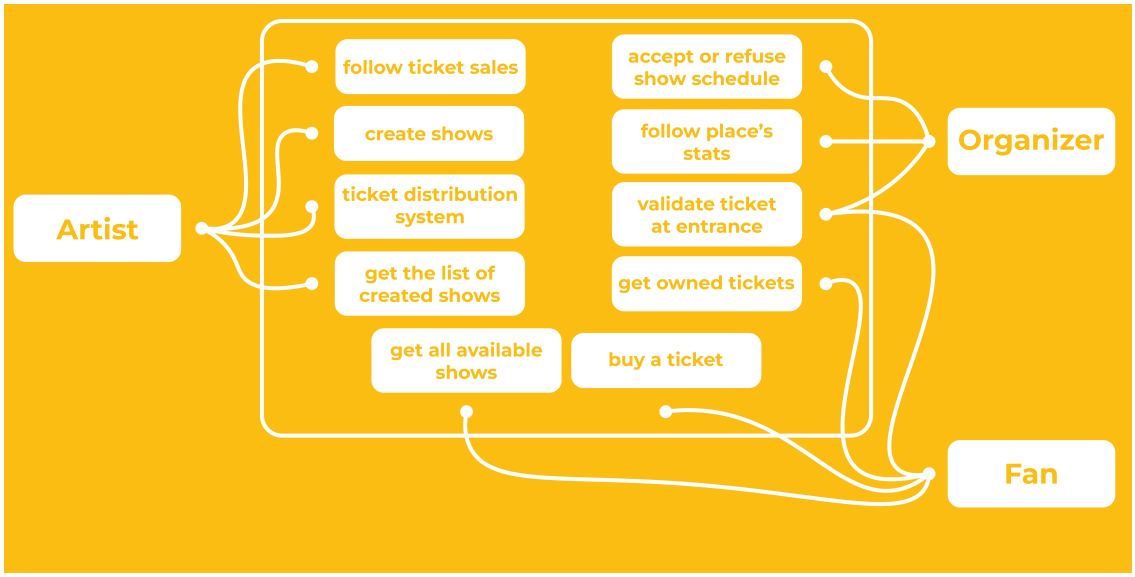


Figure 6.1: Business Use Cases Diagram

[1]The usability tests were collected using Google Forms through this link: https://forms.gle/yD7QwuDe8CN3eFs47

Figure 6.1 shows the main actors and the way they are going to use the platform. In a simplified way, the artist is responsible for scheduling shows, the fan can buy and sell the tickets, and the place responsible can accept or reject the requests from the artist to perform there.

6.1.1 Artist

As explained before, the artist is the engine of the platform because they are the entity that will bring the interest of customers.

Create shows

The artist can create shows, where he/she can choose the date, the starting time and duration, the location of the show (the place where it will happen), the minimum age to watch the show, if it is indoor or outdoor, the title, description, and price. At this time, the show isn't scheduled yet, because the selected showroom didn't accept yet.

Follow ticket sales

The app has an area dedicated to statistics. There the artist can follow the sales grouped by day, week, or month. The same for the assiduity of the public and the number of super fans (the ones that buy a ticket for more than one show).

Ticket distribution system

Although the platform automatically distributes the show through the fans without the artist having to do anything, he/she can increase the visibility by buying promotion packs to put the show on top of the searches for 7 days, or highlighting the show among the others.

Get the list of created shows

The artist has two ways of doing so: by looking at the list of created shows or by navigating through his/her calendar. In the list of created shows the artist has the possibility of editing the show properties, including its cover picture. In the calendar, the user can consult the dates of the next shows, but only the ones that were accepted by the place responsible.

6.1.2 Fan

The fan is the common user whose task is to buy the tickets and eventually sell them. In Web3 Ticket the proposal is to exist a marketplace where this kind of user can freely put a ticket up for sale at any price.

Get all available shows

Since the fan uses the app for getting the ticket information, first it is shown to him/her all the artists with shows available to buy, and all the shows ordered by user's preference. This preference is determined each time the user clicks on the 'like' button.

Buy a ticket

The action of buying a ticker has two different options. Since there is a limited supply of tickets for each show, there is a selling phase where the fan is buying directly from the artist. Then, the moment the fans start putting tickets up for sale, there will be a list of tickets that other users can reach out to buy, however, in this phase, the ticket isn't being bought from the artist, but from another fan.

Get owned tickets

The tickets that the fan owns are in his/her wallet. When requesting to see the details of a ticket, a qrcode will appear. This is important because with this qrcode the event's place can validate the ticket. It will be asked for a password, that unlocks the private key, and a message is signed to prove the authenticity.

Validate a ticket at the entrance

When the fan reaches the show's place, he/she needs to validate the ticket. The fan needs to navigate through his/her own tickets, and press the ticket to show the qrcode at the entrance. After the scan, it will be asked for a password, and the ticket will be validated.

6.1.3 Organizer

The place responsible is the person or people responsible for the showroom or enclosure. This person is responsible for describing the place, in terms of name, address, number of seats, and disponibility.

Validate a ticket at the entrance

Intimately related to the fan, when he/she needs to validate his/her ticket at the entrance. The person at the entrance just needs to point the phone to the fan's qrcode this will check its validity.

Follow the place's stats

As the artists can get statistics about their shows, here are shown the number of validated tickets daily, weekly, and monthly, as well as the number of shows performed in that place.

Accept/refuse shows for a certain day

Intimately related to artists, since this is the task that needs to be done to accept the show to perform on a certain date in that place. The value needs to be agreed upon between the two parts.

6.2 Methodology and Evaluation

To evaluate the progress and effectiveness of this project will be used a combination of quantitative and qualitative metrics to measure user satisfaction, i.e. how satisfied users are with the product and the user experience by conducting usability tests and analyzing their behavior to understand how users are interacting with the product.

Primarily, setting goals and objectives involves defining the desired outcomes of the system. Setting objectives also help to group the requirements by the system actors which are the artist, the fan, and the organizer (also referred as Place Responsible). Table 6.1 matches the actor to the desired functionality.

For the extraction of the requirements, the strategy used was mainly prototyping. However, informally, at an event organized by a friend, some questions were asked, as quoted below, to know the feeling about the current market.

- What are the current solutions for an artist to promote his/her show?
- Have you ever tried a service that allows you to distribute the show?
- How much time do you spend trying to sell tickets online?
- Can you tell me about a time when you thought you had found a solution to ticket selling, but it didn't work out?

The answers to those questions are cited below:

- "Conventional ways like social media and direct invitation"
- "The service used isn't specific for ticket selling, it was my Facebook and Instagram pages"
- "The process is painful because it needs to be sold individually and smart artists don't have a way of selling the tickets without direct interaction"
- "Precisely the social networks, but I had to deal with all the sales individually"

Although this isn't enough to conclude the whole market and its artists, it gives an idea about another problem inherent to the market which is the difficulty that unlabeled artists might have when trying to sell tickets for their show.

Then the tool used for prototyping the app was Figma, which is a web-based design and prototyping tool. Prototyping showed to be a good way of getting the starting point, the identity, and the way of creating the abstraction between the ticketing systems and the blockchain. A total of 49 mobile screens were designed to deeply understand where and how the user's problems should be fixed. Many of those screens weren't shown throughout the report because the focus of the implementation was on the interaction with the blockchain.

With the gathering of requirements done, it's now possible to use the Quantitative Evaluation Framework as the tool to measure the development of the product by assigning numerical values to the requirements related to their importance as well as their completeness level. The gathering of requirements also helped to identify the target audience which will be useful to create the surveys to distribute among artists, normal users (fans), and organizers. Further in this document, it will be given the requirements their importance, and the questions placed in the survey will be analyzed as well as the target's responses to put into practice the QEF.

Table 6.1: Functional requirements table: matching the actor to its desire

Actor	**Functional Requirement**
All	Create a native wallet
All	Recover a native wallet
Artist	Stats about selling tickets
Artist	Consult Balance
Artist	Consult Last Sales + all sales
Artist	Consult notifications and delete it
Artist	Create shows + distribute
Artist	Consult weekly Timeline
Artist	Change settings
Artist	Update Profile
Artist	Profits withdraw
Artist	Consult created shows
Artist	Consult show details
Artist	redirect to map
Artist	Edit show details
Artist	Search for shows in the gallery
Artist	Promote the show
Artist	Consult Calendar & get show details
Artist	Logout
Fan	See all the show catalog
Fan	See show details
Fan	Buy the ticket (primary and second market)
Fan	Show the ticket's QrCode to be validated
Fan	See wallet address
Fan	Copy wallet address
Fan	See all the tickets in the wallet
Fan	Put tickets at sale
Fan	See notifications
Fan	Change display name
Fan	Change profile picture
Fan	Withdraw all the money in the wallet
Fan	Filter nfts by name (search)
Fan	Delete Account
Fan	Automatically show the tickets for the upcoming shows (the shows that are opening)
Fan	Logout
Organizer	See the last requests for scheduling
Organizer	See all statistics
Organizer	Accept & Refuse Requests for scheduling
Organizer	See Show details
Organizer	See the artist's details
Organizer	Set the number of seats, the seats interdicted, and all available seats
Organizer	Add available seats in bulk
Organizer	Scan & Validate the user's ticket
Organizer	See all scheduled shows
Organizer	Delete Account
Organizer	Logout
Organizer	Withdraw all the earned money

The items listed in Table 6.2 are the core components that need to be implemented to fulfill what was defined in the hypothesis: a marketplace where the artist creates the show, the showroom responsible approves/refuses the show, and where the fans (normal users) can buy and sell tickets without the need of resorting to any other platform or entity.

Table 6.2: Functional requirements table: core components matching hypothesis

Actor	**Functional Requirement**
All	Create a native wallet
All	Recover a native wallet
Artist	Consult Balance
Artist	Create shows + distribute
Artist	Consult created shows
Artist	Consult show details
Artist	redirect to map
Artist	Edit show details
Artist	Logout
Fan	See all the show catalog
Fan	See show details
Fan	Buy the ticket (primary and second market)
Fan	Show the ticket's QrCode to be validated
Fan	See wallet address
Fan	Copy wallet address
Fan	See all the tickets in the wallet
Fan	Put tickets at sale
Fan	Logout
Organizer	See the last requests for scheduling
Organizer	Accept & Refuse Requests for scheduling
Organizer	See Show details
Organizer	Scan & Validate the user's ticket
Organizer	Logout

6.2.1 Quantitative Evaluation Framework

The QEF provides a systematic approach to evaluating and assigning weights to different requirements based on their relative importance. By assigning a "requirement weight" to each requirement, the QEF allows for a more objective development of the application's features. This chapter digs into the process of using the QEF and explores how requirement weights were determined to guide the development. The QEF proved to be an important tool in setting the impact of each requirement, enabling a more efficient allocation of the work force to meet the critical needs of the application, in order of completing the hypothesis presented before.

The following table lists the requirements as well as its weight and completeness level so that later the questions can be build to be delivered among the users.

Table 6.3: QEF table: core functional requirements matching weight and completeness level

Requirement	Weight	Completeness (%)
FA01 - (Artist) Create a native wallet	10	100
FA02 - (Artist) Recover a native wallet	10	100
FA03 - (Artist) Consult Balance	8	100
FA04 - (Artist) Create shows and distribute them among users	10	50
FA05 - (Artist) Consult created shows	8	90
FA06 - (Artist) Consult show details	8	100
FA07 - (Artist) Redirect to map to see the event's location	4	100
FA08 - (Artist) Edit show details	8	0
FA09 - (Artist) Search for shows in the gallery	7	80
FA10 - (Artist) Logout	10	100
FF01 - (Fan) Create a native wallet	10	100
FF02 - (Fan) Create a native wallet	10	100
FF03 - (Fan) See all the show catalog	8	100
FF04 - (Fan) See show details	8	100
FF05 - (Fan) Buy the ticket (primary and second market)	10	100
FF06 - (Fan) Show the ticket's QrCode to be validated	8	100
FF07 - (Fan) Put tickets at sale	10	100
FF08 - (Fan) See all the tickets in the wallet	10	100
FF09 - (Fan) See wallet address	8	100
FF10 - (Fan) Copy wallet address	8	100
FF11 - (Fan) Logout	4	100
FPO01 - (Place's) Create a native wallet	10	100
FPO02 - (Place's) Create a native wallet	10	100
FPO03 - (Place's) See the last requests for scheduling	10	100
FPO04 - (Place's) Accept & Refuse Requests for scheduling	10	100
FPO05 - (Place's) See Show details	10	100
FPO06 - (Place's) Scan & Validate the user's ticket	10	50
FPO07 - (Place's) Logout	10	0

Table 6.4: QEF table: core adaptability requirements matching weight and completeness level

Requirement	Weight	Completeness (%)
AC01 - The user controls his/her crypto assets	10	50
AC02 - The use of well known currencies helps to improve the usability of people not used to crypto	10	100
AM01 - Application presents a set of interfaces and endpoints, adding the possibity of intoducing new features	5	100
AM02 - The server must be scalable	8	100
AV01 - Application follows online store guidelines	5	50
AV02 - Application adjusts to screen resolution	5	80
AV03 - Application is multiplataform	5	100
UM01 - The user interface is simple and fast, without slowing down when the number of users is high	5	50

Table 6.5: QEF table: core usability requirements matching weight and completeness level

Requirement	Weight	Completeness (%)
UM02 - (Artist) Easy to create a show	5	100
UM03 - (Artist) Easy to distribute	5	0
UM04 - (Fan) Easy to buy a ticket either from artist or on second market	5	100
UM05 - (Fan) Easy to sell a ticket	5	100
UM06 - Application runtime does not have errors, and unexpected errors are handled and shown to the user	5	50
UM07 - User have access to a main menu	10	100
UV02 - (Place's) Easy to understand where to accept or refuse shows	10	100
UCQ01 - All the messages are easy to understand	5	50
UCQ02 - Application is clear to people not used to use crypto services	10	50
UCQ03 - All content is related to Web3 Ticket	10	100
UCQ04 - Texts are simple and concise	5	100
UCQ05 - No messages contain offensive content	5	100
UCQ06 - The app helps in the transition of the industry to blockchain technologies	5	50
UCQ07 - The Application provides visual feedback for user action	5	100
UCQ08 - A help button is provided	5	0
UCQ09 - Application provides enough information to the users so they can start managing their tickets	5	50
UI01 - Consistency of server data	5	100
UI02 - Track of user's purchases	5	100

Legends: FA - Functionality (Artist)
FF - Functionality (Fan)
FPO - Functionality (Place's Organization)
AC - Adaptability (Conversion)
AC - Adaptability (Maintenance)
AV - Adaptability (Versatility)
AV - Adaptability (Versatility)
UM - Usability (Menu Navigation)
UV - Usability (Validation)
UCQ - Usability (Content Quality)
UI - Usability (Integrity)

6.3 The Results

When performing usability tests with representative users, valuable insights and feedback were gathered to assess the three apps' user experience (Fan, Artist, and Place). These tests provided an opportunity to observe how users interacted with the application, understand their needs and expectations, and identify areas for improvement. The results show the strengths and weaknesses of the application, highlighting aspects that are accepted by the users and areas that required further improvement. The Charts in Figure 6.2 to 6.29 have the goal of presenting an overview of the outcomes from the usability tests, providing valuable insights that can guide future enhancements and optimizations to enhance user satisfaction and the overall usability of the application.

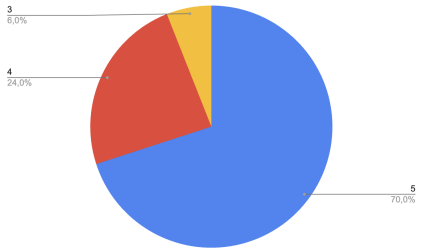


Figure 6.2: How easy was the process of creating/restoring a wallet?

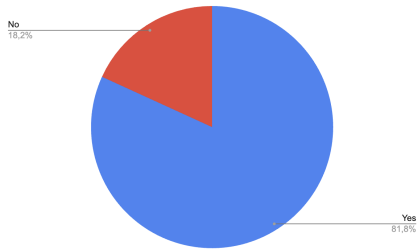


Figure 6.3: After creating/restoring your wallet you were asked to get the address of this wallet. Were you able to get your address?

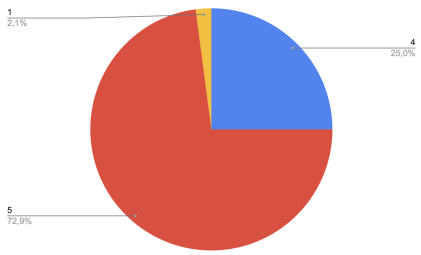


Figure 6.4: How easy was to get the address of your wallet?

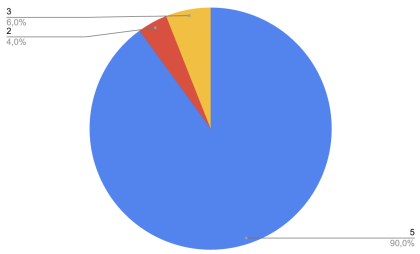


Figure 6.5: How easy is to understand where the catalog is located?

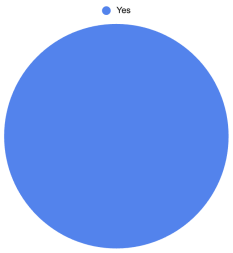


Figure 6.6: Could you see the show details?

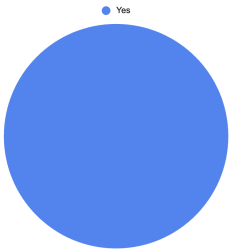


Figure 6.7: Try to know where the Place is located. Were you able of opening the google maps to see the place's location?

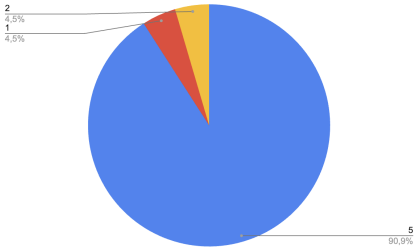


Figure 6.8: Try to buy a ticket from the primary sale. How easy was to find the button?

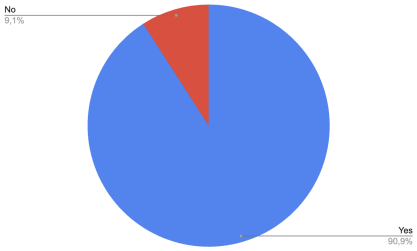


Figure 6.9: Could you choose a seat (or more) and complete the purchase?

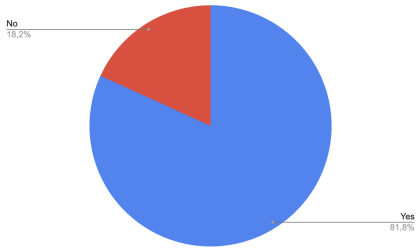


Figure 6.10: Can you see, on the wallet page, the ticket you purchased?

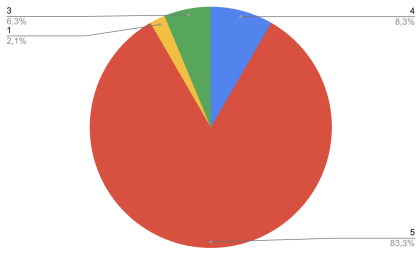


Figure 6.11: Try to sell the ticket. How easy it was to find the ticket sale feature?

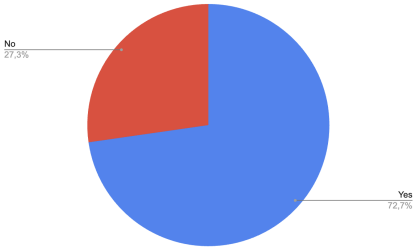


Figure 6.12: Did you managed to sell the ticket? Please remember that it can take some time to the blockchain to validate the transaction?

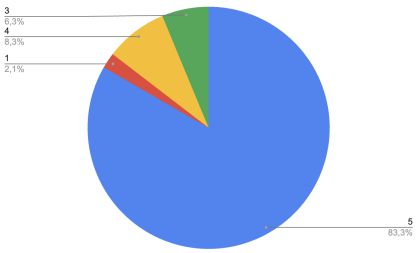


Figure 6.13: If the ticket disappeared from your wallet, we are ready to move on. Try to find the marketplace where your ticket is listed. How easy was the process of finding your ticket in the marketplace?

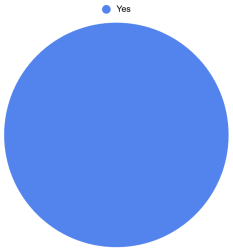


Figure 6.14: Was you able to find the available tickets to buy from second market?

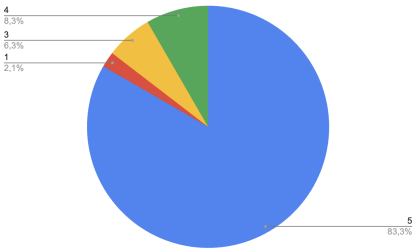


Figure 6.15: How easy was to complete the process of buying the ticket from second market?

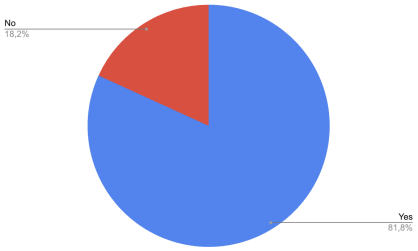


Figure 6.16: Wait a few moments for the blockchain to validate your transaction. Is the ticket in your wallet?

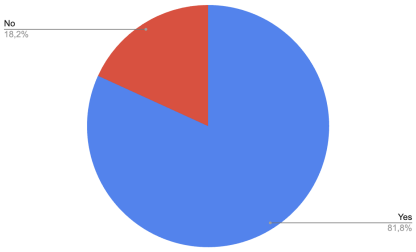


Figure 6.17: Choose a ticket to validate. Follow the steps on the screen to activate the ticket. Were you able to complete this process? Once again, the transaction needs to be validated by the blockchain, so it can take some minutes.

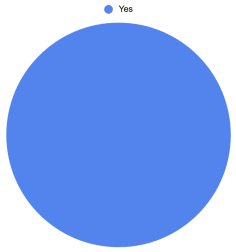


Figure 6.18: Try to logout from your account. Did you managed to do it?

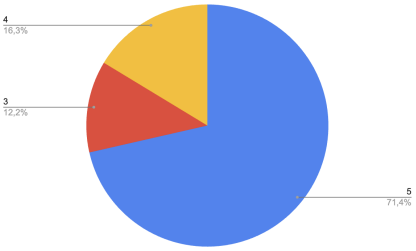


Figure 6.19: (Artist) Try to create a show, how easy was to find the functionality and to create the show?

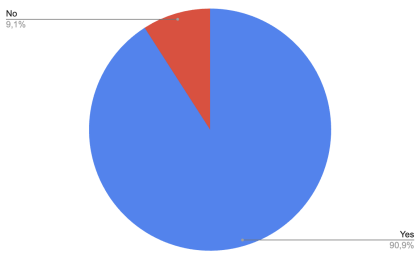


Figure 6.20: (Artist) After creating a show, were you able to consult the created ones?

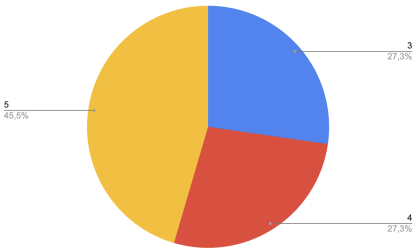


Figure 6.21: (Artist) How easy is it to search for shows in the gallery?

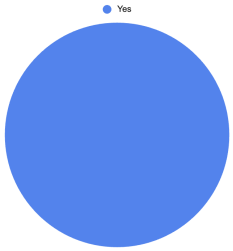


Figure 6.22: (Artist) Try to logout from your account. Did you managed to do it?

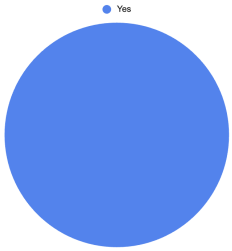


Figure 6.23: (Place) In the homepage can you see the requests for scheduling?

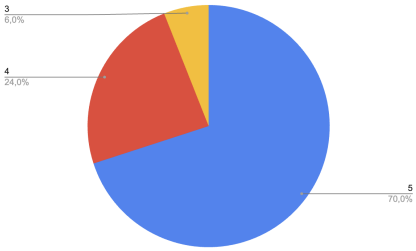


Figure 6.24: (Place) How easy it is to accept/refuse shows?

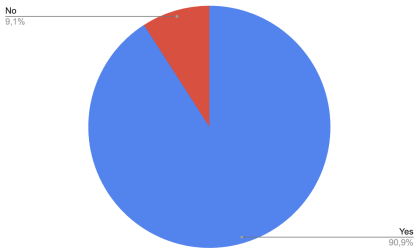


Figure 6.25: (Place) Can you see the show details?

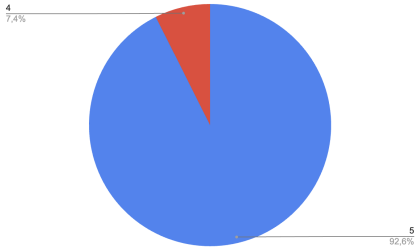


Figure 6.26: (Place) Its now time to scan a ticket at the entrance. In another device, open the qrcode of one ticket and try to scan it. How easy was to access the QR code reader?

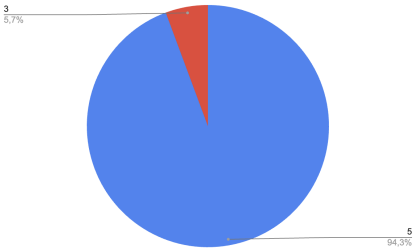


Figure 6.27: (Place) How easy it is to scan & validate the user's ticket?

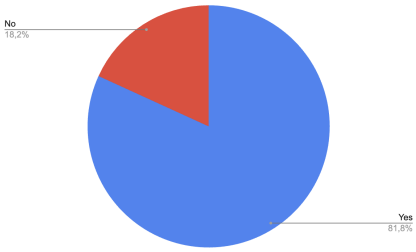


Figure 6.28: (General) Do you think all the messages are easy to understand?

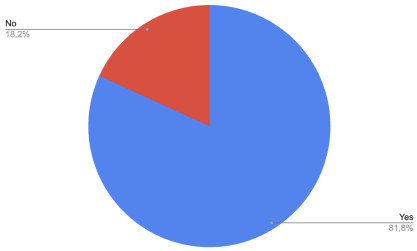


Figure 6.29: (General) Do you think the app provides enough information to the users to start managing their tickets?

The extensive collection of 28 charts, derived from the responses of 11 participants, provides an overview of the usability of the application. These charts reveal insights into various aspects of the user experience, focusing on the application's integration with the blockchain and its understanding. The data indicates that, on the whole, the application successfully fulfills its intended purpose. However, the charts also highlight that certain challenges persist, particularly regarding blockchain-related abstraction. These findings serve as a foundation for further improvement, for example, to approach the app UI design and the used terms to the ones used by the already established companies.

During the usability testing phase, participants were allowed to provide written feedback on their experience with the application. Several users shared helpful observations, highlighting areas for improvement and submitting important concerns.

" I didn't quite understand the process or where I have to go to perform the tasks that were asked in the test. I confess that I am a big noob in blockchain, but to have an app that is more accessible to everyone, you should improve the UX "

One recurring theme in the feedback was the complexity of blockchain technology, with some users expressing difficulties in understanding how to perform certain actions within the application. This feedback underscores the need for clearer explanations and user-friendly guidance regarding blockchain-related processes.

" The app feels really good overall, however I don't think the CTAs are very clear. I had a hard time figuring out that PUBLIC SALE was a button. "

" Staying on the same page seems strange, after I complete the purchase, it should redirect me somewhere. I felt like I had done something wrong, so I tried to buy another. "

User experience was also a common topic of feedback. Participants emphasized the importance of improving the overall user interface and interaction design to improve usability and ensure an intuitive experience.

" If I input a wrong password, there is no error/alert show me why I cannot pay the ticket. "

The feedback also revealed the existence of validation issues linked to erroneous password input. A user found difficulties with the validation process, was unable to buy a ticket, and was unable to determine why.

" The wallet creation/restoration was little bit annoying to go throw, having to write all the words or some of them felt irritating. "

The process of wallet recovery was also mentioned as a tiring task by some participants. This feedback highlights the importance of streamlining and simplifying the wallet recovery process to eliminate potential frustrations.

" It would be helpful having some feedback while transactions are pending for example"

" Just missing the transaction view in order to see the previous sales "

Participants also expressed a desire for the implementation of the "transactions" tab within the application, allowing them to easily access and review their recent purchases, like a transaction history feature.

" Once again, I feel like the CTA are not clear. I would expect to appear something like buy ticket on the card. "

" I don't know why but the process didn't run and it doesn't listen to any kind of transaction. "

Some users reported encountering issues when attempting to purchase tickets from the secondary market, receiving no visible feedback from their actions. Although this can be related to the user not having sufficient funds, the user doesn't receive any feedback from it.

" I wish the transactions wouldn't take as long, despite the warning popups that warn us of it - it usually took around 2 minutes everytime, which is a bit long for an impatient user. "

Lastly, participants expressed their impatience with the time it takes for blockchain transactions to be validated. This feedback, although not related to the server, represents the need for an area in the app showing the orders' status.

6.4 System Usability Scale

The SUS consists of a set of ten statements or items, designed to capture users' attitudes towards the system's usability. For this project, the responses were evaluated on a scale from 1 to 5, combined with "Yes or No" answers. This way, QEF can be complemented with the questionnaire responses, resulting in a more complete assessment. In the first phase of the evaluation, before getting the user's answers, the QEF was only evaluated from the developer's perspective and only took into account the implementation of each requirement. From this evaluation, the rating was 86%. However, the project can now be evaluated using feedback from the users, applying a grade from 1 to 5.

Table 6.6: QEF table: core adaptability requirements matching SUS and SUS Score

Requirement	SUS	Score
AC01 - The user controls his/her crypto assets	5	2,5
AC02 - The use of well known currencies helps to improve the usability of people not used to crypto	4	4
AM01 - Application presents a set of interfaces and endpoints, adding the possibity of intoducing new features	5	5
AM02 - The server must be scalable	5	5
AV01 - Application follows online store guidelines	4	2
AV02 - Application adjusts to screen resolution	4	3,2
AV03 - Application is multiplataform	5	5

Table 6.7: QEF table: core functional requirements matching SUS and SUS Score

Requirement	SUS	Score
FA01 - (Artist) Create a native wallet	4,55	4,55
FA02 - (Artist) Recover a native wallet	4,55	4,55
FA03 - (Artist) Consult Balance	4,36	4,36
FA04 - (Artist) Create shows and distribute them among users	4,45	2,225
FA05 - (Artist) Consult created shows	4,55	4,095
FA06 - (Artist) Consult show details	5	5
FA07 - (Artist) Redirect to map to see the event's location	5	5
FA08 - (Artist) Edit show details	0	0
FA09 - (Artist) Search for shows in the gallery	4	3,2
FA10 - (Artist) Logout	5	5
FF01 - (Fan) Create a native wallet	4,55	4,55
FF02 - (Fan) Create a native wallet	4,55	4,55
FF03 - (Fan) See all the show catalog	4,55	4,55
FF04 - (Fan) See show details	5	5
FF05 - (Fan) Buy the ticket (primary and second market)	4,29	4,29
FF06 - (Fan) Show the ticket's QrCode to be validated	4,09	4,09
FF07 - (Fan) Put tickets at sale	4	4
FF08 - (Fan) See all the tickets in the wallet	4,1	4,1
FF09 - (Fan) See wallet address	4,36	4,36
FF10 - (Fan) Copy wallet address	4,36	4,36
FF11 - (Fan) Logout	5	5
FPO01 - (Place's) Create a native wallet	4,55	4,55
FPO02 - (Place's) Create a native wallet	4,55	4,55
FPO03 - (Place's) See the last requests for scheduling	5	5
FPO04 - (Place's) Accept & Refuse Requests for scheduling	4,55	4,55
FPO05 - (Place's) See Show details	4,55	4,55
FPO06 - (Place's) Scan & Validate the user's ticket	4,86	4,86
FPO07 - (Place's) Logout	0	0

Table 6.8: QEF table: core usability requirements matching SUS and SUS Score

Requirement	SUS	Score
UM01 - The user interface is simple and fast, without slowing down when the number of users is high	5	2,5
UM02 - (Artist) Easy to create a show	4,45	4,45
UM03 - (Artist) Easy to distribute	0	0
UM04 - (Fan) Easy to buy a ticket either from artist or on second market	4,48	4,48
UM05 - (Fan) Easy to sell a ticket	4	4
UM06 - Application runtime does not have errors, and unexpected errors are handled and shown to the user	3	1,5
UM07 - User have access to a main menu	5	5
UV01 - (Place's) Easy access to QR Code reader	4,91	4,91
UV02 - (Place's) Easy to understand where to accept or refuse shows	5	5
UCQ01 - All the messages are easy to understand	5	2,5
UCQ02 - Application is clear to people not used to use crypto services	3	1.5
UCQ03 - All content is related to Web3 Ticket	5	5
UCQ04 - Texts are simple and concise	5	5
UCQ05 - No messages contain offensive content	5	5
UCQ06 - The app helps in the transition of the industry to blockchain technologies	3	1,5
UCQ07 - The Application provides visual feedback for user action	3	3
UCQ08 - A help button is provided	0	0
UCQ09 - Application provides enough information to the users so they can start managing their tickets	4,09	2,045
UI01 - Consistency of server data	5	5
UI02 - Track of user's purchases	5	5

From the perspective of the user, the app is evaluated in 75,11% which is calculated by taking the average of the column 'Score' which, in turn, is determined by multiplying the user's evaluation by the percentage of completeness. The total score is 3,7553 on a scale from 1 to 5, resulting in a 75,11% rating. The SUS values, it was considered the average of the evaluation given in the questionnaire. In the case of the "Yes or No" questions, the "No" corresponded to zero and the "Yes" to 5.

Chapter 7

Future Work

This chapter will explore future work and improvements that can be made to improve the application based on information gathered from the usability tests and user feedback. These inputs have provided a deeper understanding of the application's strengths, limitations, and areas for improvement.

One of the primary focuses of future work is to address the feedback received from each tester. This includes resolving any issues or difficulties they encountered while using the application and implementing the necessary fixes to improve the overall user experience.

By analyzing the feedback, it was identified specific pain points and usability challenges that need to be addressed to make the application more intuitive and user-friendly.

Additionally, some additional features are yet to be implemented. One crucial aspect is the implementation of a notifications system, using OneSignal, for example. This system will keep users informed about important updates, such as show schedules, ticket availability, order placement/confirmation, and any changes or updates related to their favorite artists or venues. By incorporating a notifications system, users can stay engaged and updated with the latest events and activities.

Another important needed development is the completion of the artist area, which will allow artists to edit their show details after the place's review. This feature will allow artists to make required changes or updates to their show information, ensuring that the Place Responsible will accept the schedule the next time the show is submitted for the Place's review.

Additionally, providing artists and place responsible individuals with access to the latest sales data is crucial for management. By implementing a feature that enables artists and places responsible individuals to analyze their sales data, they can take control of their performance, identify trends, and make informed decisions regarding future shows and events.

This also includes the development of a comprehensive place management system. This system will provide place responsible individuals with tools to manage many aspects related to their venues, including scheduling, capacity management, and coordination with artists, such as an interactive calendar as suggested in the mockups.

Lastly, it is to be noted that the UI/UX design needs to suffer a refurbishment, following the visual identity of the companies already established; that many requirements are not satisfied and need to be implemented; that to increase the abstractions, the system can allow the use of stablecoins which are tokens with intrinsic value that mimic the value of a fiduciary currency, such as the dollar.

Chapter 8

Conclusion

This master's thesis investigated the creation and testing of a unique mobile application that employs blockchain technology to improve the experience of artists and fans in the event ticketing sector. Several essential issues have been identified and explored during the investigation.

For starters, it is visible that higher abstraction between the real world and the underlying blockchain architecture would improve the application. While blockchain integration adds transparency and security to ticket transactions, there is still room for improvement in terms of streamlining the user experience and lowering the technological difficulties associated with blockchain operations. The program can be made more accessible and usable by offering a more user-friendly interface and abstracting the blockchain activities.

Secondly, it is essential to consider the regulatory framework within which the application operates. Portuguese regulations prohibit the resale of tickets at a higher price than the initial purchase. Adhering to such regulations is crucial for maintaining compliance and ensuring a fair and transparent ticketing ecosystem. Some price policies shall be implemented within the application to prevent any unauthorized price manipulation and protect the rights of both artists and fans, as presented before.

Additionally, it is recognized the importance of conducting further usability tests with a larger pool of participants to validate and refine the findings. Obtaining a more extensive range of feedback and insights will allow for a better assessment of user satisfaction and dissatisfaction, leading to targeted improvements in the application's design and features.

Moreover, the proposed application presents a promising solution for mitigating scams and fraudulent ticket activities. By leveraging the immutability and transparency of blockchain technology, the application provides a secure and inviolable environment for ticket transactions.

Lastly, there is the possibility of implementing a feature that enables artists to receive royalties for every ticket sale. A royalty mechanism in the application allows artists to be compensated for their creative performances. This can motivate artists to embrace the platform.

In conclusion, this master's thesis has explored the development, evaluation, and potential future improvements of a blockchain-powered mobile application for event ticketing. While the research has highlighted certain areas for refinement, the overall findings demonstrate the potential of this technology to enhance transparency, security, and user experience in the ticketing industry.

Bibliography

[1] Ayushi Abrol. Everything you need to know about binance smart chain (bsc), Sep 2022. visited on 2023-01-12.

[2] Binance Academy. Bnb smart chain vs. ethereum: What's the difference?, Oct 2022. visited on 2023-01-18.

[3] Team Crypto APIs. Utxo and account-based blockchains, Apr 2022. visited on 2023-01-17.

[4] Binance Authors. Binance and bnb chain: What's the difference?, Oct 2022. visited on 2023-01-12.

[5] BNB Chain Authors. Consensus engine: Bnb chain documentation, Jan 2023. visited on 2023-01-12.

[6] Emurgo Authors. Fibo 101: The 5 unique features of cardano nfts and why they matter, Aug 2022. visited on 2023-01-16.

[7] Monetsociety Authors. Understanding on-chain nfts vs off-chain nfts: The monet society. visited on 2023-01-29.

[8] Queensland Government Authors. Swot analysis, Dec 2022. visited on 2023-02-08.

[9] XRP Ledger Authors. Use cases & featured projects. visited on 2023-01-13.

[10] Cody Born. Ethereum proof-of-authority on azure, Aug 2018. visited on 2023-01-10.

[11] Marco Cavicchioli. Which big companies use blockchain and crypto?, Mar 2022. visited on 2023-01-03.

[12] Cci. What is cardano?, Jan 2023. visited on 2023-01-19.

[13] James Chen. Fiat money: What it is, how it works, example, pros & cons, Dec 2022. visited on 2023-01-10.

[14] Daniel Chicksand and Jakob Rehme. Total value in business relationships: Exploring the link between power and value appropriation. *Journal of Business & Industrial Marketing*, 33(2):174–182, 2018.

[15] Jonathan Clough. *Principles of cybercrime*. Cambridge University Press, 2010. visited on 2023-01-10.

[16] CoinMarketCap. Cryptocurrency prices, charts and market capitalizations. visited on 2023-01-11.

[17] Cointelegraph. What is the lightning network in bitcoin and how does it work?, Oct 2021. visited on 2022-12-21.

[18] Signum Community. Proof-of-commitment. visited on 2023-01-10.

[19] Lyle Daly. How is crypto taxed & do you pay taxes on bitcoin? visited on 2022-12-22.

[20] Dennis Dawson. Mint and burn nftokens, Dec 2022.

[21] Solana Developers. Solana status. visited on 2023-01-20.

[22] Solana Develops. Solana cookbook: Versioned transactions. visited on 2023-01-20.

[23] Jack Dunhill. Man searches through landfill for 8 years for $350 million lost bitcoin wallet, Dec 2021. visited on 2023-01-03.

[24] William Entriken, Dieter Shirley, Jacob Evans, and Nastassia Sachs. Erc-721: Non-fungible token standard. *Ethereum Improvement Proposals*, 1 2018.

[25] Tanja Eschberger-Friedl. The fuzzy front end of the innovation process, Jan 2018. visited on 2023-02-07.

[26] Cardano Foundation. Delegate your stake to build the network, earn rewards, and become part of the cardano journey. visited on 2023-01-10.

[27] Jake Frankenfield. 51% attack: Definition, who is at risk, example, and cost, 2022. visited on 2022-12-31.

[28] Jake Frankenfield. Ripple explained, Jan 2023. visited on 2023-01-13.

[29] Vishal Gaur and Abhinav Gaiha. Building a transparent supply chain, 2020. visited on 2022-12-14.

[30] The go-ethereum Authors. Home: Go. visited on 2023-01-19.

[31] Julija Golosova and Andrejs Romanovs. The advantages and disadvantages of the blockchain technology. *2018 IEEE 6th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*, 2018.

[32] The Australian Government. Ticket reselling in australia - consultation regulation impact statement. *The Treasury of The Australian Government*, November 2017.

[33] Patricia Guevara. Qfd: A guide to quality function deployment, Feb 2023. visited on 2023-02-11.

[34] Himanshi. Consensus mechanisms in blockchain, Jun 2022. visited on 2023-01-06.

[35] Morris B. Holbrook. *Consumer value: A framework for analysis and Research*. Routledge, 2006. visited on 2023-02-12.

[36] B2B International. What is the value proposition canvas?, Nov 2022.

[37] Thaddeus Dryja Joseph Poon. The bitcoin lightning network: Scalable off-chain instant payments, Jan 2016. visited on 2023-01-10.

[38] Tommy Kammerer. Minting nfts: Cardano developer portal, Sep 2022.

[39] S. Keshav. How to read a paper. *ACM SIGCOMM Computer Communication Review*, 37, 2007.

[40] Anoushk Kharangate. How to mint nfts on solana using rust and metaplex, May 2022.

[41] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. *Advances in Cryptology – CRYPTO 2017*, page 1–67, 2017.

[42] Peter A. Koen, Greg Ajamian, S Boyce, Allen Clamen, Eden Fisher, S. G. Fountoulakis, Albert Johnson, Pushpinder Puri, and Rebecca Seibert. 1 fuzzy front end : Effective methods , tools , and techniques. In *1 Fuzzy Front End : Effective Methods , Tools , and Techniques*, 2002. visited on 2023-02-06.

[43] Yulia Kosarenko. Use case diagrams and how to use them, Aug 2022.

[44] Shilpi Kumar. Innovation goes beyond new product development (npd) and requires a cohesive strategy, Oct 2018. visited on 2023-02-06.

[45] lakshita. Introduction to merkle tree, Sep 2022. visited on 2022-12-21.

[46] I.-C Lin and T.-C Liao. A survey of blockchain security issues and challenges. *International Journal of Network Security*, 19:653–659, 09 2017. visited on 2022-12-21.

[47] Masha McConaghy, Greg McMullen, Glenn Parry, Trent McConaghy, and David Holtzman. Visibility and digital art: Blockchain as an ownership layer on the internet. *Strategic Change*, 26(5):461–470, 2017. visited on 2022-12-23.

[48] Rachel McIntosh. Is the binance smart chain centralized? messari researchers raise concerns: Finance magnates, Apr 2021. visited on 2023-01-20.

[49] Avradip Mandal Mic Bowman, Debajyoti Das. On elapsed time consensus protocols, 2021. visited on 2023-01-10.

[50] Monia Milutinović. Cryptocurrency. *Ekonomika*, 64(1):105–122, 2018. visited on 2023-01-11.

[51] Sumi Mudgil. How to write & deploy an nft (part 1/3 of nft tutorial series), 2021.

[52] Nftify. Nft gas fees and how gasless transactions are possible. visited on 2023-01-19.

[53] Carnegie Museum of Art (CMOA) Authors. The cmoa digital provenance standard., Oct 2016. visited on 2022-12-23.

[54] P4Titan. Slimcoin : A peer-to-peer crypto-currency with proof-of-burn, May 2014. visited on 2023-01-10.

[55] Glenn Parry, Oscar F. Bustinza, and Ferran Vendrell-Herrero. Copyright and creation: Repositioning the argument. *Strategic Direction*, 30(3):32–35, 2014. visited on 2023-01-07.

[56] William F. Patry. *Moral panics and the copyright wars*. Oxford University Press, 2009. visited on 2023-01-10.

[57] Ferdinand Regner, André Schweizer, and Nils Urbach. Nfts in practice – non-fungible tokens as core component of a blockchain-based event ticketing application. *ICIS 2019*, pages 1–9, 12 2019.

[58] Paul Roberts and Warwick Manufacturing Group. Quality function deployment. visited on 2023-02-10.

[59] Eric Rosenberg. What is a consensus mechanism?, Sep 2022. visited on 2023-01-10.

[60] Amazon Web Services. What is decentralization in blockchain? visited on 2022-12-14.

[61] Petra Sevcikova, Monica Ibido, and Claudia Lueje. Innovation management — fundamentals and vocabulary, 2020. visited on 2023-02-06.

[62] Manoj Sharma. Value analysis: Meaning, phases, merits and limitations, Apr 2015.

[63] Sheda Sheda. How to: Business model canvas explained, Oct 2022.

[64] Myounghyun Shim, Sam Gyun Oh, Patricia Cook, and Martha Casantosan. Iso 21127:2006, Oct 2014. visited on 2022-12-23.

[65] Fábio Carvalho Silva. Criptoativos taxados a 28% mas hÁ exceções. intermediários pagam imposto de selo, Oct 2022. visited on 2022-12-21.

[66] SoFi. What is ripple xrp? everything to know for 2022, Dec 2022. visited on 2023-01-20.

[67] Cryptopedia Staff. What is ethereum blockchain; and its key use cases?, Dec 2021. visited on 2023-01-11.

[68] Scott Nadal Sunny King. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake, Aug 2012. visited on 2023-01-10.

[69] S. Supraja and P. Kousalya. A comparative study by ahp and topsis for the selection of all round excellence award. *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, 2016.

[70] Andrew Stuart Tanenbaum. Distributed operating systems. *1996 CERN School of Computing*, 1996.

[71] Indeed Editorial Team. What are functional programming languages and why use them?, Oct 2022. visited on 2023-01-20.

[72] Namecoin Team. Is namecoin's support of atomic name trades a feature primarily aimed at squatters? visited on 2022-12-19.

[73] Edgar Mondragón Tenorio. Advantages and disadvantages of blockchain: Bbva suiza, Jun 2022.

[74] Ediomi Udoh. How to mint a music nft on binance smart chain, Sep 2022.

[75] Ziyuan Wang, Lin Yang, Qin Wang, Donghai Liu, Zhiyu Xu, and Shigang Liu. Artchain: Blockchain-enabled platform for art marketplace. *2019 IEEE International Conference on Blockchain (Blockchain)*, 2019. visited on 2022-12-20.

[76] M M Widianta, T Rizaldi, D P Setyohadi, and H Y Riskiawan. Comparison of multi-criteria decision support methods (ahp, topsis, saw & promenthee) for employee placement. *Journal of Physics: Conference Series*, 953:012116, 2018.

[77] Valarie A. Zeithaml. Consumer perceptions of price, quality, and value: A means-end model and synthesis of evidence. *Journal of Marketing*, 52(3):2–22, 1988.

Appendix A

Database Diagram

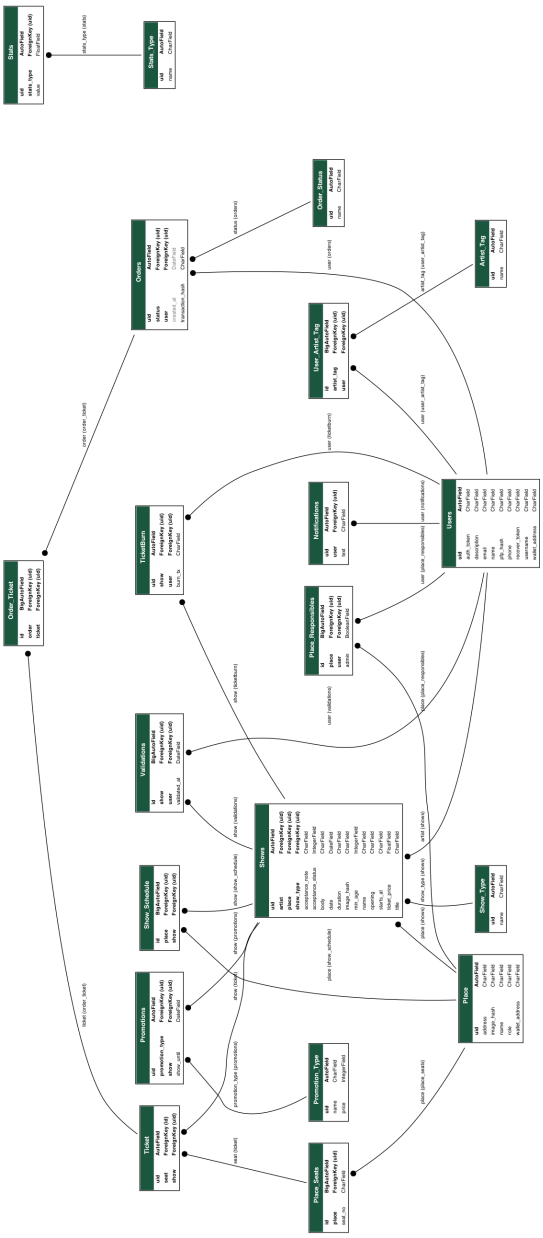


Figure A.1: Overview of the database diagram