POLITECNICO DI MILANO

Corso di Laurea Magistrale in Ingegneria Informatica

Dipartimento di Elettronica e Informazione

DeMusic
A decentralized application for artists

Relatore: Prof. Francesco Bruschi,

Correlatore: Prof. Vincenzo Rana

Tesi di Laurea di:

Stefano De Cillis, matricola 927879

A mia madre e alla famiglia Palmieri,
che mi hanno cresciuto ed educato nonostante le difficoltà.
Senza di loro, tutto ciò non sarebbe stato possibile.

Acknowledgements

I would like to thank my advisors, Professor Francesco Bruschi and Professor Vincenzo Rana, for giving me the opportunity to work on this project and supporting me during its development.

It is paramount to point out that both this document and the entire project have been accomplished with the support of Lucas Manini, Tommaso Paulon and two spin-off of Politecnico di Milano, KNOBS and BCode.

I would also like to thank both my friends and colleagues who have always supported me throughout my journey at Politecnico.

SD

Contents

List of Figures

List of Abbreviations

API	Application Programming Interface
CRYPTO	Cryptocurrency
DAPP	Decentralized Application
IPFS	Inter-Planetary File System
P2P	Peer to Peer
PRO	Performing Rights Organizations
PoS	Proof of Stake
PoW	Proof of Work
WT	WebTorrent Client
YAML	YAML Ain't Markup Language

Sommario

Il mercato discografico è in continua espansione. Solo dal 2020, tenendo presente che la pandemia ha messo in ginocchio tutti i live, si registra comunque un incremento nel giro d'affari del 7,4% secondo "Il Sole 24 Ore" [1] facendolo arrivare complessivamente ad un valore che si aggira sugli 21,6 miliardi di dollari. Soltanto in Italia, questo mercato è cresciuto del 1,44%. Sebbene da un lato questo mercato sia in forte crescita, dall'altro presenta degli ostacoli per artisti emergenti o poco conosciuti. Data la forte competitività, non è raro che artisti più giovani si tirino indietro per paura di non essere all'altezza. Inoltre, bisogna mettere in conto che, la carriera da musicista non risulta essere profittevole, se non per artisti già affermati.

Dato il gran numero di canzoni ogni giorno pubblicate, è inoltre difficile costruire un database centralizzato che sia sempre aggiornato. Questo problema risulta dunque nelle royalties mancate. Questo comporta un mancato introito da parte degli artisti e una difficoltà oggettiva nel raccogliere quotidianamente tali informazioni dalle case discografiche.

La blockchain è una tecnologia che viene già usata in diversi settori, partendo dalla finanza decentralizzata fino ad arrivare a vere e proprie applicazioni. Il suo obiettivo è quello di rimuovere intermediari che sono superflui nel processo.

Questa tesi affronta i problemi degli artisti emergenti e delle royalties tramite l'aiuto di un sistema decentralizzato facilmente accessibile da tutti. Al giorno d'oggi, affinché un artista possa far ascoltare un suo brano ad una persona, una moltitudine di intermediari

deve essere presente affinché il brano venga pubblicato e condiviso. Con il nostro lavoro, vogliamo andare a costituire un sistema che permetta all'artista la distribuzione dei propri contenuti. L'utilizzo di quest'ultimi verrà dunque pagato direttamente all'artista dagli utenti della rete, senza utilizzo di intermediari.

Un altro importante strumento che andremo ad utilizzare sarà l'impiego dei token all'interno della blockchain per generare valore. Questo valore, generato dall'artista stesso, potrà essere facilmente venduto alla community come investimento nella sua carriera.

Summary

The music industry is continuously growing. According to Il Sole 24 Ore, only in 2020, this business increased by 7,4% bringing its value around 21,6 billion dollars [1]. This growth has been meaningful considering the pandemic made suffering all the live events. Only in Italy, the music industry increased its value by 1,44%. Although on one side this market is growing at such a fast pace, on the other side it has obstacles for young and emerging artists. Given the strong competition, it is not a rare case in which younger artists quit their careers because they fear no being able to emerge. Moreover, it is important to take into account that their career does not provide earnings if the artist is not yet well known.

Given the large number of songs that every day are published, it is difficult to build a centralized database that is always up to date. Sometimes, this results in missing royalties. The consequences of it are mainly two: artists can not earn and difficulty to daily track royalties by the Labels.

The blockchain is a technology applied to different areas, beginning from decentralized finance and arriving at real applications. The tenet on which the blockchain is based is to remove unnecessary intermediaries in the process.

This thesis develops a decentralized system, easily accessible by everyone, that attempts to solve the problems for young artists and the royalties, previously explained. Nowadays, in order to distribute music content, multiple intermediaries are needed to the

process before the song can be published and shared with the listeners. With our project, we want to build a system that allows artists to distribute their content without the need to involve other intermediaries. The use of this content will be directly paid to the artist, creator of the content.

Another important tool we are going to use in our project are tokens, reshaped in order to generate tangible value inside the blockchain. This value, generated by the artist itself, can be easily sold to the community as an investment in their career.

Chapter 1

Introduction

The proposed study seeks to create a decentralized environment where young artists can be financially sustained by the community. A way in which the music streaming service can be enabled by the community itself without using intermediaries will be presented. To achieve this idea, the project develops a decentralized application where artists can generate digital assets and sell them to other users. With this approach, the artist can be funded directly by his fan base and the community.

A decentralized application (dApp) is an application or program that exists and runs in a decentralized environment, such as the blockchain, a decentralized and distributed ledger, or Peer to Peer (P2P) network. There is no single authority. In order to ensure the proper behaviour of a decentralized application, simple programs, called smart contracts, are stored into the blockchain so that clear rules are established.

The blockchain is resistant to changes since data can not be altered without affecting all the subsequent blocks. This approach has two main advantages: durability and traceability. Once the asset is generated by the artist, it will persist inside the blockchain. It is decoupled from the life of this project. The state of a blockchain can be changed by mining a new block. Powerful computers solve complex computational math problems in order to create a block of transactions, records of the changes from the last block. Since all the transactions are stored inside each block, everyone can easily track assets.

A second milestone for the project is to easily manage the royalties the artists and third actors have on the music. Retrieving such information is a common problem of both Labels and third-party companies. In many cases, the information retrieved is not imprecise. Royalties are used by each intermediate to distribute the earnings.

Crypto tokens are a type of cryptocurrency that represents an asset or specific use and resides on their blockchain. With the help of tokens (crypto tokens), the idea is to track the ownership of each digital asset so that everyone can query the blockchain and retrieve the latest data as if it were a giant database.

The technologies previously mentioned will be discussed in detail in the next chapter.

1.1 Context of the artist

Nowadays, there is no easy way for musicians to be yet profitable in what they do and no possible way to be independent. In order to understand the problems we are going to express later in the document, we need to dig down into the mechanism in which a normal artist needs to be if he wants to emerge.

Differently from other careers, young artists have no easy way to succeed without pay upfront for several services to record their music and distribute it to users. It is also known that during their early stage, many artists perform in as many places as they can without being paid. In particular, according to Tunedly [2], a platform where music creators get connected with world-class session musicians for professional music production and music publishing, the fundamental struggles are:

- funding the projects by himself since getting well-supported through music can require a great deal of time and perseverance
- promoting the music into the right channels can be difficult and expensive
- having a tight competition can drown out the hard work and kill artist's ambitions
- feeling burned out given the tight budget they rely on, the large amount of effort they are putting into their music and the uncertainty of their future

He needs to find a label [3] that helps him in several processes such as marketing and distribution. Usually, labels get royalties over the songs they help to distribute. It is important to say that, at this point, the artists are not profitable till they reach enough people. It can take years to have the first earnings. Therefore, artists are discouraged to pursue their dreams.

On the other hand, there is the problem of royalties. As previously said, labels help artists to distribute their music to as many people as possible in exchange for royalties. Especially during Covid-19, when all the concerts have been cancelled, what should've been the most profitable way for artists to earn over their own creations is through streaming services.[4]

According to *RollingStone.com*, the artist earns the only 12% of the profit made through streaming services. The rest of the profit is leaked involved in producing and distributing music, e.g. the costs of running record labels, streaming companies, satellite radio and other midpoints that have to exist between artists and listeners. Given this scenario, the main source of income for an artist is the live events and the merchandising, even if the streaming services are growing in terms of usage. Indeed, this problem has been highlighted during the pandemic since their main source of income has suffered.

In this scheme, we can spot four fundamental actors: the artist, the recording label (from now on we will call it "Label"), the streaming service provider (from now on we will call it "Provider") and the people who want to listen to the artist's music collection.

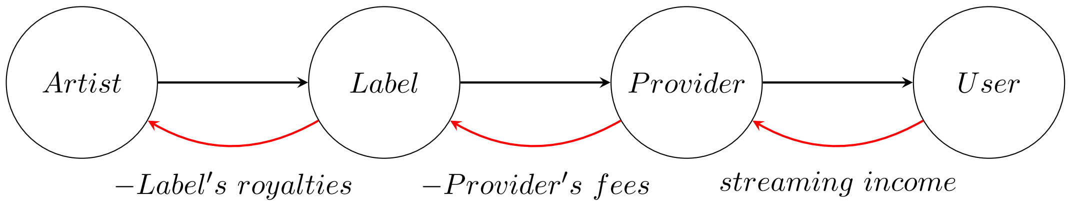


Figure 1.1: Interaction between the actors in the system. The red arrows are meant to be the income flow.

As we can see from the previous graph, the artist is the first who starts the process but the latter who has gained. As we said before, the earning of an artist is close to 20% (even less)[5] of the total income. For a young artist, there is no possible solution to escape from this scheme.

Furthermore, we have more actors involved in the system which are pure intermediaries. Taking into account the pretty complicated formula that Spotify uses to distribute royalties, the label often receives a significant part of the income. Moreover, we need to remember that the revenue that artists get may vary depending on their contract with their label.

1.2 Royalties tracing

In order to introduce the concept of *royalties* and how they can be optimized, we need to provide a definition. According to *Investopedia.com*, "a royalty is a legally binding payment made to an individual or company for the ongoing use of their assets, including copyrighted works, franchises, and natural resources" [6]. In music, royalties are split between the composition, the production and the distribution of the music, as the main asset.

In order to further explain how royalties work and how they are distributed among the artists and labels, we should give a look and take for instance the document of the S.I.A.E. in Italy [7]. Let's keep in mind that they work in a similar way in the rest of the world.

When registering a song with the PRO, Performance Rights Organisation, the Performance Royalty is actually split 12/12 into two sub-royalties: Songwriting and Publishing. Songwriter Royalties will always be paid out to the credited songwriters of the composition. There is absolutely nothing any record label, publisher, producer, manager, or bandmate can do to change this royalty.

Publishing makes up the other 50% of the Performance Royalty and, unlike Songwriter Royalties, Publishing can be assigned to outside entities called publishing companies. Music Publishing Companies temporarily take ownership of the artist songs and manages the lifespan and monetary potential for that music.

Royalties are difficult to keep track of. In many cases, royalties are never corrected or updated. There is also no easy way to compute their distribution. At the moment, the state of the art is to call each label and ask how royalties are distributed. This process is done day by day for each song. Due to the endless publications of new songs, having a centralized database and set standards for music metadata has stumped many of music's largest and most powerful entities for decades.

1.3 Document Structure

This thesis is organized as follows:

- In Chapter 1 we introduced the problems related to young artists, the distribution of their content and the royalties.

- In Chapter 2 we define the technologies we are going to use. In particular, we provide an overview to the main decentralized tools that made possible the realization of this thesis.
- In Chapter 3 we show the analysis of the NFTs market and the volume of transaction in the blockchain in the recent years.
- In Chapter 4 we present the state of arts of the current projects that already exist and try to solve these problems. In particular, we will have a look at decentralized systems, what they do and how they are built.
- In Chapter 5 we propose our solution. We will show all the components inside deMusic platform and how they work.
- In Chapter 6 we briefly evaluate the two fundamental metric on which the project is based.
- In Chapter 7 we discuss the evolution of the project and the next steps. In particular, we will have a look to what we achieved and what we have designed for the next period.
- In Chapter 8 we conclude the thesis by defining what we have done so far and insight to future development for this project.

Chapter 2

Decentralized Technologies

In a centralized network, all users are connected to a central server that stores complete network data and user information. On the contrary, a decentralized network has several peer-to-peer user clusters wherein each one has its separate server that stores data and information relevant to only that particular cluster.

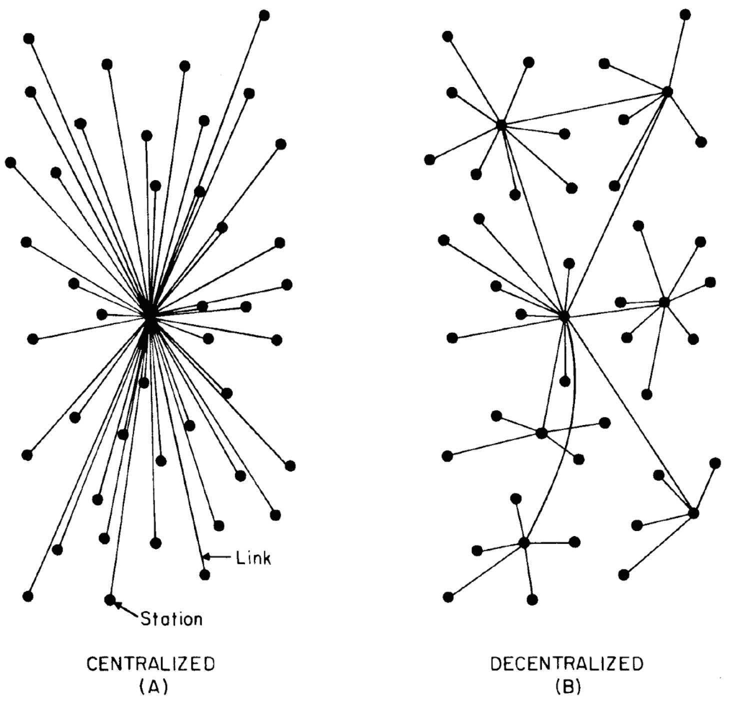


Figure 2.1: Decentralized vs Centralized networks

The technologies we are going to describe are used to build decentralized and distributed systems. In particular, the term *distributed* means computation is spread across multiple nodes instead of just one. *Decentralized* means no node is instructing any other node as to what to do. Therefore, the code runs on a peer-to-peer network of nodes and no single node has control over the other ones.

In this chapter, the main technologies on which deMusic is based will be discussed in detail. In particular, the following sections will describe the blockchain, BitTorrent and the Distributed Hash Table (DHT). All of them are used to make every component decentralized and distributed.

The main goal of using the blockchain in this project, as a shared and distributed ledger, is to enable trustless interactions and business disintermediation, thus lowering the transaction costs. Moreover, the benefit of not having a central server brings redundancy and security, since the information is shared among all the nodes in the network.

A whole chapter is dedicated to them so that the reader has a brief but detailed description of each technology previously mentioned.

In case the reader is aware of how they work, they can jump directly to Chapter 3.

2.1 Blockchain

A blockchain is a digital ledger of transactions that is duplicated and distributed across the entire network of computer systems on the blockchain. It is designed in a way that makes it difficult or impossible to change, hack, or cheat the system. Nowadays, most companies run on information. The faster it is received and the more accurate it is, the better. Blockchain is ideal for delivering that information because it provides immediate, shared and completely transparent information that can be accessed only by permission network members. A blockchain network can track orders, payments, production and much more.

The blockchain was invented by Satoshi Nakamoto (alias of a person or a group of people) in 2008 to deliver the public transaction ledger of the cryptocurrency bitcoin. Based on its whitepaper [8], Bitcoin is "a purely peer-to-peer version of electronic cash that would allow online payments to be sent directly from one party to another without going through a financial institution". The invention of the blockchain made it the first digital currency to solve the double-spending problem without the need for an intermediary, such as a trusted authority or central server. Since Bitcoin was raised, new blockchains and cryptocurrencies emerged.

The second-largest blockchain is Ethereum, a decentralized, open-source blockchain with smart contract functionality. A smart contract is a simple program that runs on Ethereum and it enables different activities, including decentralized applications

The key elements a basic blockchain is built on are transactions and blocks. The transactions are used to keep track of the movement of an asset that can be tangible (a product) or intangible (intellectual). For instance, we can use transactions to record the state of an item during its production. A set of transactions is then collected into a new block which is attached to the latest one inside the network. These blocks form a chain of data as an asset moves from place to place or ownership changes. The blocks confirm the exact time and sequence of transactions, and the blocks link securely together to prevent any block from being altered or a new block being inserted between two existing blocks.

A new block is linked to the previous one through the latter hash. A cryptographic hash function is an algorithm that takes an arbitrary amount of data as input and produces a fixed-size output of enciphered text called *hash*. A slight difference in the input generates a completely different output. In this way, building a new block based on the previous block hash ensures that middle blocks inside the blockchain can not be altered.

Every time a new transaction occurs on the blockchain, it is added to every node in the network. This results in a decentralised database managed by multiple participants, known as Distributed Ledger Technology (DLT).

The blockchain is built in such a way to enable different scenarios. It can be used in multiple areas such as cryptocurrencies, smart contracts, financial services, videogames, etc. This is possible because the blockchain can keep track of the movement of an asset. Virtually anything of value can be tracked and traded on a blockchain network, reducing risk and cutting costs for all involved. For instance, blockchain technology can be used to enhance supply chain management and business processes. One of the biggest projects in this area is VeChain, a blockchain platform that can track quality, authenticity, storage temperature and last-mile delivery of a medicine pack or an alcohol bottle right from the manufacturing facility through to the final delivery to the end customer.

In Ethereum there exist Ethereum Request for Comments, ERC, that are technical documents used by smart contract developers at Ethereum. A smart contract is a simple program that runs inside of the Ethereum Virtual Machine (EVM), a particular virtual machine that runs on top of the nodes of this blockchain, and it is used to implement a set of rules for the transactions that interacts with.

To track assets, tokens are used. A token means an asset that can be utilised by the user and it can be classified into one of the following types: fungible, non-fungible and semi-fungible. Fungible tokens, backed by the ERC20 [9], are interchangeable tokens. Such tokens work fine for cryptocurrencies, and in fact, fungibility is the fundamental feature of any currency. They are built in such a way that each fraction of a token is equivalent to the next one.

Non-fungible tokens, backed by ERC721 [10], are special tokens that represent unique, collectable items. Crypto Kitties is the most popular example of non-fungible, collectable tokens. Every CryptoKitty is unique, and no two CryptoKitties are the same.

Semi-fungible tokens (SFT), backed by ERC1155 [11], are NFTs much like any ERC721 token. But to the end-user, these SFTs operate as one of a larger volume of scarce tokens. For instance, it is possible to associate this kind of tokens with concert tickets. There are multiple tickets for the same concert but they are limited and associated with one concert only.

Definition 2.1.1 (Blockchain Wallet)**.** A blockchain wallet is a digital wallet that allows users to store and manage their bitcoin and ether. Once the wallet is created, the user is provided with a public key, which is a unique identifier similar to a bank account number.

2.1.1 BitTorrent

BitTorrent is a peer-to-peer (P2P) protocol used to download files using a distributed peer-to-peer network. This means that the computers in a BitTorrent "swarm" (downloading and uploading the same torrent with other machines) transfer data between each other without the need for a central server.

BitTorrent is unique because it distributes the total sharing effort across all users who have downloaded or are in the process of downloading a file.

Its advantage over plain HTTP is that when multiple downloads of the same file happen concurrently, the downloaders start backing the upload bandwidth, making possible to support very large numbers of downloaders with only a modest increase in its load.

To share content inside the network, torrent files are used. A torrent is a format of a file that includes all the information needed to download such content. In particular, it can be found information of the file, such as the length and the number of chunks, and the information of the creator of that torrent.

Given a torrent file, the BitTorrent client contacts a “tracker” specified in such file. The tracker is a special server that keeps track of the connected computers. The only purpose of the tracker is to keep track of the BitTorrent clients connected to the swarm. The tracker shares the IP addresses of the nodes, allowing them to connect to each other. Once connected, the BitTorrent client downloads bits of the files in the torrent in small pieces, downloading all the data it can get. Once the BitTorrent client has some data, it can then begin to upload that data to other BitTorrent clients in the swarm. In this way, every downloader is also uploading the same file to other nodes. Each downloader called also leecher, ensures the torrent stays fast by increasing the total upload bandwidth in the network.

The key to scalable and robust distribution is cooperation. With BitTorrent, those who get your file tap into their upload capacity to upload the file to others at the same time.

It is important to notice that, in recent years, BitTorrent has been used for illegal activities, such as sharing copyrighted content over the web. For instance, this protocol is used by "Pirate Bay", a Swedish anti-copyright group, to share several types of content [12]. However, BitTorrent is not synonymous with piracy. For instance, Blizzard uses a custom BitTorrent client to distribute updates for games bringing players all the advantages previously explained.

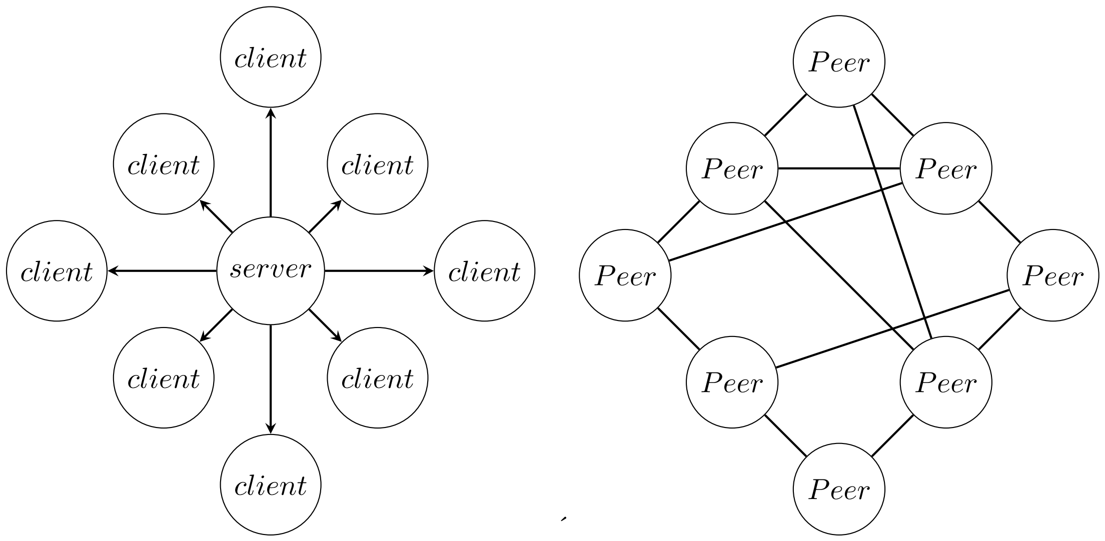


Figure 2.2: Centralized protocols compared to P2P protocol.

2.1.2 DHT

Even if the P2P architecture was designed to be decentralized, the fundamental problem remains the locations of each node. In order to find and connect nodes in the swarm, a tracker needs to exist to keep track of all the addresses. Recently, a decentralized "trackerless" torrent system, the Distributed Hash Table (DHT), was introduced. It allows BitTorrent clients to communicate with each other without the need for any central servers.

Magnet links can be used in a number of contexts, they are particularly useful in peer-to-peer file sharing networks. They allow resources to be referred to without the need for a continuously available host, and can be generated by anyone who already has the file, without the need for a central authority that issues them [13].

When adding a new torrent using a magnet link, the DHT node contacts the closest nearby nodes and this process repeats in an attempt to locale the information about the torrent.

Each node has a globally unique identifier known as the "node ID." Node IDs are chosen randomly from the same 160-bit space as BitTorrent infohashes: SHA1 hashes over the part of a torrent file that includes the information of the file. A "distance metric" is used to compare two node IDs or a node ID and an infohash for "closeness". Nodes must maintain a routing table containing the contact information for a small number of other nodes. The routing table becomes more detailed as IDs get closer to the node's own ID. Nodes store information about many other nodes in the DHT that have IDs that are close to their own and few of them with IDs that are very far away.

With the DHT protocol each peer becomes a tracker. This means that BitTorrent clients no longer need a central server managing the swarm. Indeed, BitTorrent becomes a fully decentralized peer-to-peer file transfer system.

DHT can also work alongside traditional trackers. For example, a torrent can use both DHT and a traditional tracker, which will provide redundancy in case the tracker fails.

Chapter 3

NFTs Economic Analysis

In order to launch the project, some market analysis has been carried out. The following analysis has been placed here so that the reader has an overview of the exponential growth the blockchain had in terms of volume of transactions and the value in recent years.

For the purpose of this document, this chapter will be the only one focused on the economic side, exploring the market we are attacking. All the data that we are going to analyse are related to the years 2018-2020.

3.1 Performance

The exponential growth related to the NFTs has been impressive during the year 2020. In the past years, many Decentralized Finance (DeFi) projects and digital assets raised the interest of big companies such as Tesla, PricewaterhouseCoopers, Facebook, etc. It can be noted that this exponential growth matches the period of the pandemic. According to Cointelegraph, a leading independent digital media resource, the traditional financial system was questioned during the pandemic because the Europeans shifted the payments toward cashless methods and cryptocurrencies [14].

As it can be seen, the web queries on Google related to NFTs reached a new all time high during September 2020. The figure below has been taken from Nonfungible, a blog that covers blockchain news [15].

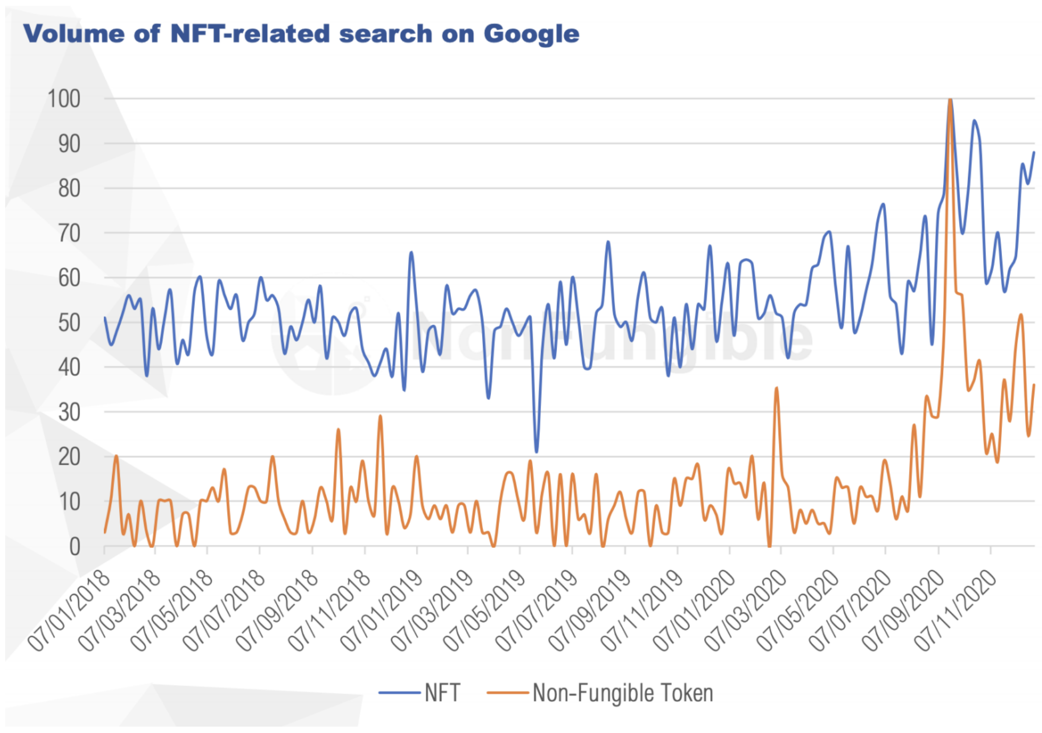
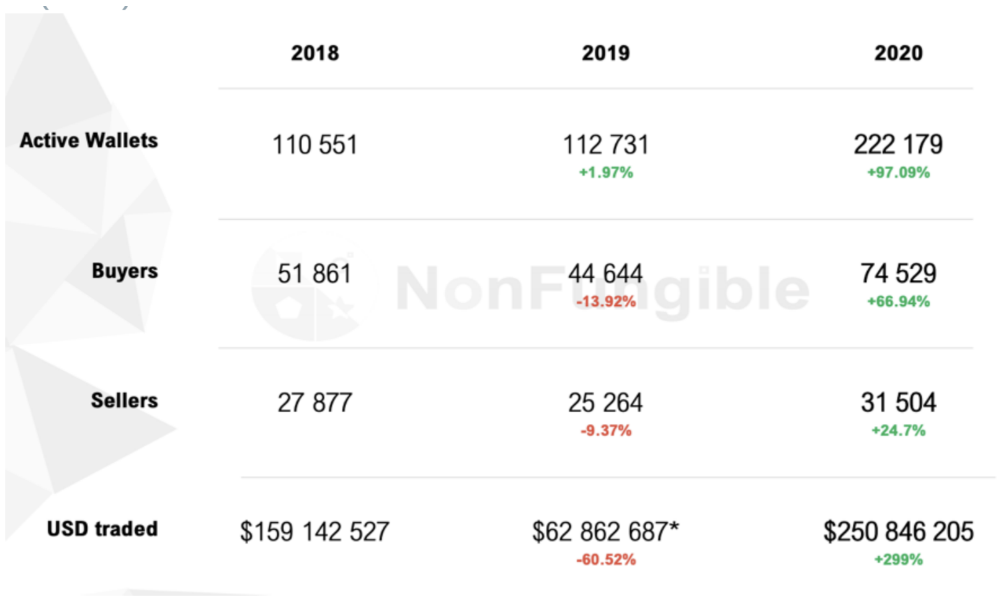


Figure 3.1: NonFungible - Google Search 2020

It is important to notice that the term NFT is related to a particular technique used in farming. This explains the growth during the queries made on Google during the first period of 2018.

The growth we had during 2020 raised exponentially as a consequence of the hype associated with DeFi projects that were using NFTs. During this period, we experienced rapid growth of the market that raised the interest related to decentralized projects and the blockchain itself.



	2018	2019	2020
Active Wallets	110 551	112 731 +1.97%	222 179 +97.09%
Buyers	51 861	44 644 -13.92%	74 529 +66.94%
Sellers	27 877	25 264 -9.37%	31 504 +24.7%
USD traded	$159 142 527	$62 862 687* -60.52%	$250 846 205 +299%

Figure 3.2: NonFungible - Volume of NFT-related search on Google

The year 2020 has been a peculiar year for the blockchain. It is possible to see how the number of active wallets between 2019 and 2020 increased significantly. The growth is almost 60% more for the buyers which enlarge, even more, the community and the owners of NFTs.

The transactions volume, made in USD, during 2020 scaled as much as the value of cryptocurrencies we had in 2019 and the new decentralized projects of DeFi launched during the same year.

3.2 Capitalization

The capitalization of a market represents the total value of the active projects in a specific market.

Related to NFTs, this value is quite difficult to estimate since it requires a particular evaluation of each project and goods. The estimation needs to consider that some projects lost all the value they had and many assets have too much high value.

NonFungible made this estimation based on the volume of all the assets that exist, times the average value of this type of asset. In order to avoid the value of some assets that lost interest over time, a new metric has been introduced, the liquidity presents in this field.

The figure below suggests a new possible exponential growth of this market, taking into consideration the capitalization of the previous years.

It is worth noticing that 2020 has been a particular year. The pandemic could have affected the market in such a way that the growth we have had so far, can reduce in the future. The estimations have been taken carefully, including the offer of all the NFTs for each project and it is related to their liquidity.

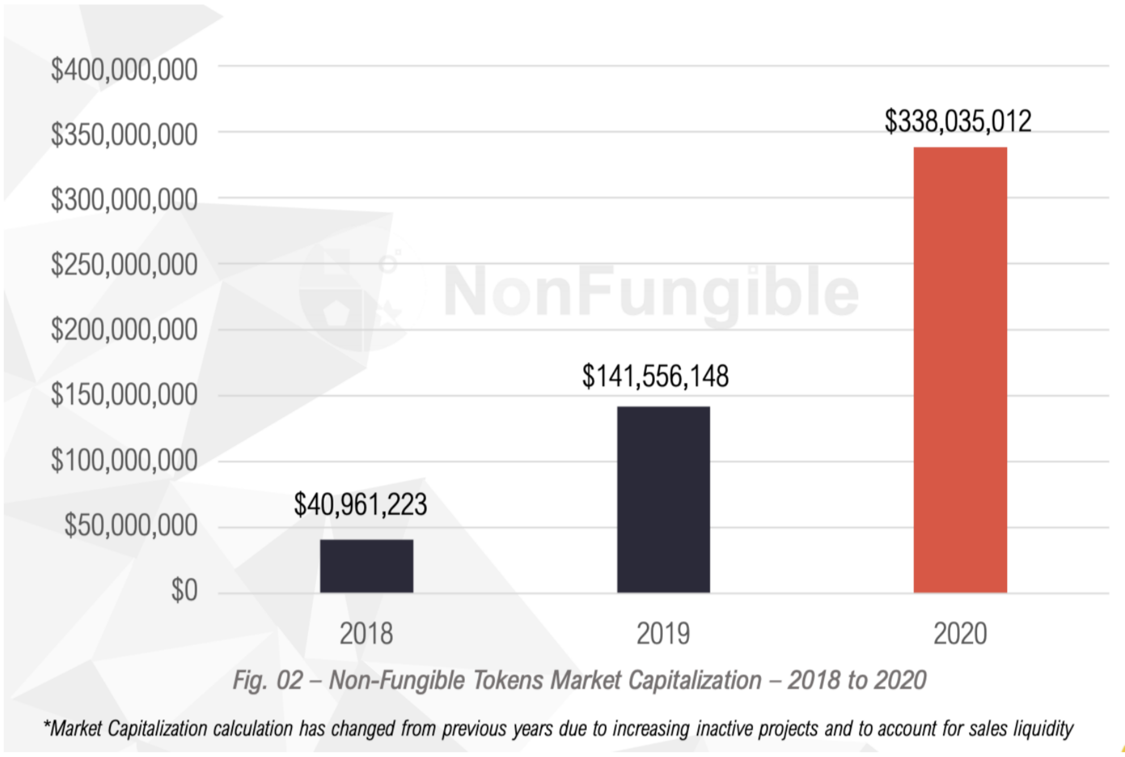


Figure 3.3: NonFungible - NFT market capitalization

3.3 USD traded

Regarding the amount of dollars traded during the past years, we take two key parameters:

- USD from sales

  this type of volume is generated by the transactions made by the buyers and the sellers such as artist or projects.

- USD from dApp

  this type of volume is generated by the interactions with a smart contract. The dollars from dApps are used to improve and modify the project itself, it can be seen as the activity behind each single project.

With respect to the active wallets, we have a peak of usage in the second half of 2020 and they are related to many projects that used NFTs as the main product. For instance, the peak we had in 2020 depends mostly on projects like Decentraland, which is a virtual reality platform powered by Ethereum that was launched to the market in February 2020.

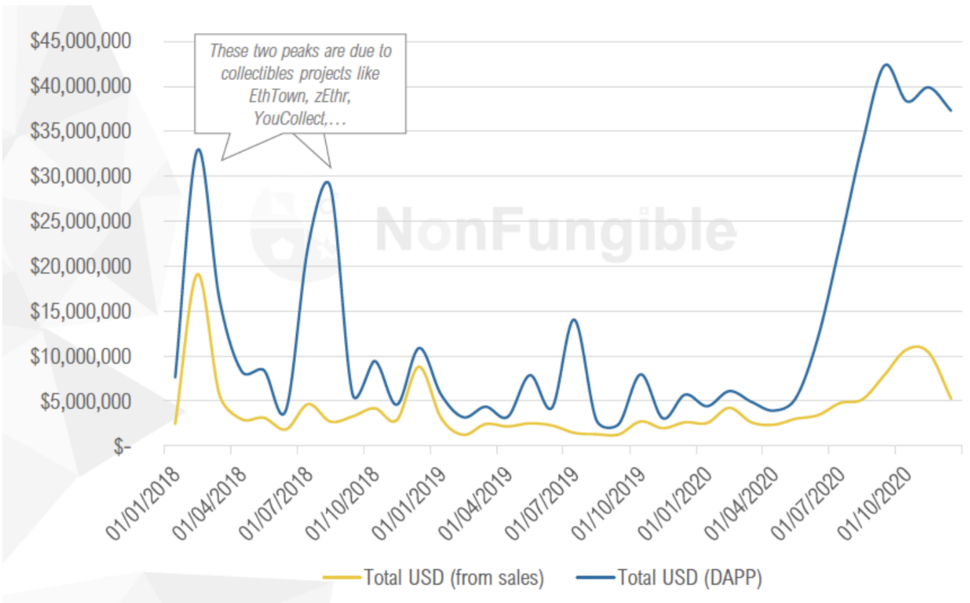


Fig. 05 – USD traded in NFT Sales vs. dApp volumes – 2018 to 2020

Figure 3.4: NonFungible - USD traded in NFT sales vs dApp volumes

The data we have seen so far shows that this sector is reaching a new level of maturity.

It is also important to notice that the gap between dollars from sales and dollars from dApps proves that the use of tokens has been scaled to more people and therefore, the NFTs market is no more for those who do speculation.

3.4 Considerations

The assumption is that this market will grow in the next years since the volume is increasing the hundreds of new projects are raising. To make the analysis more detailed, we need to wait for the data for the year 2021 since it is impossible to say with confidence that we are not inside a speculative bubble or that the interest in these projects is strictly related to the pandemic.

A critical parameter is the loyalty of the users. NonFungible made a survey on their Twitter profile which testifies how the loyalty the users have on their purchases is really high. In particular, there are two different segments, users who want to invest for a long-term profit and users that are in love with it.

Chapter 4

Decentralized Projects Overview

In recent years, many decentralized projects attempted to solve the problems of reduced earnings and the distribution of content in the music industry.

This chapter begins by defining the current solutions available at the time being that supports music creators. Here it is describe the approach of the most important project in the context.

4.1 Pindify

Pindify is a subscription-based marketplace that helps creative providers turn their passion into a source of income. It gives audiences access to exclusive content from their favourite artists. According to Pindify's whitepaper, a small number of fans, who typically represent 10-20% of a provider's user base, can drive 80% or more of the provider's overall business value. Exclusive content and special relationship efforts must prioritize initiatives, specifically aimed at serving them.

Like other user-generated content platforms (such as Youtube and Instagram), listeners are able to publish content to the platform. Pindify uses blockchain technology to track and store the analytics of both streams and views. These data are used to distribute streaming payments of original content to artists.

Content creator accounts enable artists to create and upload original content with optional monetization. "Supporter" accounts enable users to view a selection of content according to their subscription tier. Artists earn JEMS tokens, which can be withdrawn on a weekly basis in fiat, a government-issued currency that is not backed by a physical commodity, such as gold or silver, but rather by the government that issued it [16].

Pindify's token Pindex (PDI) is an ERC-20 token [17] on the Ethereum blockchain. Providers, subscribers and businesses will be then incentivized to use Pindex as a primary currency on the Pindify marketplace for subscription, trading, funding and branding.

With the usage of blockchain, Pindify wants to pay directly the providers involving however other actors, such as businesses and influencers, to promote the creators and their content. In this context, the blockchain will secure identity, authority, trusted payments, ownership and user rights. It is important to say that Pindify does not transact in cryptocurrency. They are based on Stripe, a financial services company that processes payments online.

4.2 Musicoin

Musicoin is a streaming platform that leverages the blockchain to enable Peer-to-Peer (P2P) payments and data storage in a transparent way. Automated P2P payments enforced by smart contracts on the Musicoin blockchain enable fair and transparent distribution of value across all parties, from miners and project developers to musicians and listeners, without any need for outside intermediaries. Moreover, instead of using centralized servers, Musicoin is storing and distributing its content through a decentralized P2P file distribution system known as Inter-Planetary File System (IPFS).

Through Musicoin smart contracts, when content is consumed by a user, the content creator earns a profit. In case the content was created by more people, the earnings are split depending on their shares.

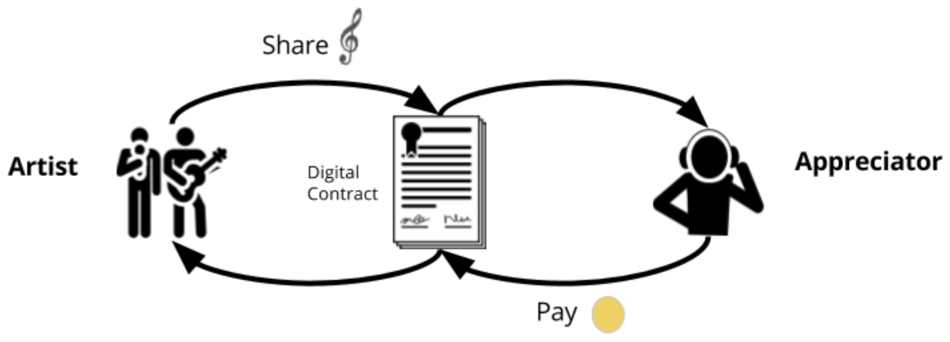


Figure 4.1: Musicoin sharism

Musicon Pay Per Play (PPP) [18] smart contract is designed in such a way that all musicians retain full ownership of their content, and are rewarded fairly and automatically through autonomous smart contracts. This design brings a new level of transparency and clarity to the music industry.

The PPP smart contract pays artists from every single stream of their content. PPP is a smart contract on the Musicoin blockchain that enforces and executes licensing terms to reward a certain fixed amount of $MUSIC (the native currency of the Musicoin platform) per playback. No intermediaries are required.

The system is smooth but it does not take into account that peers store the copyrighted content on their own machine without any level of security. The streaming is still a centralized service provided by Musicon.

During the last years, this project failed due to the lack of concrete tokenomics.

Definition 4.2.1 (Tokenomics)**.** [19] Tokenomics is the science of the token economy. It covers all aspects involving a coin's creation, management, and sometimes removal from a network.

All the artists involved in the project have been overcompensated at the expense of the community. During the writing of this document, Musicoin planned to relaunch a new version of the project in the next months with a proper tokenomics.

4.3 Audius

Audius is a decentralized project that stands close to Musicoin regarding its values and it is one of the most popular and promising projects at the moment. A deeper study into its design has been made.

The mission of Audius is to give everyone the freedom to share, monetize, and listen to any song. The main problems that they want to solve are:

1. There is little transparency on the origins of artists payout.
2. Incomplete rights ownership data often prevents content artists from getting paid[20].
3. There are layers of middlemen and large delays involved in payments to artists.
4. Licensing issues due to the poor management

The Audius protocol allows artists, fans, and node operators to collectively provide a high-quality end-user music streaming experience without a centralized infrastructure.

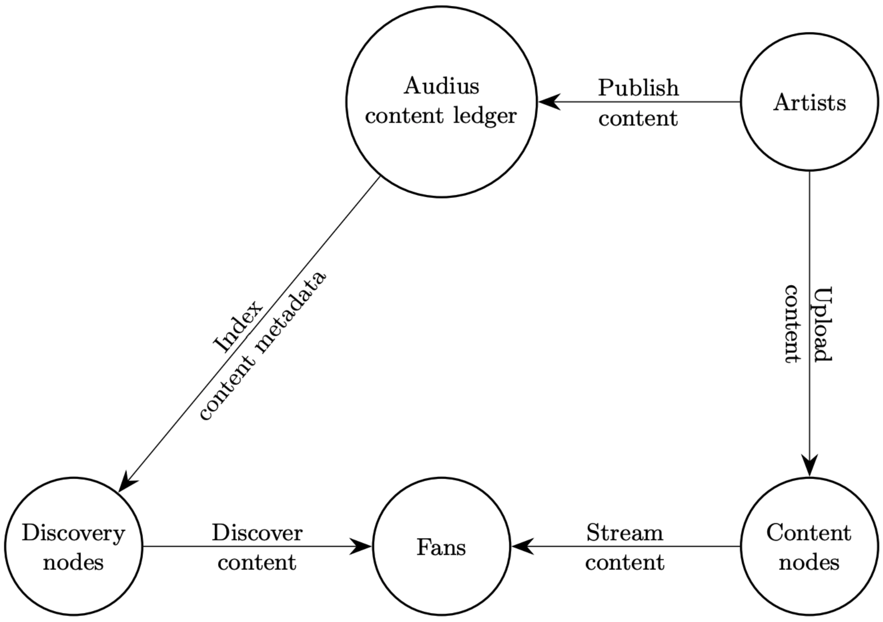


Figure 4.2: Audius architecture. Image taken from Audius's repository

Through the Audius Token[21], $AUDIO, the system is incentivated to collaborate. From the above image, we can spot four main actors in the Audius infrastructure:

1. **Audius content ledger**: The single source of truth in the whole Audius system. It lives on a smart contract and it stores all the metadata of the contents stored inside the content nodes.

2. **Artists**: They can publish their songs directly on the system by controlling the content nodes (storing their music) and updating the Audius content ledger.

3. **Content nodes**: Content nodes maintain availability of content and metadata in Audius. In particular, they use AudSP which is an extension of IPFS. These nodes can be run by node operators alongside an active network stake. In this way, they are given the opportunity to earn part of the ongoing Audius token issuance and aggregated fee pools. A content node can be run by an artist to host their own content.

4. **Discovery node**: The discovery node has a mechanism for indexing metadata. It is used by listeners to find the songs in the application.

4.3.1 Token

Audius platform tokens (ticker $AUDIO) have three prongs of functionality within the protocol unlocked by staking: security, feature access, governance. These tokens are staked as collateral for value-added services. In exchange, stakers earn ongoing issuance, governance weight and access to exclusive features.

Staking is a way of earning rewards for holding certain cryptocurrencies in a network that supports Proof of Stake (PoS) consensus algorithm. A consensus algorithm is a mechanism that allows users or machines to coordinate in a distributed setting and it is used to validate new blocks of a blockchain.

Node operators must stake Audius tokens to operate a discovery node or content node. A larger stake corresponds to a higher probability of being chosen by fan clients. With the use of Audius on-chain metrics, based on the value each actor brings to the network, Audius tokens are received from node operators.

4.3.2 Content node

Content nodes maintain the availability of the content and their metadata in Audius on AudSP, the Audius-native extension to IPFS. AudSP differs from IPFS because AudSP provides a staking-based incentive structure for users to host network content.

The artists can choose to delegate their music to external content nodes in the Audius network or to host their own music. With the second approach, the artists have more control over the distribution of their content by keeping control of content-encryption keys in the infrastructure they control. In this way, they can allow custom permissions extensions that are not native to the protocol. On the other hand, if the content is self-hosted, they need to be sure that their nodes are elected in the network otherwise their content will not be able to be retrieved by participants in the system.

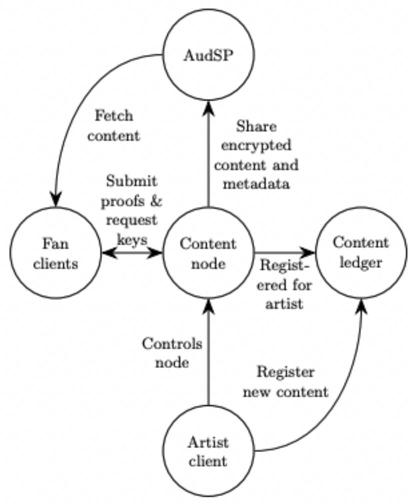


Figure 4.3: Content node interactions with protocol participants

AudSP, custom extension of IPFS, is used to store and deliver content on these nodes. IPFS enables modular object-level encryption, global distribution capability, secure content addressing, and object immutability. In order to ensure high availability for files stored through the Audius protocol, AudSP provides a staking-based incentive structure for users to host network content.

4.3.3 Content Ledger

The Content Ledger is the blend of smart contracts on Ethereum and other blockchain networks that store pieces of Audius. The content ledger also serves as a central source of truth for fan/artist interactions happening within Audius such as:

1. Stream a track
2. Like a track and add it to the users' library

3. Follow other users

4. Repost track to followers

5. Create and share playlists to followers

It is important to say that off-chain solutions are taking into account as solutions to scalability.

4.3.4 Discovery Node

Discovery nodes are used in Audius as a mechanism for indexing metadata that is efficiently queryable by users and discover content on the network.

1. Decentralized

2. Efficient and straightforward for user clients to consume

3. Probably correct and transparent, eliminating profit incentives to manipulate the results returned to users

4. Extensible, so that the Audius community can explore different ranking and searching methodologies

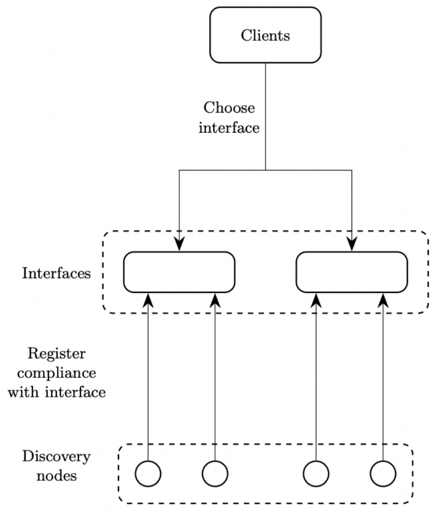


Figure 4.4: Discovery API interface registration and usage. Image taken from Audius's repository

Fan clients select discovery nodes, to query from, via the content ledger's node registry. Discovery node operators earn revenue by registering a node with an active network stake, letting them earn part of the ongoing Audius token issuance and aggregated fee pools. In this case, the main concern in their infrastructure is about the content nodes. In particular, the user can listen to a song when this is available on at least one node. When the user downloads the music, the request is consumed by one server only.

Therefore, the content node becomes a "central server" that serves that song. The reality is that centralized solutions, such as Spotify, are more likely to provide a better experience since they have a solid infrastructure behind them. In Audius, the infrastructure is built on top of nodes. Everyone can join this network but it is unlikely that everyone has high-end machines to support this workload.

Audius lacks a way to distribute the computing power among all the nodes in the network.

Chapter 5

deMusic

In this section, we present, in detail, how the deMusic platform provides an easy way to incentivize artists to produce music and help users to listen to songs through a decentralized system. In particular, deMusic differs from the previous solutions as follows:

1. The computing power of the streaming service is shared and distributed among all the nodes of the network.
2. Artists can distribute their music on the network nodes that scales automatically as the number of listeners scale.
3. The possibility of creating new tokens into the blockchain with which they can produce value, used to fund their career.
4. Tokens are backed by legal contracts that ensure the authenticity and the value of each token generated by each artist.
5. Transaction costs are cut since an artist can upload himself the content into the network and the distribution is taken care by the community itself.

The idea behind this thesis is to help young artists to be funded by their fan base or business angels and help them to grow. The solution described in this chapter will reshape the use of tokens to achieve this goal. Moreover, the artists should be able to distribute their music into a network that ensures no disruption of the service, without intermediaries, cutting the transaction cost to its minimum.

With the help of the blockchain, we are going to provide artists with a tool that allows them to create sets of tokens (fungible and non). These tokens can sold in a marketplace in order to fund their careers. In this way, new assets are generated on-demand and recorded in a public blockchain. This type of token allows the owner to access certain services or wait for them to gain value. The owner of the token is able to sell it back in the marketplace as attempt to gain profit.

For example, a young artist can generate ten NFTs (Non-Fungible Tokens) and assign one autograph each. The user can buy one or more tokens and burn them at any point in time to get the autograph from the artist. It is important to say that each token is directly bound to a legal contract.

A legal contract is needed to ensure that the artist is obliged to do exactly what he wrote into the token he generated. This contract is stored into the IPFS to ensure the integrity and the availability of the document and it will be connected directly to the token by storing the document address. This work does cover the creation of the legal agreements needed to ensure that blockchain tokens are recognized as currency for the services artists offer. At the time being, the development of this part is still in progress due to its sensitivity and it has been delegated to people with the right skills.

From the users' point of view, each token bought becomes an asset. Its value grows depending on the artist's visibility. The same token can be then sold or transferred to other users depending on its current value.

The tokens have a wide range of usage. They can be assigned to autographs, concert tickets, dinners and anything that can be considered valuable for both artist and user.

A streaming service has been also designed to distribute the music among the community. In the next sections, we show all the components needed to build the system. First, it will be described the smart contracts used to generate the tokens and later it will be discussed how the music is streamed into the network with the help of relayers.

It is important to say that all the smart contracts have been developed with *Solidity* (version 0.8.0).

For the purpose of the document, all the smart contracts will be described but my personal contribution in this topic is limited to their design and supervision. Each smart contract has been developed by Lucas Manini.

5.1 Artist Identity

The Artist Identity is a particular smart contract designed to represent the artist's collection of tokens. This component is the identity of the artist that lives inside the blockchain. It is directly managed by the artist itself through its wallet, and not by deMusic.

The Artist Identity enables different scenarios. At first, the artist is able to generate NFTs for its songs. In this way, he has ownership and the ability to split it among several actors. In the same way, he can decide to sell fractions of royalties in a marketplace. Each NFT generated will grant the capability to earn from the streaming service based on the current owner over the asset.

The second functionality is the collection of tokens. This smart contract is designed to generate and keep track of artist's tokens. As explained before, one of the fundamental deMusic is built on is the support of young artists. In particular, each artist can generate tokens and give a particular meaning to each of them. In this way, the artists can sell the tokens to their fan base or investors in order to support their carrier.

An implementation of the Artist Identity can be seen in Appendix A.

5.1.1 ERC-1155 comparison

During the design of the Artist Identity, we faced the problem of which standard to use. The first solution we landed on was ERC-1155. This standard outlines a smart contract interface that can represent any number of fungible and non-fungible token types. Existing standards such as ERC-20 require the deployment of separate contracts per token type. The ERC-721 standard's token ID is a single non-fungible index and the group of these non-fungibles is deployed as a single contract with settings for the entire collection. In contrast, the ERC-1155 Multi Token Standard allows for each token ID to represent a new configurable token type, which may have its metadata, supply and other attributes. It is important to consider that the ERC-1155 supports batch minting, which makes it possible to mint multiple tokens at once, saving on the transaction gas cost.[11]

The drawback brought by this token is the $IERC1155MetadataURI$ which does not suit very well with the customization we want to achieve for each type of token. This URI is used to retrieve all the metadata for a type of token. In our case, each token is unique and has different metadata. In conclusion, we design a factory to create and deploy both ERC-20 and ERC-721.

5.2 Artist Tokens

The Artist Token is essential for deMusic. This component allows the artists to generate digital assets inside the system that are ready to be sold. We built it as a means to easily invest in young artists and support them.

The range of applications is wide. Given the nature of tokens, the artist is free to assign them different meanings, thus value, restricted by the policies of deMusic. The value is kept outside the platform through legal contracts signed by the artist itself and attached to the tokens themselves. Depending on the identity of the token, to keep the platform as decentralized as possible, part of the metadata is stored inside the IPFS (Inter Planetary File System).

Definition 5.2.1 (IPFS)**.** The InterPlanetary File System (IPFS) is a protocol and peer-to-peer network for storing and sharing data in a distributed file system. IPFS uses content-addressing to uniquely identify each file in a global namespace connecting all computing devices. Any user in the network can serve a file by its content address, and other peers in the network can find and request that content from any node that has it using a DHT.

In order to make the tokens, we used two standards well known in the blockchain world: ERC-20 and ERC-721. In the case of the ERC-721, it has been provided the ability to use the tokens through the use of $ERC721Burnable$ extension which enables the burning of the tokens once it is used. In this way, it made sure that one NFT is used only once.

It is important to remember that each token will be backed by a legal contract that ensure that the token is recognized as currency for the services artists offer.

5.3 deMusic Governance

Governance tokens are tokens that developers create to allow token holders to help shape the future of a protocol. Governance token holders can influence decisions concerning the project such as proposing or deciding on new feature proposals and even changing the governance system itself.

In many cases, the changes proposed, vetted and then voted on through on-chain governance accessed by using governance tokens are applied automatically due to smart contracts. Proponents of systems that use governance tokens believe that they allow for user control, which holds to the original cryptocurrency ideals of decentralization and democratization.

In deMusic, the governance is built forking the one of SushiSwap. SushiSwap is a community-based project. It is ultimately governed by its community, via forum discussions and, when pertinent, voting on proposals. SUSHI (SushiSwap native governance token) holders have governance rights over the protocol's future.

This decision of forking SushiSwap governance has been made in order to have well-designed governance that empowers token holders, in this case the community. [22]

5.4 Decentralized Streaming

Before diving into this topic, it is important to say that Tommaso Paulon and I worked closely to develop the following service.

In a world in which big players offer streaming services through the classic client-server architecture, our goal was to build a community where every single user can share and consume content at the same time. Unlike the classic architecture, the advantages of being decentralized are:

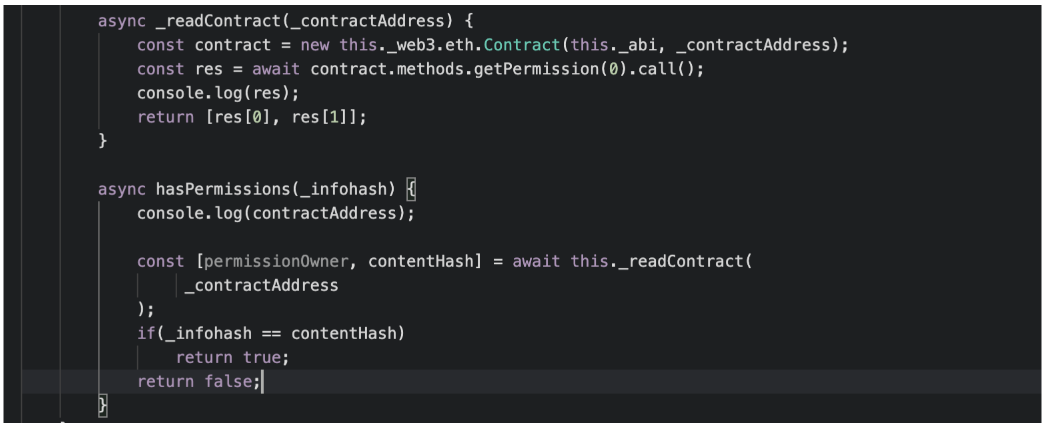
- the effort is split among all the nodes in the system
- high availability
- no sovereignty

Inside deMusic, we designed the decentralized music streaming service with the use of BitTorrent protocol. In this way, besides the advantages previously listed, we minimize the load on each node and maximize the throughput.

To make all of this possible, the first concern was building a BitTorrent client which could work on every machine regardless of which operating system was running on top. None of the solutions found was suitable since each of them needed to be compiled for each operative system. To solve our problem, we used the WebTorrent Client.

WebTorrent is a peer-to-peer (P2P) streaming torrent client written in JavaScript by Feross Aboukhadijeh and the team at WebTorrent and on GitHub, for use in web browsers, as well as a WebTorrent Desktop stand-alone version able to bridge WebTorrent and BitTorrent serverless networks. The idea behind WebTorrent is to make a BitTorrent-like protocol that works on the web browser, maintaining as much compatibility with BitTorrent as possible. [23]

In deMusic, two fundamental requirements need to be met in order to start storing and streaming music: each seeder has to be able to check peer's permissions questioning the latest state of the blockchain and each play has to reward both the effort of the seeder and the owner(s) of the song. To do so, we developed a new extension of BitTorrent Protocol, named deMusic Protocol, which allows the seeder to check the peer's permission through its public key inside the smart contract that rules the system, during the handshake.



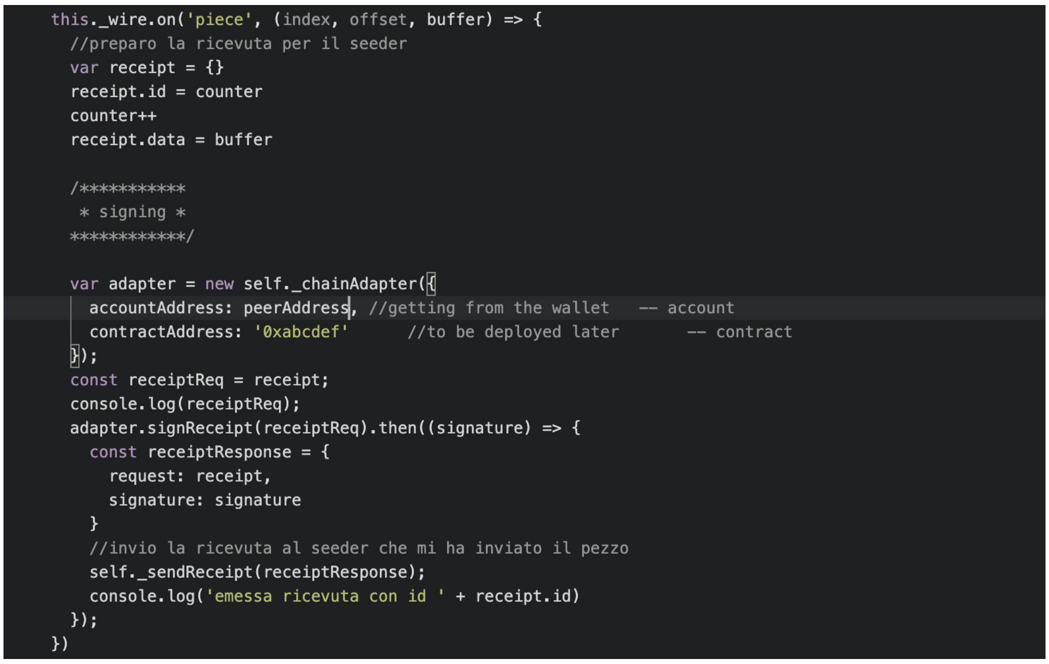
```
async _readContract(_contractAddress) {
    const contract = new this._web3.eth.Contract(this._abi, _contractAddress);
    const res = await contract.methods.getPermission(0).call();
    console.log(res);
    return [res[0], res[1]];
}

async hasPermissions(_infohash) {
    console.log(contractAddress);

    const [permissionOwner, contentHash] = await this._readContract(
        _contractAddress
    );
    if(_infohash == contentHash)
        return true;
    return false;
}
```

Figure 5.1: chainAdapter.js

The second and the most important requirement is that the streaming service needs to reward both the seeder who practically streams the music and the content owner. In order to meet this prerequisite, we made up a system of receipts.

A receipt is a proof of work that is sent by the seeder and signed by the peer. In this way, the seeder can testify the effort it put inside the transaction and which content was delivered to whom. Considering each actor untrusted, the seeder waits for a signed receipt every time it sends a chunk of the content. In this way, we mitigate the event that the user consumes the content without sign the receipt at the end.

It follows the piece of code, strictly connected to the previous one, in which we enabled the signing process.



```
this._wire.on('piece', (index, offset, buffer) => {
  //preparo la ricevuta per il seeder
  var receipt = {}
  receipt.id = counter
  counter++
  receipt.data = buffer

  /***********
   * signing *
  ************/

  var adapter = new self._chainAdapter({
    accountAddress: peerAddress, //getting from the wallet   -- account
    contractAddress: '0xabcdef'      //to be deployed later       -- contract
  });
  const receiptReq = receipt;
  console.log(receiptReq);
  adapter.signReceipt(receiptReq).then((signature) => {
    const receiptResponse = {
      request: receipt,
      signature: signature
    }
    //invio la ricevuta al seeder che mi ha inviato il pezzo
    self._sendReceipt(receiptResponse);
    console.log('emessa ricevuta con id ' + receipt.id)
  });
})
```

Figure 5.2: ut_ receipt.js

It is important to say that the communication between the seeder and peer has been enhanced. It is now possible to send all the information required, such as the public key and signed receipt.

5.4.1 Relayer

A Relayer is a computer that stores chunks of songs and helps the network to stream music inside the platform. This actor is key for the system since it lends computing power and storage to make streaming possible. In exchange, for each chunk the relayer helped to stream, the user signs a proof of work that can be used by the relayer itself to redeem a reward from deMusic.

The Relayer is the node that stores music and embodies the role of the seeder. To accomplish a decentralized solution, several relayers are needed to serve the contents to the entire community.

In order to achieve zero downtime, we designed a system in which the relayers are rewarded by redeeming the receipts earned. In this way, relayers get recompensed with deMusic tokens with which they are able to actively participate in platform decisions or exchange them with stablecoins such as USD Tether and TrueUSD. A stablecoin is a type of cryptocurrency which aims to keep cryptocurrency valuations stable.

It is important to say that the Relayer does not store the whole song. In order to mitigate piracy, each song will be divided into chunks and encrypted. In this way, each Relayer won't have all the chunks but a subset of them. This feature is not yet developed since a further investigation is needed.

Chapter 6

Evaluation

In this chapter we show the two fundamental metrics we analyzed to ensure the project was scaleable and feasible in terms of cost and overhead. Both metrics are paramount since they are beared by the community, both artists and users.

6.1 Infrastructure scalability

The use of BitTorrent protocol to enable the streaming service opens the possibility to scale as much as the people use the network. After developing the BitTorrent protocol extension, with which we enabled music streaming service based on permissions in the blockchain, we ran several tests with physical machines and a simulated environment with more than fours peers. Two files were used as means to measure the downloading speed: an image of 84KB and a song (copyrighted-free) of 7.2MB.

Firstly, we ran the test with three physical machines over the internet where we had only one seeder and two leechers (peers that consume the content). In this environment, we did not see a tangible improvement. Hence we used Docker, a set of the platform as a service (PaaS) products that use OS-level virtualization, to increase the number of peers in a simulated environment inside a single physical machine. In particular, we noticed how tangible improvements of the overall downloading speed were achieved with two seeders and two leechers.

The result of the simulation showed how the workload for the seeders was the same in a 1:1 peers connection, the same we could have in a simple architecture client-server. It is important to keep in mind that, with the same workload, shared among the two seeders, we provided the content to the two leechers without increasing the time to download it. Hence we duplicated the upload bandwidth inside the network.

Another simulation was analyzed in which we had three peers, two seeders and one leecher. In this case, we experienced how the workload was distributed and split among the two seeders. The overhead for each seeder was almost half while the leecher consumed the content in the same amount of time as the last simulations.

It is important to say that, since we were on the same machine, the upload bandwidth was not affected by the internet traffic. If we had run the last simulation over the internet, we could have experienced two different scenarios. In the first case, we could have the same result we had. The workload of the seeders is split in (almost) half, having the leecher being the bottleneck in the connection. In the second case, we can have that the two seeders are the bottleneck in the system. In this scenario, the leecher is downloading with a network upload bandwidth which is the sum of each seeder's upload bandwidth.

x(t)	number of downloaders in the system at time t
y(t)	number of seeds in the system at time t
λ_0	the initial value of peer arrival rate
τ	the attenuation parameter of peer arrival rate
μ	the uploading bandwidth
c	the downloading bandwidth ($c \gg \mu$)
γ	the rate at which seeds leave the system
η	the file sharing efficiency, meaning the probability that a peer can exchange chunks with other peers

According to the study [24], the average of downloading speed of peers at time t is:

$$\mu(t) = \mu \frac{\eta x(t) + y(t)}{x(t)} = \mu(\eta + \frac{y(t)}{x(t)})$$

which roughly converge with the results we achieved running our simulations. To better converge with this model, a higher number of peers are needed.

An important notion to highlight is that an average torrent lifespan, the duration in which at least a couple of seeder and leecher is present in the network ensuring the downloading of the content, is **about 8.89 days** [25]. As soon as no couple is in the network, the torrent is considered dead and the content is lost. However, in our design, this is not possible since the relayers provide a minimum number of seeders always available.

6.2 Blockchain costs

Inside an EVM-compatible blockchain (We will see what an EVM is in a moment), storing a smart contract and execute a transaction are actions with a cost.

An Ethereum Virtual Machine (EVM) is a particular virtual machine, hosted inside every node in the blockchain and it is used to execute actions that change the state of the blockchain, such as storing data or executing a transaction. The EVM is used by each node to compute the new state by providing as inputs the current state of the blockchain and the new transaction. All the nodes will eventually generate the same new state. To mitigate the case in which the nodes loop forever in the same state - we can think of a smart contract with a "while true" loop, each transaction is executed with a cost. The cost, calculated in *gas*, is associated with each statement the code runs in the EVM and it is paid in GWEI, also called nanoether, which is $10^{-9} ETH$.

Therefore, to execute the entire transaction, the owner needs to bear all its costs. For instance, let's take for example the transfer of a value from a wallet to another. The amount of gas needed to execute it is 21000. To ensure that the transaction is attached in a block within fives minutes, the current gas price to be paid is 64 GWEI/gas. Hence, the amount of GWEI needed to be paid for the transaction is

$$64 \, \frac{GWEI}{Gas} \times 21000 \, Gas = 1'344'000 \, GWEI$$

The gas price changes based on the current volume of transactions in the network. In this example, we did not take into consideration and explained how base fees and priority fees work because it is slightly out of scope and we should introduce the old system for the gas price and what changed after the London hard fork in Ethereum.

Thus it is important to remember that the amount of GWEI beared by the owner of the transaction is

$$gasprice \times gas = gwei$$

.

In this project, we had to face the problem of choosing a blockchain in which the community should operate. The main concern is that both artists and users should operate in a cheap blockchain in order to ensure that most of the value is focused on funding the artists' careers.

In this regard, we chose to deploy our service in Polygon. The latter is a side-chain, a separate blockchain that is attached to Ethereum using a two-way peg. The two-way peg enables the interchangeability of assets between the parent blockchain and the sidechain. Polygon is fast and cheap with an average block time of two seconds and it is powered by MATIC.

As we previously said, a GWEI is $10^{-9}ETH$ or $10^{-9}MATIC$ depending on the blockchain we are operating. To better understand the difference of cost, in terms of dollars, for using either Polygon or Ethereum, we can take the previous transaction under consideration. In Ethereum, the cost of the transaction would be:

$$10^{-9}\,\frac{ETH}{gwei}\;\times\;1'344'000\,gwei\;\times\;3183\,\frac{USD}{ETH}=4.25\,USD$$

The cost of 1 ETH is 3183 dollars at the moment this chapter has been written. Since this value is volatile, the amount of dollars, needed to execute the transaction, changes accordingly.

In Polygon, the MATIC is equal to 1.417 dollars a the moment this chapter has been written, so the cost for the same transaction would be:

$$10^{-9}\,\frac{MATIC}{gwei}\;\times\;1'344'000\,gwei\;\times\;1.417\,\frac{USD}{MATIC}=0.00189\,USD$$

With the value we just calculated, it is easy to understand how operating in Polygon is much cheaper than doing the same things in Ethereum. In this way, we ensure that artists can generate profit from the tokens they generate and the users are not afraid of purchasing in this environment.

Chapter 7

Evolution Of The Project

The infrastructure we want to achieve is paramount to be discussed. In this chapter, the focus will be addressed to the evolution of the system that will carry the entire project. In this way, we aim to provide the reader with both technical and business aspects.

The next sections will cover the entire project starting from the POC, Proof Of Concept, and arriving at the final blueprint. It is important to point out that, during the drafting of this document, the project has almost reached the stage of being an MVP, Minimum Viable Product.

7.1 Proof Of Concept

After taking charge of the project, analysis has been taken in place to ensure its feasibility. The crucial component of the entire design was the streaming service in a decentralized system and how it can be used based on the users' permissions.

In this phase, Tommaso and I worked closely to ensure that everything was working correctly.

In order to check permissions, several steps have been followed. First, we built a P2P on top of *WebTorrent* in which each user can be both the relayer and the listener. In this regard, we achieved the streaming of music between two different tabs in the same browser without having a server in the middle. We also have obtained the same results with different computers over the internet using BitTorrent trackers and DHT, previously explained.

In order to simulate an environment with more the two peers, we use Docker containers. In particular, we verified that peers were sharing the same file with other peers that didn't have it.

Definition 7.1.1 (BitTorrent tracker)**.** A BitTorrent tracker is a special type of server that assists in the communication between peers using the BitTorrent protocol.

The second step we took was to enable the possibility to check for permissions inside the blockchain during the handshake of peers before the streaming. In order to do so, we built a testing environment using Ganache by Truffle which simulates a working blockchain environment. After implementing a smart contract able to store permissions, we developed an adapter, used by the P2P client, to access the blockchain and check permissions. In this way, each client can use their own wallet as proof of identity. With this method, it can be easily checked that the peer has the right permission to access specific content.

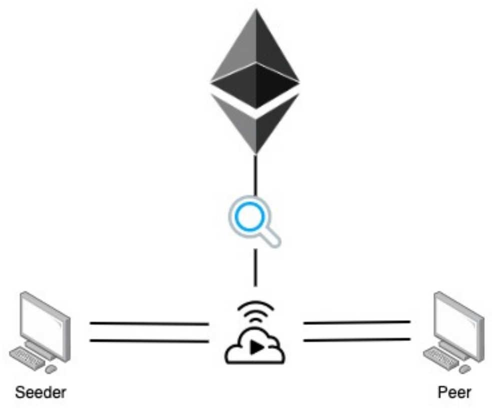


Figure 7.1: PoC infrastructure

A third step has been taken in order to pay relayers based on their performance. To do so, we designed a receipts scheme. The relayers earn a receipt signed by the peer per each chuck transmitted as proof of their work. In this case, relayers can then easily redeem values directly from the users, based on their effort.

It is important to say that the testing environment has been built on top of the Docker engine. Hence, we have tested easily the interaction of peers and how it was scaling.

7.2 Minimum Viable Product

After accomplished the streaming service in a blockchain context, the next step was to build the MVP. The goals we wanted to achieve are, on one hand, to develop a platform that brings value to artists and user friendly. On the other hand, we need to build a solid foundation for the project as a stepping stone for potential investors.

The product we want to promote will have the following features:

- Create User profile
- Create Artist profile
- Artist can upload music and generate NFTs
- Artist can create crowdfunding campaigns to promote their works
- User can purchase NFTs, inside or outside crowdfunding campaigns
- User can transfer their own assets
- Users and Artists can access the marketplace inside the app in order to buy and sell assets

- User can listen music directly inside the app

In this regard, we designed an infrastructure to sustain the load of such activities. With the support of KNOBS, the infrastructure will take place inside their server for the entire duration of this phase and scale it out in case of need with AWS or Google Cloud Platform.

It is important to notice that the infrastructure can easily scale maintaining low costs since the overhead of the streaming service is shared within the community, given the P2P blueprint we realized.

During this phase, I worked closely with Tommaso and Lucas under the supervision of BCode and KNOBS. In particular, my contribution in this phase is focused on the development of the backend application and the development of the infrastructure. The term "infrastructure" refers to the SysAdmin (System Administrator) and DevOps operations done to host several services such as the database, both frontend and backend applications and the pipeline used to enable the Continuous Integration and the Continuous Delivery (CICD) from our repositories and the production server.

Definition 7.2.1 (DevOps)**.** DevOps is a set of practices that combines software development (Dev) and IT operations (Ops). It aims to shorten the systems development life cycle and provide continuous delivery with high software quality.[1] DevOps is complementary with Agile software development; several DevOps aspects came from the Agile methodology.

A testing environment is still in development while writing this document.

For the purpose of the product, we designed a containerized backend application, running on top of Node.js. This layer has the goal to stream music, as it will be the first relayer, and to store and provide the client with the information needed. We added Redis as a cache system. In this way, we want to speed up the process of retrieving data and unload the computational effort of the application.

Definition 7.2.2 (Redis)**.** Redis is an in-memory data structure store, used as a distributed, in-memory key–value database, cache and message broker, with optional durability.

Concerning the data we will store, we prepared two different databases: on one hand, we have a NoSQL database to store users and music data and, on the other hand, we prepared a SQL database to store data that will be used for further analysis and improve the overall experience for both artists and users. Later, we will discuss how data will be managed to ensure privacy and security.

Most of the effort is addressed to the client. For the end user we designed a simple but powerful application that uses all the components we discussed in the previous chapter. In particular, it can access deMusic smart contracts, create and sign transactions and store NFTs both purchased and generated.

Moreover, it enables the possibility to listen and stream music directly inside the application without taking further installation steps.

Here it is possible to see an overview of the entire infrastructure.

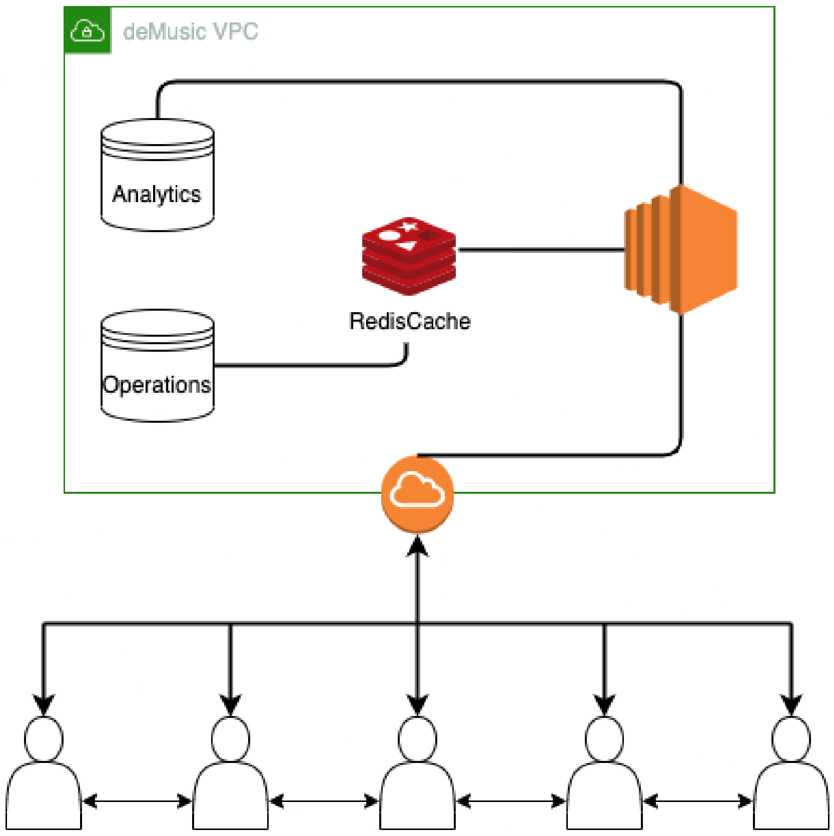


Figure 7.2: deMusic infrastructure

During the drafting of this document, we did not add yet the marketplace and the crowdfunding platform. We are constantly working with BCode to develop these platforms and integrate them as soon as they are ready for production. However, we plan to add these features in the fall of this year.

7.2.1 Data Privacy

Data privacy is core for deMusic and, in general, for the blockchain community. To ensure that the data are readable only by the owners, we designed the database such that all the data are encrypted using asymmetric cryptography. In particular, each data we store is referred to only one wallet, hence a user inside the platform. Hence it is possible to encrypt the data with the public key such that the only actor who has the private one is able to access and modify the data inside the platform.

Chapter 8

Conclusions

In this thesis work, we have presented deMusic, a solution that aims to improve the career of young artists and ease the distribution of their content. The proposed solution is a new decentralized platform where artists can distribute their music and be funded by their fans base and investors. Without any intermediates between the creation of the content and the distribution, the transaction costs and the launch time are minimized.

We started by describing the problems that affect young artists and how their earnings are reduced and delayed due to the number of intermediates they need to distribute their content. Moreover, we discussed royalties. Due to the large number of music being generated each day, a centralized database is unfeasible to be developed in order to keep track of each song and royalties.

An analysis has been made to highlight how the target market grew in recent years in terms of both volume and value, and how the adoption of the blockchain is increasing, both for cryptocurrencies and NFTs.

After quick considerations about why we chose a decentralized solution over a centralized one, we introduced the technologies we used to achieve a decentralized and distributed solution. In particular, we explained how the blockchain and the BitTorrent protocol work and how they can be useful in our situation.

We explored the current decentralized solutions already present on the market and which problems they try to solve. We analyzed their structure and how they work. In particular, a deeper analysis has been made on Audius, a decentralized project that tries to solve one of the problems we are going to solve in our thesis work, which is the distribution of music content.

We showed deMusic, a decentralized and distributed platform that helps artists to generated tokens and distribute their content easily to the community. Moreover, we explored how the use of tokens can be useful to fund artists' careers. Each token refers to exclusive content, generated by the artist, that can be sold to other users. Since each token is generated on a public blockchain, a decentralized and distributed ledger, its value is recognized by the community as a unique element. To ensure that its value remains consistent over time, a legal contract has been theorized to back each token and it is stored in the IPFS, a peer-to-peer hypermedia protocol designed to preserve data so that the document can be accessed easily. With the use of tokens, it is also possible to generate an NFT, a non-fungible token, that refers to a particular song. In this type of token is possible to attach all the metadata needed to recognize a particular song, such as its hash and title, and the distribution of their ownership. In this way, we can achieve an up-to-date and easy-to-access database in which everyone can make queries and check the status of each song.

We developed a new BitTorrent protocol extension that is able to question the blockchain and stream music to peers depending on their permissions. The streaming workload is distributed among the community through relayers in order to scale the infrastructure as the community and the users grow in number. With the help of Lucas Manini, we designed smart contracts, simple programs that can be stored in the blockchain with which artists are able to create, generate and sell tokens. In this way, they are able to fund their career with the help of their fans and investors.

Once we built all the components needed, we decided to launch the project on the market. In order to do so, we built a POC, Proof Of Concept. The latter consists of two applications deployed in a physical server that mock the behaviour we should have in production. In particular, we built a backend application in a NodeJS environment that stores the main information regarding artists who joined the platform. In this way, they are able to create a presentation that is displayed on the frontend application. The latter is a Progressive Web Application (PWA) with which artists and users can interact easily with the blockchain, generate and buy tokens. Even if the music streaming is feasible, we decided to not introduce this feature already due to the copyright and piracy problems. Before bringing this feature to the users, we need to further investigate how the content can be stored securely into the IPFS and shared with the community without any problem. For the duration of the POC, the blockchain in which we deploy the smart contracts and the tokens are the testnet of Polygon.

It is important to say that legal contracts are not yet introduced since they need to be further developed to ensure the right recognition of the tokens.

8.1 Future Works

In this section, we outline some of the future works for deMusic and we provide insight on some of the additional features which would improve on this work. First, deMusic has been designed to speed up the distribution of the content and have an infrastructure, ready to be used, where the creators gain value from their work. In particular, the very next step would be to develop a special smart contract that ensures musicians are fairly paid per each play of their content. The earnings will be split based on the royalties for that specific content. Secondly, we see the additional use of this platform as a marketplace for concert tickets. In particular, our focus would be to remove scalping in ticketing by developing a marketplace in which artists and promoters can sell concert tickets backed by the blockchain. The use case of a user selling a ticket back into the marketplace can be designed.

A final thought is related to the use of this platform for other types of content that differ from music. We strongly believe that, once the platform is ready, we can reuse it to both stream content and provide value to the creators and the community. Generally, we can help talents to be funded by both early investors and their fans base.

Bibliography

[1] Il Sole 24 Ore, “Musica, la discografia è in salute: mercato globale in crescita, tiene l’Italia.”

[2] Tunedly, “10 Unknown Struggles of Being a Musician.”

[3] Wikipedia, “Record Label or Record Company.”

[4] K. Flynn, “How Spotify consumption has changed during the pandemic,” 2020.

[5] C. G. G. P. . Solutions, “PUTTING THE BAND BACK TOGETHER,” 2018.

[6] Investopedia, “Royalty.”

[7] S.I.A.E., “Quote minime di distribuzione.”

[8] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System.”

[9] OpenZeppelin, “ERC20.”

[10] OpenZeppelin, “ERC721.”

[11] OpenZeppelin, “ERC1155.”

[12] Wikipedia, “The Pirate Bay.”

[13] Wikipedia, “Schema Magnet URI.”

[14] Cointelegraph, "How has the COVID-19 pandemic affected the crypto space? Experts answer."

[15] NonFungible.

[16] Investopedia, "Fiat Money."

[17] Pindify, "Pindify whitepaper."

[18] Musicoin, "Musicoin whitepaper."

[19] Decrypt, "What is Tokenomics?."

[20] T. Verge, "METADATA IS THE BIGGEST LITTLE PROBLEM PLAGUING THE MUSIC INDUSTRY."

[21] Audius, "Audius whitepaper."

[22] Sushi, "Sushi documentation."

[23] Wikipedia, "WebTorrent."

[24] Z. X. E. T. X. D. Lei Guo, Songqing Chen and X. Zhang, "Measurements, Analysis, and Modeling of BitTorrent-like Systems."

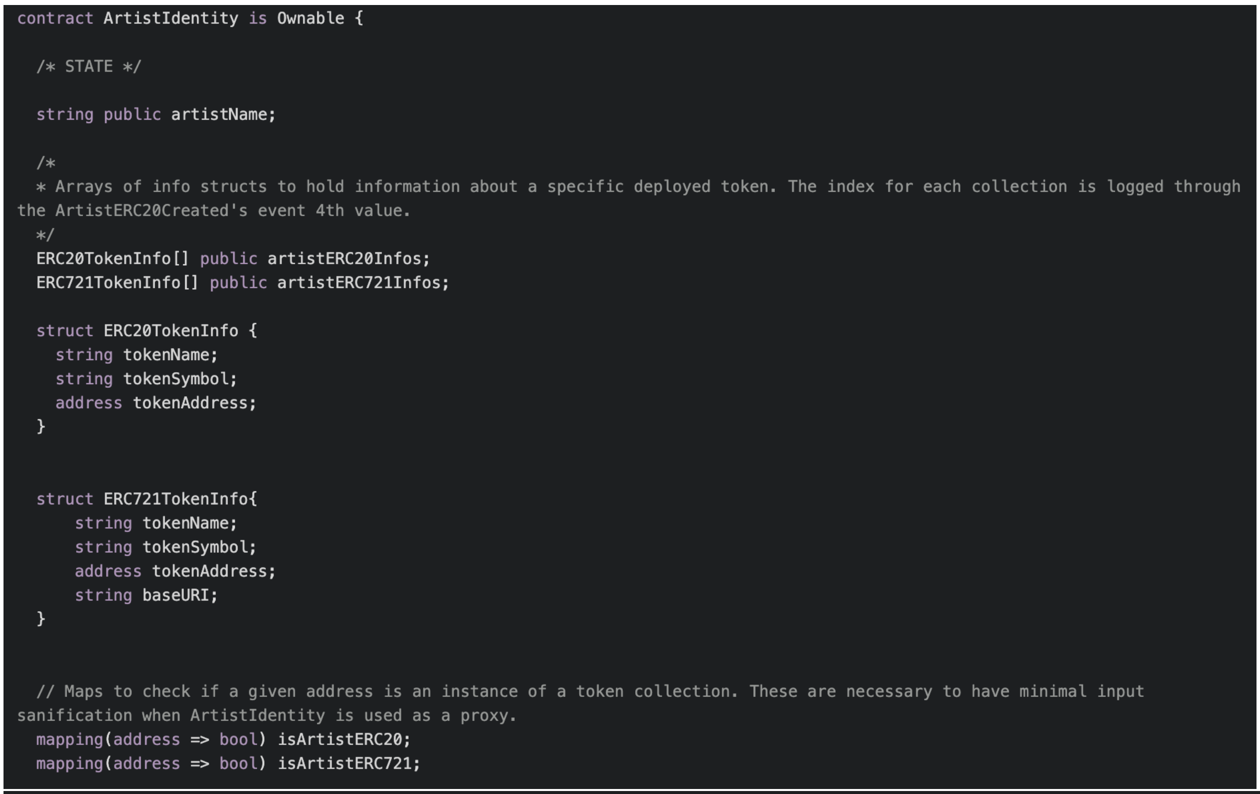
[25] S. B. Jahn Arne Johnsen, Lars Erik Karlsen Sebjørn, "Peer-to-peer networking with BitTorrent."

Appendix A

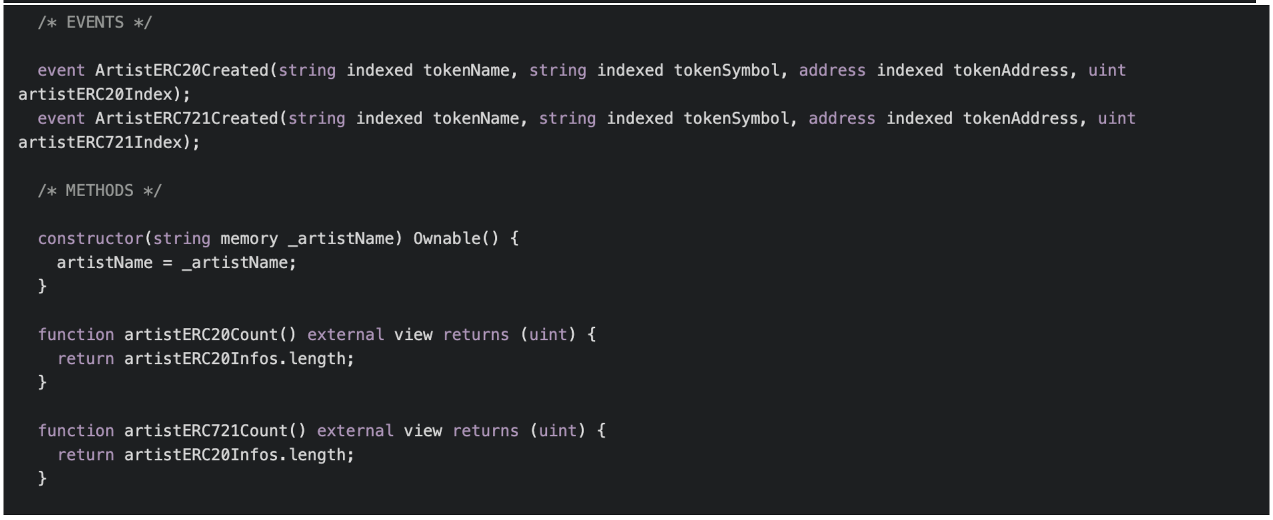
deMusic Artist Identity

This appendix shows the Artist Identity used in deMusic. The goal is to generate the tokens for an artist that can be ready to be sold to his fan base.

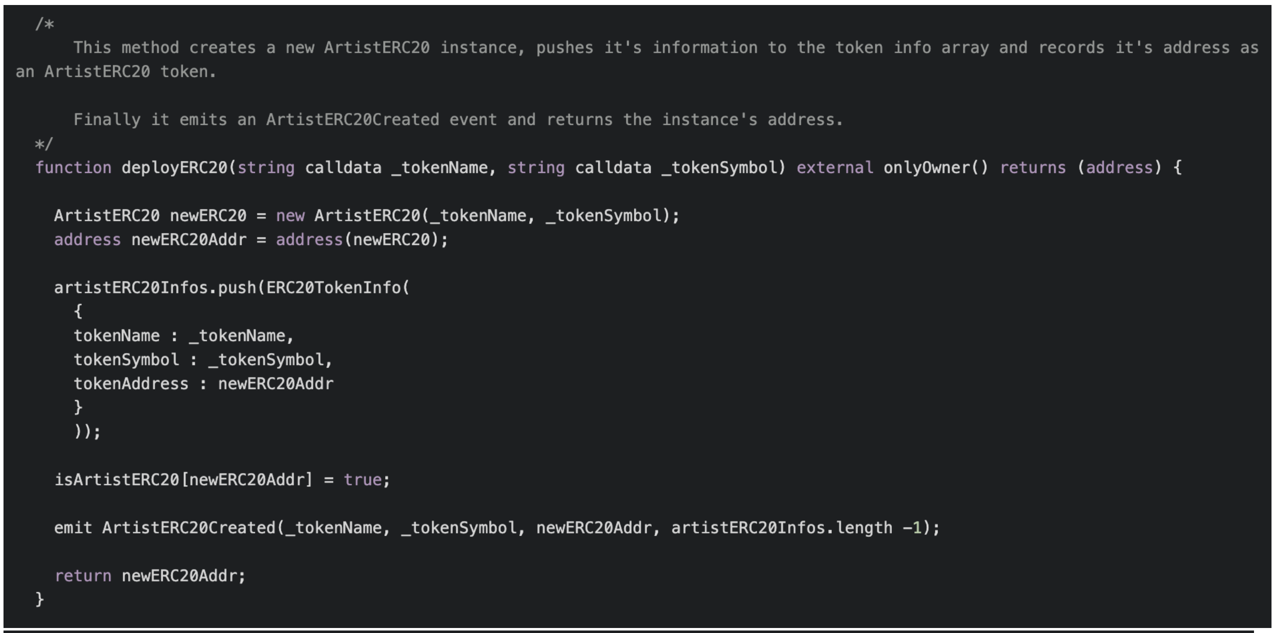
Here it will be possible to have a look at the code of the smart contract and how is it able to keep track and deploy both fungible tokens (ERC-20) and non-fungible tokens (ERC-721).[9][10]



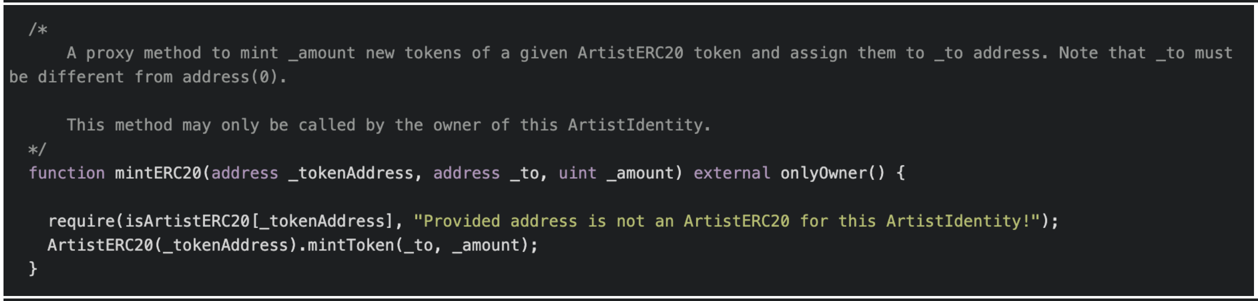
```
contract ArtistIdentity is Ownable {

  /* STATE */

  string public artistName;

  /*
  * Arrays of info structs to hold information about a specific deployed token. The index for each collection is logged through
the ArtistERC20Created's event 4th value.
  */
  ERC20TokenInfo[] public artistERC20Infos;
  ERC721TokenInfo[] public artistERC721Infos;

  struct ERC20TokenInfo {
    string tokenName;
    string tokenSymbol;
    address tokenAddress;
  }


  struct ERC721TokenInfo{
      string tokenName;
      string tokenSymbol;
      address tokenAddress;
      string baseURI;
  }


  // Maps to check if a given address is an instance of a token collection. These are necessary to have minimal input
sanification when ArtistIdentity is used as a proxy.
  mapping(address => bool) isArtistERC20;
  mapping(address => bool) isArtistERC721;
```



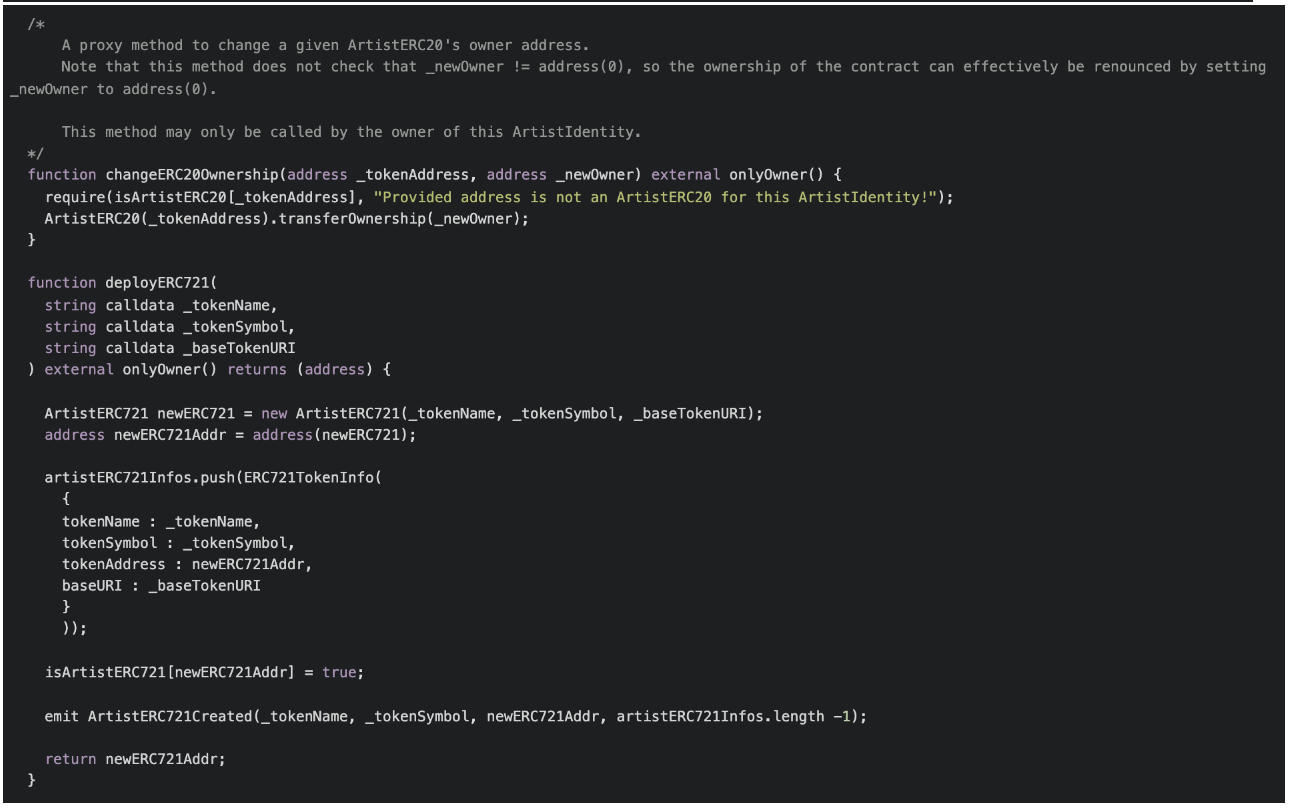
```
  /* EVENTS */

  event ArtistERC20Created(string indexed tokenName, string indexed tokenSymbol, address indexed tokenAddress, uint
artistERC20Index);
  event ArtistERC721Created(string indexed tokenName, string indexed tokenSymbol, address indexed tokenAddress, uint
artistERC721Index);

  /* METHODS */

  constructor(string memory _artistName) Ownable() {
    artistName = _artistName;
  }

  function artistERC20Count() external view returns (uint) {
    return artistERC20Infos.length;
  }

  function artistERC721Count() external view returns (uint) {
    return artistERC20Infos.length;
  }
```



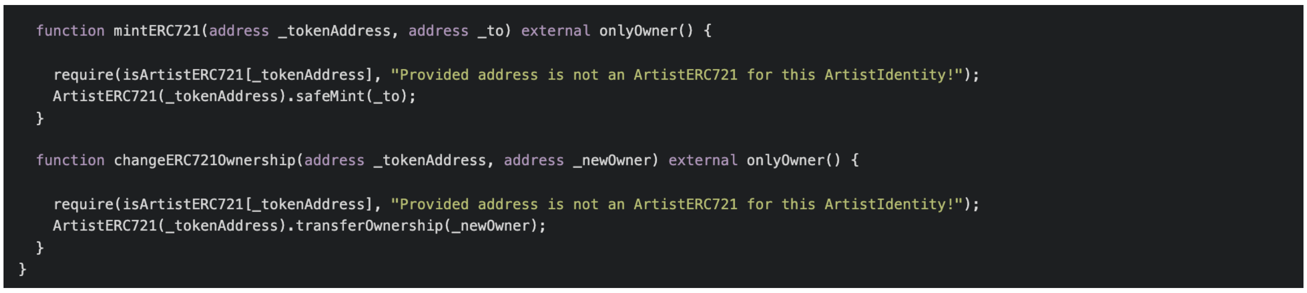
```
  /*
      This method creates a new ArtistERC20 instance, pushes it's information to the token info array and records it's address as
an ArtistERC20 token.

      Finally it emits an ArtistERC20Created event and returns the instance's address.
  */
  function deployERC20(string calldata _tokenName, string calldata _tokenSymbol) external onlyOwner() returns (address) {

    ArtistERC20 newERC20 = new ArtistERC20(_tokenName, _tokenSymbol);
    address newERC20Addr = address(newERC20);

    artistERC20Infos.push(ERC20TokenInfo(
      {
      tokenName : _tokenName,
      tokenSymbol : _tokenSymbol,
      tokenAddress : newERC20Addr
      }
      ));

    isArtistERC20[newERC20Addr] = true;

    emit ArtistERC20Created(_tokenName, _tokenSymbol, newERC20Addr, artistERC20Infos.length -1);

    return newERC20Addr;
  }
```



```
  /*
      A proxy method to mint _amount new tokens of a given ArtistERC20 token and assign them to _to address. Note that _to must
be different from address(0).

      This method may only be called by the owner of this ArtistIdentity.
  */
  function mintERC20(address _tokenAddress, address _to, uint _amount) external onlyOwner() {

    require(isArtistERC20[_tokenAddress], "Provided address is not an ArtistERC20 for this ArtistIdentity!");
    ArtistERC20(_tokenAddress).mintToken(_to, _amount);
  }
```



```
  /*
      A proxy method to change a given ArtistERC20's owner address.
      Note that this method does not check that _newOwner != address(0), so the ownership of the contract can effectively be renounced by setting
_newOwner to address(0).

      This method may only be called by the owner of this ArtistIdentity.
  */
  function changeERC20Ownership(address _tokenAddress, address _newOwner) external onlyOwner() {
    require(isArtistERC20[_tokenAddress], "Provided address is not an ArtistERC20 for this ArtistIdentity!");
    ArtistERC20(_tokenAddress).transferOwnership(_newOwner);
  }

  function deployERC721(
    string calldata _tokenName,
    string calldata _tokenSymbol,
    string calldata _baseTokenURI
  ) external onlyOwner() returns (address) {

    ArtistERC721 newERC721 = new ArtistERC721(_tokenName, _tokenSymbol, _baseTokenURI);
    address newERC721Addr = address(newERC721);

    artistERC721Infos.push(ERC721TokenInfo(
      {
      tokenName : _tokenName,
      tokenSymbol : _tokenSymbol,
      tokenAddress : newERC721Addr,
      baseURI : _baseTokenURI
      }
      ));

    isArtistERC721[newERC721Addr] = true;

    emit ArtistERC721Created(_tokenName, _tokenSymbol, newERC721Addr, artistERC721Infos.length -1);

    return newERC721Addr;
  }
```



```
  function mintERC721(address _tokenAddress, address _to) external onlyOwner() {

    require(isArtistERC721[_tokenAddress], "Provided address is not an ArtistERC721 for this ArtistIdentity!");
    ArtistERC721(_tokenAddress).safeMint(_to);
  }

  function changeERC721Ownership(address _tokenAddress, address _newOwner) external onlyOwner() {

    require(isArtistERC721[_tokenAddress], "Provided address is not an ArtistERC721 for this ArtistIdentity!");
    ArtistERC721(_tokenAddress).transferOwnership(_newOwner);
  }
}
```

Figure A.1: ArtistIdentity.sol file - identity of an artist inside the blockchain

The latter has been developed in order to keep track and generate all the tokens the artist needs. It is possible to see that, each time it deploys a token, it saves its information in a struct and store it inside a list, depending on the token type.

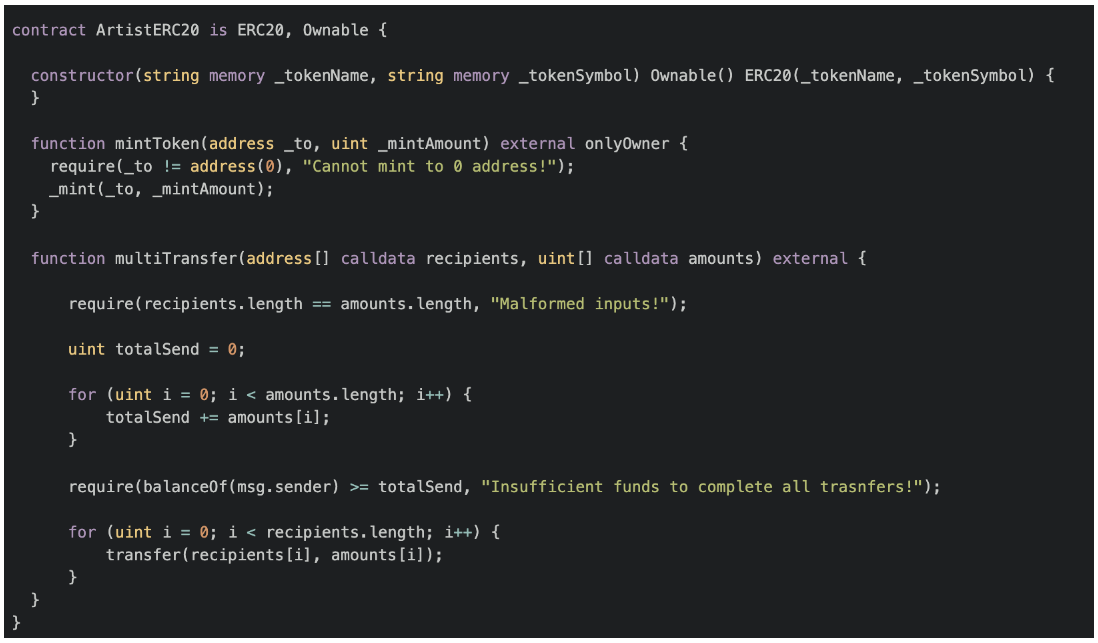
Appendix B

Artist Tokens implementation

This appendix shows the implementation of the artist tokens.

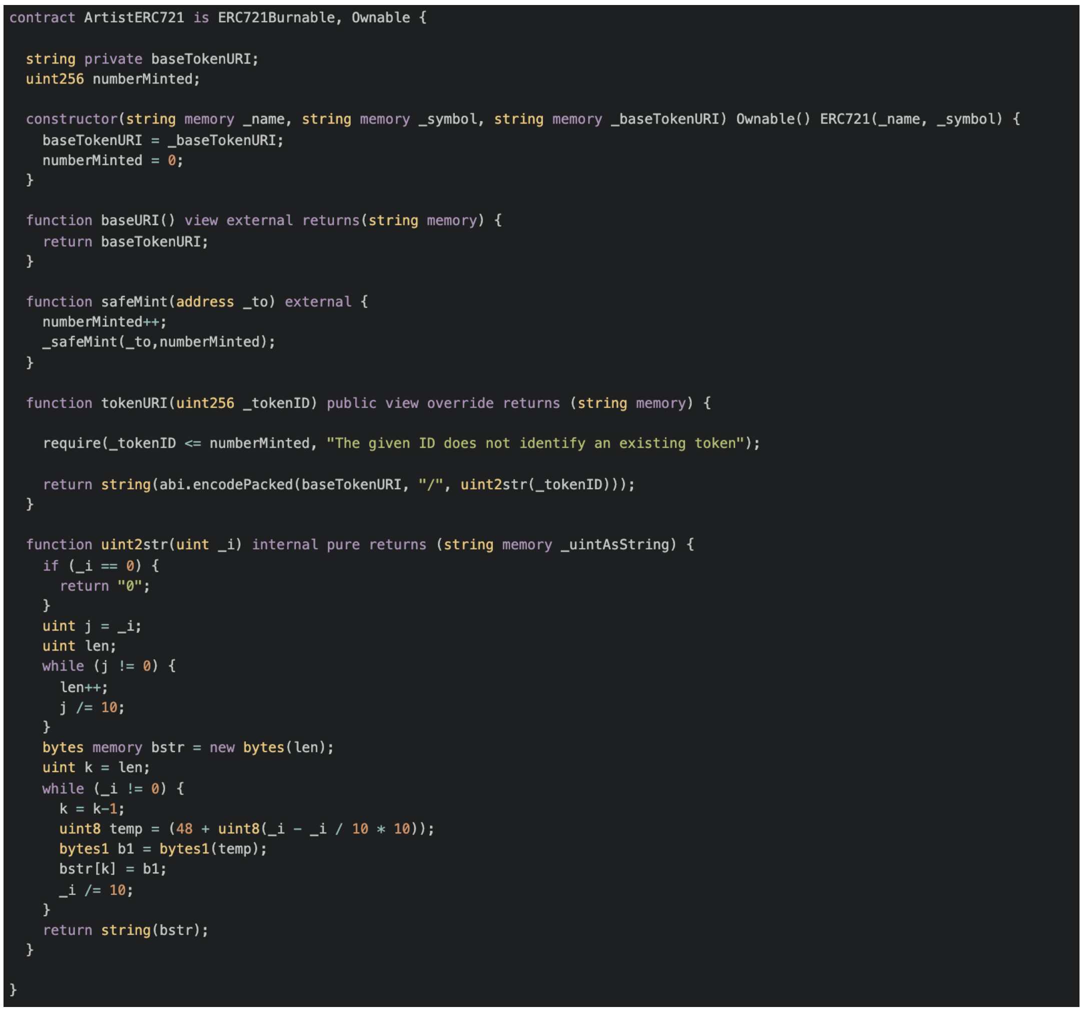
The smart contracts that regulate both fungible and non-fungible tokens are respectively *ArtistERC*20*.sol* and *ArtistERC*721*.sol*.

Follows the code of both tokens.



```
contract ArtistERC20 is ERC20, Ownable {

  constructor(string memory _tokenName, string memory _tokenSymbol) Ownable() ERC20(_tokenName, _tokenSymbol) {
  }

  function mintToken(address _to, uint _mintAmount) external onlyOwner {
    require(_to != address(0), "Cannot mint to 0 address!");
    _mint(_to, _mintAmount);
  }

  function multiTransfer(address[] calldata recipients, uint[] calldata amounts) external {

      require(recipients.length == amounts.length, "Malformed inputs!");

      uint totalSend = 0;

      for (uint i = 0; i < amounts.length; i++) {
          totalSend += amounts[i];
      }

      require(balanceOf(msg.sender) >= totalSend, "Insufficient funds to complete all trasnfers!");

      for (uint i = 0; i < recipients.length; i++) {
          transfer(recipients[i], amounts[i]);
      }
  }
}
```

Figure B.1: ArtistERC20.sol



```
contract ArtistERC721 is ERC721Burnable, Ownable {

  string private baseTokenURI;
  uint256 numberMinted;

  constructor(string memory _name, string memory _symbol, string memory _baseTokenURI) Ownable() ERC721(_name, _symbol) {
    baseTokenURI = _baseTokenURI;
    numberMinted = 0;
  }

  function baseURI() view external returns(string memory) {
    return baseTokenURI;
  }

  function safeMint(address _to) external {
    numberMinted++;
    _safeMint(_to,numberMinted);
  }

  function tokenURI(uint256 _tokenID) public view override returns (string memory) {

    require(_tokenID <= numberMinted, "The given ID does not identify an existing token");

    return string(abi.encodePacked(baseTokenURI, "/", uint2str(_tokenID)));
  }

  function uint2str(uint _i) internal pure returns (string memory _uintAsString) {
    if (_i == 0) {
      return "0";
    }
    uint j = _i;
    uint len;
    while (j != 0) {
      len++;
      j /= 10;
    }
    bytes memory bstr = new bytes(len);
    uint k = len;
    while (_i != 0) {
      k = k-1;
      uint8 temp = (48 + uint8(_i - _i / 10 * 10));
      bytes1 b1 = bytes1(temp);
      bstr[k] = b1;
      _i /= 10;
    }
    return string(bstr);
  }

}
```

Figure B.2: ArtistERC721.sol

In the case of the ERC-721, it provided the ability to use the tokens through the use of $ERC721Burnable$ extension which enables the burning of the tokens once it is used. In this way, it made sure that one NFT is used only once.