Creating a website for a programmatically generated NFT collection

Case: Solana Network

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Bachelor thesis

Degree Programme in Business Information Technology

or Computer Applications

fall, 2022

Eszter Péterfay

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology Abstract

Author Eszter Péterfay Year 2022

Subject Creating a website for a programmatically generated NFT collection: Solana Network

Supervisors Deepak Kc, Elina Hietaranta

ABSTRACT

Blockchain facilitates the foundation of the decentralization. Compared to traditional banking systems, it offers a more efficient and secure way of handling transactions. This innovative technology serves as a platform for digital tokens as well. These tokens can be categorized based on their fungibility. For example, there are so called non-fungible tokens (NFTs). The main objective of the thesis was to find an efficient way to generate NFT collections and to create a website where these NFTs can be bought for cryptocurrency.

If one would like to create an NFT art collection, which usually consists of thousands of images, it would be a tedious job to create them manually one by one. The thesis describes how to create an NFT collection programmatically. The creation of the minting website, which communicates with the blockchain, and so-called smart contracts is also described along with the underlying concepts, that are essential to grasp the idea of the workflow.

The implementation phase is supported by the theoretical framework, which covers an overview of the blockchain itself followed by the asset categorization, where the NFTs are introduced, then the networks and smart contracts in general are explained. Eventually the tools used for the development are introduced along with the overall purpose of the thesis.

The thesis answers the research questions by the description of the development work, for which HashLips Art Engine and The Candy Machine were proven to be useful tools. The thesis provides useful information for the overall process of the NFT collection and minting website creation.

Keywords blockchain, cryptocurrency, non-fungible token, Solana

Pages 45 pages

Glossary

Airdrop	Sending small amounts of coins or tokens to active users who in return contribute to the promotion of the token in some form. Thus, airdrops are the most common marketing methods of startups to gain attention.
Byzantine fault tolerance (BFT)	It is a property of system that can resist the maliciously acting network nodes keeping the system reliable by means of the majority of computers remaining intact.
Consensus Mechanism	It is a fault-tolerant mechanism that is used in computer systems to reach an agreement among the network of computers on single data value or a single state to validate operations and keep record on them.
Gulf stream	Mempool-less transaction forwarding protocol.
Market capitalization (market cap)	It is the total dollar market value of the outstanding shares of stock of a company, which is equal to the total number of shares multiplied by the current market price of one share.
Memory pool (mempool)	It is a smaller database shared between nodes that holds unconfirmed and pending transactions.
Minting	In cryptocurrency, minting is a decentralized method that enables a person to generate a new token without the involvement of a central authority. It can either be a non-fungible token or a crypto coin.
Proof-of-History (PoH)	A stack of proofs, each of which proves that some data existed before the proof was created and a precise amount of time passed between them.
Proof-of-Work (PoW)	It is a decentralized consensus mechanism that solves arbitrary puzzles by complex mathematical calculations in order to deter malicious uses of computer power.
Sealevel	Solana's parallel smart contracts run-time.
Turbine Block Propagation	Multi-layer block propagation mechanism used by Solana clusters with the aim of broadcasting transaction shreds to all nodes with the least number of duplicate messages.
Lamports	A fractional native token with the value of 0.000000001 sol.

Contents

Figures, program codes, commands and tables

1 Introduction

Blockchain handles enormous traffic and its massively increasing popularity urges people worldwide to integrate the technology into their businesses or invest in cryptocurrencies, which are running on the blockchain. According to the statistics of CNBC, one in ten people invest in cryptocurrencies in the United States. (Reinicke, 2021) Cryptocurrency is a decentralized digital money, meaning that they are not controlled by any central authority. Cryptocurrencies can be exchanged for other kind of tokens on the blockchain, like NFTs (non-fungible tokens).

NFTs have wide variety of use cases and can be extremely lucrative, therefore this market is highly competitive and different marketplaces are filled with enormous amount of collections these days. Moreover, these collections often have their own websites giving insight into the concept of the collection and making its pieces available for purchase as well. These websites have a complex system to communicate with the blockchain involving so-called smart contracts and specific web development principles.

The NFT collections tend to be large - consisting of thousands of pieces, which can have artistic features or even represent real estates. In the case of digital art collections – which the thesis will focus on –, it would be tedious and time-consuming to create the belonging pieces one by one. The process however can be coded which enables the programmatical generation of the collection. The goal of the thesis is creating a collection generated by a program as well as making a website to make the pieces of this collection available for sale on the blockchain.

Blockchain is a huge and very complex topic, therefore the thesis will discuss in more general terms and the fundamental concepts which are indispensable to provide a coherent picture of the knowledge base including but not limited to the blockchain itself, NFTs and networks.

This thesis aims to shed light on:

- What is the most effective way of creating NFT collections?
- How to create a website to sell NFTs on Solana Network?

2 Fundamentals of blockchain

This chapter will expand on different concepts, that serve as the knowledge base of the thesis. It will discuss what blockchain is, what benefits it can offer as well as the related concerns. This will be followed by the definition of cryptocurrency and related terms, like wallets and fungible and non-fungible tokens. The networks which cryptocurrencies are based on will be also explained and introduced by the comparison of two relevant networks on the blockchain, which will give a better insight into what technologies are used by these networks and how those affect the performance.

2.1 Blockchain

Blockchain technology was first introduced by two mathematicians in 1991. Stuart Haber and W. Scott Stornetta wanted to implement a system, which eliminates the chances of getting targeted by hackers tampering with the date time stamps of digital documents. It has been eventually implemented first in 2009. (Hayes, 2022)

Blockchain is a digital platform primarily used for running cryptocurrencies. It is responsible for secure, fast, and efficient handling of transactions in a decentralized way, by eliminating financial intermediaries, like banks. (Investopedia, n.d) In more technical terms, blockchain, which is also known as distributed ledger technology (DLT), is a distributed and secured database or ledger that guarantees fidelity and trust without involving third party authorization. However, its structure is different from typical databases. In a blockchain, the data is stored in chunks called blocks. After the amount of data has reached the storage capacity of a block, it gets linked to the previously filled block forming a chain via cryptography. (BuiltIn, n.d.)

Then another block will be formed to store the new information, (usually) strictly following a chronological order, and keeping records on the date, when a new block has been added to the chain by assigning an exact timestamp to them. The data stored on blockchain is immutable, meaning that these records are irreversible and cannot be altered, deleted, or destroyed. (IBM, What is blockchain technology?, n.d.)

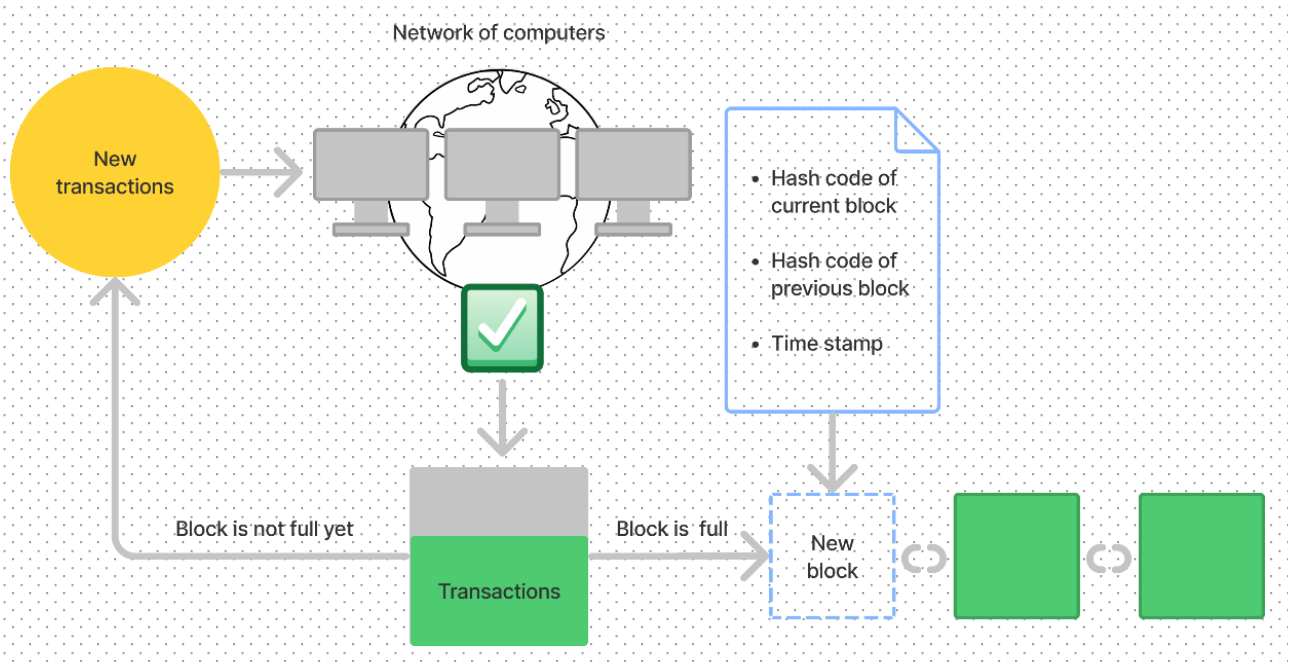
Blockchain has many advantages over the traditional banking system. For example, the transactions can go through at any time whereas in the case of banks people need to adjust to

business hours. There is a significant difference between the speed of the transactions as well. The transactions on blockchain should normally take less than one hour, whereas a transaction in banks normally takes at least one day. There are always some transaction fees for banking transactions to take into account, that usually cost up to $50. However, in case of blockchain, depending on the congestion and the network, the transactions fees can cost higher or lower or even might have no cost. (Hayes, 2022)

2.1.1 Security

The security of blockchain can be assessed by observing the steps of the transactions happening on the blockchain. Figure 1 illustrates the stages of these transactions.

Figure 1 Transactions on blockchain



Each block on the blockchain is stored linearly and chronologically, hence new blocks are added to the end of the blockchain. Each block is tagged with a hash and has a record on its timestamp as well as on the hash of the previous block. If one would change a previous block on the chain, it would change the hash assigned to it and the timestamp as well. That would result in conflicts in the versions of the blockchain. The outstanding version could be noticed and ignored as it is no longer aligned with the original version that others have. This makes blockchain immutable and

helps to prevent hackers to do changes on blockchain. These attacks could be successful only if at least 51% of the copies of the blockchain would match with the hacker's version. That is nearly impossible to manage, because that means that the hacker should manipulate simultaneously more than half of the copies of the blockchain concerning the hash and the timestamp of the affected blocks. Especially that the size of the cryptocurrency networks is growing so fast, meaning that drastic alterations should be made to the blockchain without getting caught. Hence, this way of storing data turned out to be very reliable and for this reason many companies have adopted blockchain such as Walmart, Pfizer, AIG, Siemens and Unilever. (Hayes, 2022) (Radocchia, 2018)

There are plenty of advantages to blockchain, however there are certain drawbacks as well which will be discussed in the following chapters.

2.1.2 Advantages

The advantages of blockchain include:

- Accuracy: almost all human involvement is replaced by computers in the validation process, which leads to more accurate transactions, hence fewer human errors occur. If by any chance an error is nevertheless made in the process it will be present only in one copy of the blockchain, so that would not affect the whole system of blockchain.
- Cost reductions: because blockchain eliminates the third-party authorities, the associated costs will be reduced as well.
- Decentralization: blockchain does not have a central location to store information but spread across a network of computers. This makes the system less vulnerable, because it is hard to compromise the entire network of computers.
- Efficient transactions: transactions in central authorities face a lot of barriers such as business hours and several steps of the validation process in a bureaucratic system, therefore it can take at least one day to get a transaction through. Blockchain on the contrary is operating every time and gets a transaction processed in about 10-60 minutes.
- Private transactions: the transaction on the blockchain can be accessed by anyone with an internet connection. It does not mean however that the users can be identified behind the transactions.

- Secure transactions: each block has a unique hash and timestamp linked to a previous block forming a chain. Any attempt to alter a block would change the hash and the timestamp of the block but would not respectively affect other blocks and that way the altered one can be easily noticed in the system.
- Transparency: blockchains are open source meaning that anyone can view its code. (GeeksForGeeks, 2022)

2.1.3 Disadvantages

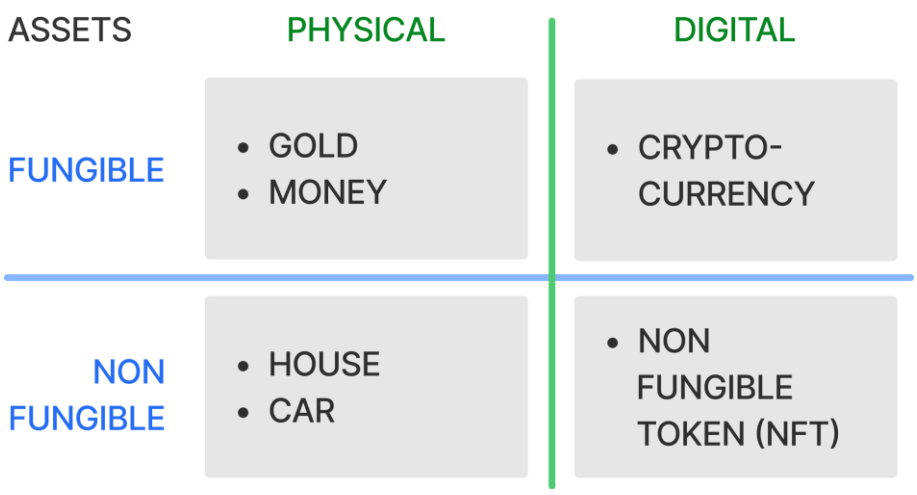
On the other hand, there are some disadvantages of blockchain as well, such as:

- Technology cost: running all computers that are used by the blockchain costs vast amount of energy. Solar power, and wind farms are however used to balance the consumption.
- Speed and data inefficiency: blockchain can handle a very limited number of transactions per second (TPS) - commonly not more than twenty, which is not effective compared to Visa that can process 65,000 (TPS). There are new networks on the blockchain however that can handle transactions in a better rate.
- Illegal activity: the technology in wrong hands enable illicit and untraceable transactions. For example, this technology is used by dark web where illegal-drug and money laundering marketplaces operate. The concern of many people for this reason, that using blockchain technology can do more harm than good. (Hayes, 2022)

2.2 Fungible vs non-fungible tokens

Assets are most commonly associated with money or gold. These two share common attributes as they are both physical and can be exchanged for one another. They can be used directly or indirectly to express the value of other assets like a house or a car, therefore it makes sense to use money for purchase. However, it is not working other way around, meaning it is not possible to express one unit of money with a house or a car. However, both are physical. Digital assets can be categorized the same way. There are cryptocurrencies that can be exchanged for one another. They have the same function as money, but there are also tokens which can be bought and sold using these cryptocurrencies, but one token of this kind cannot be exchanged to another token of the same kind because their value might be different. As Figure 2 illustrates, these assets cannot be categorized as physical or digital assets only but based on their fungibility as well. Fungibility can be defined as the ability of assets to be exchanged for one another regardless of the type of the other asset. (Frankenfield, 2021b)

Figure 2 Asset categorization



According to the definition, fungible assets are identical to each other, like money, gold or cryptocurrency that can be used to express the value of one another as well as non-fungible assets, like a house or car or even so called non-fungible tokens in the digital world. The next subchapters will explain what cryptocurrencies and non-fungible tokens are.

2.2.1 Cryptocurrencies

The earliest attempt to create a decentralized currency was in 1998. This currency called "bit gold" was invented by Nick Szabo, a blockchain pioneer who proposed using blockchain to secure digital payments. Although bit gold was never used, it is widely considered to be the precursor to Satoshi Nakamoto's bitcoin protocol. (Sharma, 2021)

Cryptocurrency is a virtual currency that runs on a blockchain and secured by cryptography, which enables secure online payments, with no third-party authorization required. Cryptography makes transfers secure by encrypting their data. Cryptocurrency can be mined or just like official currencies like Dollar or Euro, it can be purchased also on different exchanges online such as Binance or Coinbase. (Frankenfield, 2022a)

In fact, cryptocurrencies are not commonly used for general purposes, due to their increasing value and popularity, however, they are becoming widespread as trading instruments. Moreover, there are countries that are considering supporting cryptocurrencies as official currency. According to CoinMarketCap, El Salvador became the first country that accepts Bitcoin as a Legal Tender as of 2021. (CoinMarketCap, n.d.)

Bitcoin is the first and most valuable cryptocurrency that was introduced in 2008 and launched one year later. A mysterious person – or a group of people - with a pseudonym, Satoshi Nakamoto is associated with Bitcoin, whose identity has remained unclear since. (Frankenfield, 2021a) Different types of cryptocurrencies have flooded the market after Bitcoin, which are called altcoins indicating that those are alternatives to Bitcoin. Ethereum is the second largest coin in terms of market cap. (ELEV8, 2020)

People need to store these currencies in a secure way, especially if it has high value. As people tend to have wallets to hold their cash, they can have digital wallets for cryptocurrencies as well. There are different kind of wallets to choose from depending on several factors to consider, such as the type of cryptocurrency, the balance, and the purpose of holding crypto in long term. (Coinbase, n.d. -a)

These wallet types are the following ones:

- Hosted wallets: the most popular wallet because it is easy to set up and considered to be secure. It is called hosted, because a third-party is responsible for keeping users' crypto. The benefit of hosted wallets is that it cannot be stolen or missing like a physical object. The downside of this kind of wallet is that users might not be able to take advantage of the full potential of their crypto if some features of them are not supported by the host. The setup can be done in a few steps: the right platform needs to be selected taking into consideration the most important aspects such as security, ease of use, and compliance with government and financial regulations where an account should be created while choosing secure password. To enhance security, it is recommended to set up a 2-factor authentication. (Coinbase, n.d. -a)

- Self-custody wallets: it is self-custody because there is no third party – "custodian" – involved, but the user has the full responsibility and control. Only the software is provided in this case to store crypto, hence safeguarding the password is the responsibility of the user only. The password consists of twelve random words, which is a "seed phrase" functioning as a “private key”. There is a temporary password which can be used after the “seed phrase” has been set up, which can be changed, but that requires the “seed phrase” too. There is no possibility to recover or change the “seed phrase” if lost, meaning that the crypto cannot be accessed anymore. Moreover, if someone discovers that phrase, they can access all the assets on the wallet. However, these kinds of wallets give more control, which enables more advanced crypto activities such as lending and staking. During the setup there is no need to share any personal information to create a wallet, but it is crucial to keep the generated “seed phrase” in a secure location, because if it is lost or forgotten, there is no way to get a new “seed phrase” if one needs to change the custom password. (Coinbase, n.d. -a)

- Hardware wallets: as its name suggest it is a physical device, about the size of an ordinary USB stick, which can be used offline. It is relatively expensive (about $100) and hard to set up, but its key benefit is the advanced security. The wallet remains intact even if the computer is hacked. This wallet is most suitable for users who keep significant value of cryptos on their wallet for longer period. (Coinbase, n.d. -a)

2.2.2 Non-fungible tokens

NFT is the abbreviation for non-fungible token, which is a specific cryptographic asset on the blockchain. These assets are unique and tend to have some artistic features, like a painting or music. Having different values, they cannot be exchanged for one to another. They can be purchased with fungible tokens, which can make up the exact value of them. When someone buys an NFT, its unique identification code and metadata will be registered on the blockchain by the transaction, which serves as a proof of the ownership, which is represented by the NFT. NFTs can be even the tokenized representation of physical non-fungible assets. (Coinbase, n.d. -b)

It is controversial what will be the future of NFTs in long term, but many people claim that the impact of NFTs will be permanent on the crypto market (Schmidt, 2022). The most successful NFTs are in fact incredibly lucrative both for the seller and the buyer, and therefore lots of people see in NFTs exceptional opportunities. The most expensive NFT was sold in 2021 for over $69 million. This piece of art was compiled by Beeple, out of five thousand days of work. (Sharma, 2022)

There are many forms of NFTs which make the market diverse. Even a single Tweet can be turned into an NFT. For example, the first tweet has been tokenized by Jack Dorsey and sold for $2.9 million. There are NFTs which can be used in games as avatars or combined with another to "breed" a new one from those. The most famous example of these kind of NFTs is CryptoKitties, a collection of reproducible virtual cats having a unique set of attributes, which has been launched on Ethereum's blockchain. Cryptokitties gained a massive fan base that spent $20 million worth of Ether within a couple of weeks. Certain NFTs have more serious use cases as they can represent anything from an artwork to a real estate and in that sense NFTs constitutes a significant evolutionary leap in the modern finance system, which promotes market efficiency and decentralization. (Sharma, 2022)

NFTs on blockchain are generally secure. However, there are many precedents of frauds that are most often derived from compromised or intentionally misconfigured smart contracts. The victims of these frauds might lose access over their NFTs or after purchase they cannot sell them. (Modderman, 2022)

2.3 Networks on the blockchain

Cryptocurrencies are based on decentralized networks, which are using blockchain technology. A brief, yet comprehensive definition by Cointelegraph (n.d.) describes networks as

> a technical infrastructure that allows applications to access ledger and smart contract services. Smart contracts are primarily used to originate transactions, which are then transmitted to each peer node in the network and recorded immutably on their copy of the ledger.

Ethereum was considered the primary network during the last years. It was launched in 2014. However as mentioned in chapter 2.1.3, one of the drawbacks of blockchain technology is that commonly it handles transactions at a relatively low speed. This applies to Ethereum, which can handle 13-15 transactions per second (TPS), which is not an advantage. Moreover, the transaction fees are quite high on Ethereum. The most relevant competitor became Solana, which was launched in 2020 and provides higher transaction speed (65K TPS) at lower cost. (Trends, 2022)

Ethereum has enormous market cap, $497 B, in addition its upcoming upgrade is very promising, which will be able to process over 100K TPS. Solana on the other hand has considerable features to offer over the current version of Ethereum and it is getting more attention lately. The fees are currently the biggest reason of choosing Solana over Ethereum by many users. However, Ethereum reigns still due to its complex ecosystem and once the upgrade is delivered, it might be beating Solana in terms of speed. (Avyan, n.d.)

The underlying technology makes a huge difference between the two networks in question. Each has different consensus mechanism which constitutes their core, and each has their ways of solving the problems concerning scaling. (Avyan, n.d.) In the following chapters the technologies used by the networks will be discussed and their impact on the performance, which makes relevant differences between the two solutions.

2.3.1 Ethereum overview

Ethereum 1.0 is based on a Proof-of-Work (PoW) mechanism similarly to Bitcoin's blockchain. Enormous network of computers is taking care of the consensus on the blockchain. These computers are also used for mining at the expense of the transaction speed and overall performance. (Trends, 2022)

On the second place Ethereum's stateful nature that makes it different from Solana. It means that all transactions on the network are recorded into one state, meaning that on any new transaction the entire network including all the computers taking care of the mining and the consensus must update their copy of the blockchain to make the transaction valid. The process makes great demands on computing power which significantly slows down the system. (Yaffe, 2018)

Ethereum handles transactions slower, however it has still a lot to offer. Ethereum supports a variation of technologies and programming languages, which provides a good platform for creating different smart contracts - but the most widely used is Solidity. The scalability of Ethereum is limited, due to its stateful nature. However, it supports multi-chain networks that improves the scalability. (Avyan, n.d.)

2.3.2 Solana overview

The most crucial difference between the two networks is the underlying consensus mechanism, which in case of Solana is known as Proof-of-History (PoH). This requires essentially a sequence of computational steps that determine the time passage between two events cryptographically. Then timestamps are added accordingly to each transaction, but the transactions are not chronologically ordered, and this differentiates Solana primarily from Bitcoin and Ethereum. The other significant difference is that while Ethereum has a stateful, Solana has stateless architecture, the latter helps to reduce the memory consumption and makes Solana highly scalable. (Avyan, n.d.)

By utilizing Tower Byzantine Fault Tolerance (BFT), the nodes of Solana do not need to communicate with each other in real time, which makes the Solana network more powerful in overall. In addition to this, so-called Gulf Stream implementation accelerates the transactions. This

enables the network to process transactions much ahead of the stipulated time at over 50,000 TPS. (Yakovenko, 2019)

The excellent scalability of Solana makes it outstanding amongst its competitors. This scalability is enhanced by the Turbine Block Propagation protocol that is the key component of the blockchain. Using this protocol, the data can be transferred in smaller fragments, which makes transactions more efficient. Another important innovation of Solana is called Sealevel that enables to process programs in parallel. (Yakovenko, 2019)

2.4 Smart contracts

Smart contract is a computer coded and self-executing agreement that is configured into the blockchain to facilitate and verify a set of conditions. If the conditions are validated according to the agreement, its terms will be fulfilled between the buyer and seller. This piece of code allows to carry out trusted transactions without the need for any central authority or legal system. Smart contracts were proposed by the inventor of "Bit Gold", Nick Szabo. (Frankenfield, 2022b)

Many of the predictions from Szabo's paper got justified in the later development of blockchain. About the execution of the smart contracts, he wrote in his paper:

> "These new securities are formed by combining securities (such as bonds) and derivatives (options and futures) in a wide variety of ways. Very complex term structures for payments can now be built into standardized contracts and traded with low transaction costs, due to computerized analysis of these complex term structures". (Frankenfield, 2022b)

Different kind of smart contract are used by Solana and Ethereum. The most widely used smart contracts are written in Solidity. Solidity is not relevant concerning the practical part of the thesis, but it is worth to mention, because it has an essential role on Ethereum Network. Solidity is an object –oriented, high-level language, which is influenced by C++, Python and JavaScript. It is using curly brackets in its syntax, statically typed and supports inheritance. It is designed to target the Ethereum Virtual Machine (EVM). It is used to implement smart contracts for uses such as voting and crowdfunding. (Solidity, n.d.)

Solana is however using different kind of smart contracts, which are called Programs. Rust, C and C++ are used to build programs on this network. There are existing programs that can be used for development created by Solana Labs. There are two sets of programs that are maintained with Solana's core system. These are so-called Native Programs and the Solana Program Library (SPL). Native Programs are the core of Solana. For example, the System program is a Native Program of Solana, which is used for creating accounts and transferring SOL (Solana's native token). The Program Library includes many different programs. One of the most popular Program Library is the Token program, Solana's Ethereum ERC-20 equivalent, which enables the interactions on Solana blockchain with tokens including NFTs. These interactions in this case are minting and transferring tokens. (Barker, 2021)

3 The core concepts of Solana

The following chapters will dive deeper into specific aspects of Solana explaining what clusters it has and the architecture of the network which will support the comprehension of the development processes. These chapters heavily rely on a few sources, because being official documentations, those provide the most reliable information on the covered subjects.

3.1 Solana Clusters

Solana maintains three types of clusters for different purposes. Two of them is used for development purposes, and this chapter will explain what the difference is between those two clusters that are used as a sandbox and how they differ from the real cluster. The real cluster of the network is Mainnet Beta, where the tokens are real. It is permissionless and persistent. (Solana, n.d. -c)

Devnet is a sandbox that meant to be used to test user experience either as a user or a developer. It is recommended for developers or potential validators to first target devnet. Devnet has a faucet for airdrops for application testing from where developers can obtain tokens that are not real. It typically runs newer software version than the Mainnet Beta. (Solana, n.d. -c)

Testnet is similar to devnet so the tokens used in this cluster are not real either but can be also obtained from a faucet by airdrops. Testnet is used however primarily to test recent release features, to see network performance, stability, and validator behaviour. This cluster runs the newest software releases. (Solana, n.d. -c)

The accounts, tokens, transactions, programs, and blocks on each cluster can be inspected on Solana Explorer by their address or id. Each can communicate with different entrypoints and endpoints. (Solana, n.d. -c)

3.2 Accounts

Solana's stateless structure means that unlike Ethereum, Solana's Programs do not hold data. The data is stored on so-called accounts which the Programs are communicating with to retrieve information. Accounts are essential part of the Solana architecture. By default, all accounts are owned by the System Program of Solana, which is a native program. (Cookbook, 2022a)

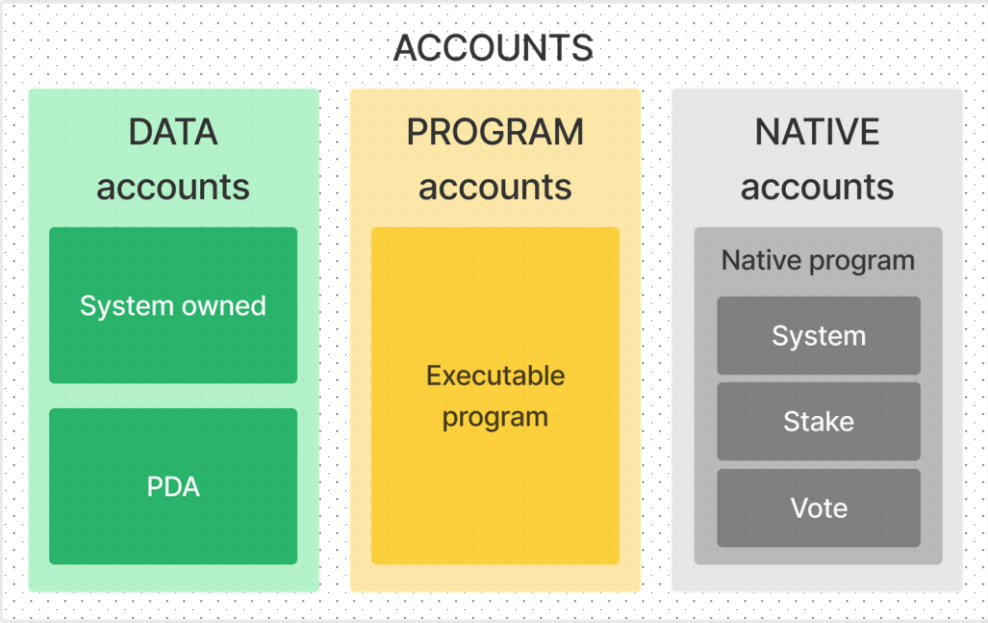
As Figure 3 shows, there are three different types of accounts:

- Data accounts that store data. There are two types of data accounts. System owed accounts that are owned by the System Program as its name implies and Program Derived Address (PDA) accounts which are owned by a Program which is not native program. PDA accounts can be used to sign on behalf of a program.

- Program accounts that store executable programs. Executable programs determine the behaviour of the program accounts. These accounts hold Programs themselves that are used to process instructions.

- Native accounts that are reference to native Programs of Solana, from which the most important is the System Program. It will be covered more in depth in chapter 3.3.

Figure 3 Account categorization and relations



Each account has an address, which is usually a public key, and an owner. Besides this information accounts store:

- Lamports. The number of lamports owned by the account. Accounts have a lifetime, and that lifetime is indicated by lamports. Lamport are used to pay for the persistence of the metadata lives on the accounts. Validators are taking care of collecting this ”rent” periodically from all accounts. (Solana, n.d. -a) The condition for accounts to remain exempt from paying rent is to maintain a minimum balance on an account which is equivalent to 2 years of rent payments.

- Owner. Each account is owned by a program, but programs themselves are also stored on an account.

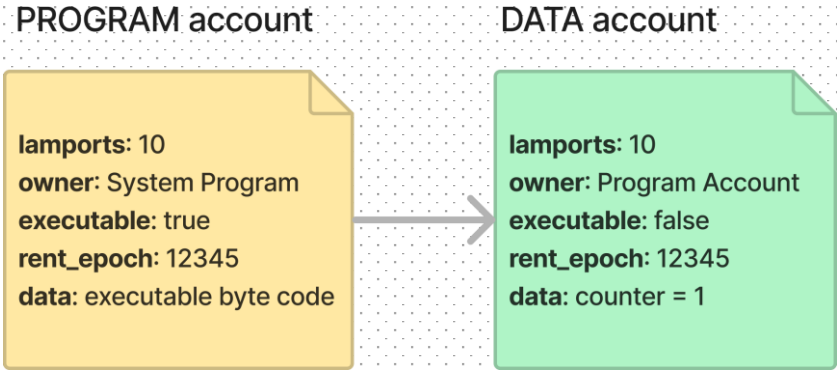
- Executable. This is a boolean indicating whether the account can process instructions.

- Data. The raw data stored by this account in byte array.

- Rent_epoch. The next epoch that needs to be paid for.

There is a Program account and a Data account on Figure 4. The Data account is owned by the Program Account. The Data account holds data affected by the program code stored on the Program Account. Therefore, the Program stored by the Program Account can be executed but it does not store actual data, but a program to be executed. On the other hand, Data Account stores data, but it is not executable. (Cookbook, 2022a)

Figure 4 Program Account and Data Account (Cookbook, 2022a)



3.3 Programs

Smart contracts, that are called Programs on Solana blockchain play significant role in handling on-chain activities, by processing instructions that are coming from end users or other Programs. Programs are stateless which is a great innovation of Solana Network because these programs are not holding any data which makes them more effective. All Programs are executable and stored on accounts. They can communicate with other accounts that store the data. The data is received via instructions as references. Any developer can write Programs using Rust or C++ and deploy them to the blockchain. Once they are deployed, it is accessible by anyone who knows how to communicate with them. There are two types of Programs: Native Programs and Solana Program Library (SPL) Programs. (Cookbook, 2022b)

Solana Program Library consists of different Programs that are responsible for taking care of different on-chain activities around tokens, such as creating, swapping, and lending. The most well-known SPL Program is the Token Program, which defines a common implementation of fungible and non-fungible tokens. (Cookbook, 2022b) (Solana, n.d. -e)

The Native Programs control the validator nodes. These Programs are always upgraded with the cluster itself. Each has a Program id and supported instructions. These are the most significant Native Programs (Solana, n.d. -b):

- System Program. It has different account related jobs, like creating accounts, allocating data on them, assigning accounts to owning programs, transferring lamports to the accounts owned be the System Program and paying transaction fees.

- Config Program. It adds configuration data to the chain as well as a list of public keys that are permitted to modify the configuration data.

- BPF Loader. Deploys, updates, and executes Programs on the chain. (Solana, n.d. -b)

3.3.1 Transactions

Transactions are one or more instructions that are signed by a client using one or more keypairs. There are two possible outcomes of the execution, which are success, or failure. Program executions begin with a transaction submitted to the cluster. The transactions contain a compact array of signatures and a message. (Solana, n.d. -d)

Figure 5 Message content

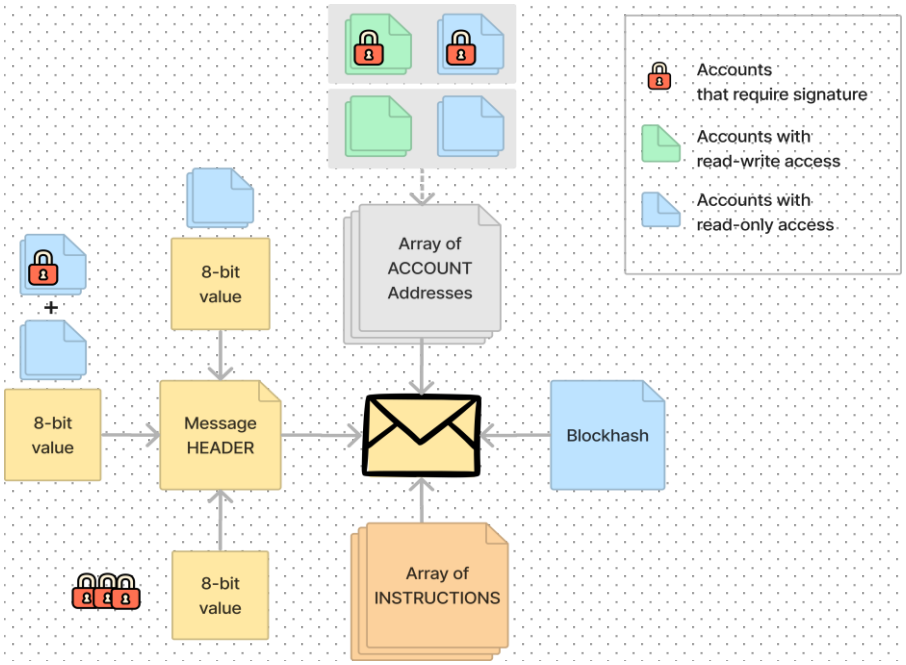


Figure 5 illustrates the message, that contains the followings:

- Message Header. It contains three unsigned 8-bit values. The first value represents the number of required signatures that are included by the transaction besides the message. The second value is the number of the read-only account addresses included in the message. The third is the number of read-only addresses that do not require signatures. (Solana, n.d. -d)

- Compact-Array of Account Addresses. There are accounts with read-write as well as with read-only access. But in both cases, there are accounts that require signature. This array of accounts always includes first those accounts that require signature and from those the ones with read-write access. The same order applies to the accounts that do not require signature, so the accounts with read-write are followed by the accounts with read-only access. (Solana, n.d. -d)

- Blockhash. This contains a 32-byte hash, that is meant to indicate when a client observed last the ledger. (Solana, n.d. -d)

- Compact-Array of Instructions. It contains a program id index, a compact-array of account address indexes, and a compact-array of opaque 8-bit data. The program id index is used to identify which program on the chain can interpret the opaque data. (Solana, n.d. -d) Opaque datatype is called opaque because its internal structure is unknown to the database server. It is a fully encapsulated data type that is built-in or user-defined. (IBM, 2021)

4 Aim and purpose of the development work

The aim of the development project is to create an application which can communicate with the blockchain through smart contracts, to be more precise with so-called programs, as Solana Network will be used. The main functionality of the website will enable users to buy NFTs for SOL. To do so, the user needs to be able to connect their wallets to the website. This way the user can safely connect and manage purchases on the site. The amount of NFTs the user wants to mint also could be set on the page.

The purpose of the project is to get an idea of the workflow and how communication with the blockchain happens under the hood, and what configurations, frameworks and programming languages are needed to achieve that.

4.1 Tools and methods

The development will happen on Linux based system as some frameworks and tools used might not support Windows. Therefore, Windows Subsystems for Linux (WSL) will be used for the development work. WSL is a lightweight Linux architecture, where developers can easily access command-line tools without the need of running a traditional Virtual Machine. A great advantage of WSL is that it does not require to have any hypervisor installed, such as VirtualBox, which enables the user to run Virtual Machines on top of a Host Machine. WSL 2 is running on Windows 11.

For the website, React will be used, which is the most popular JavaScript library maintained by Meta. For that, Node.js or Node Version Manager (NVM) needs to be installed. NVM is going to be installed because it makes it easier to change node version if needed, which also comes with the installation in multiple versions. The website will be deployed on the Solana network, on Solana's devnet, which provides a sandbox to deploy programs that establish the connection between the client and the blockchain. Solana CLI is needed to communicate with the network. This command-line interface is going to be used mostly for obtaining tokens (fake tokens on devnet for testing purposes). It requires a wallet, where the SOL is stored. The most popular wallet that is supported by Solana is called Phantom wallet. It is a self-custodial wallet, meaning that the user has full

responsibility and control over it. These wallets have their id called wallet address which is required for identification.

Solana's programs are written in Rust, which is considered a difficult language to learn and therefore, developers who have experience in creating these programs using Rust are in very high demand and well paid. Thus, coding reliable and secure programs is hard and requires solid knowledge and relevant experience. Fortunately, there is a Solana-based protocol called Metaplex, that makes easier the development work for less experienced developers as well, especially in the case of NFT projects, via its set of tools and smart contracts (programs). It has two major projects, namely Storefront, which is a generalizable NFT selling standard and the other one is The Candy Machine, which can be used for NFT minting. The NFTs will be stored on Arweave, which is a protocol that enables to store data permanently by the decentralized network.

For the NFTs, any photo editor or vector graphical program can be used. There are some open-source programs that can do the job, like Gimp or Inkscape. The most important is that the program should enable the creation of layers with transparent background. PNG is the file format that supports the transparent background.

5 Design and implementation of the project

The most challenging aspect of the development work is that Solana being a very new Network under development, resources and beginner-friendly practical approaches are hard to find. However, there are some channels on Discord with a strong community of dedicated developers who support the learning of beginners and each other providing help on specific issues. First, different installations and setups need to be done. Then the NFT generation, deployment and the website creation are explained.

5.1 Setting up the environment

The most important is to install WSL, which will provide the platform for the development work. Command 1 can be used in the Command Line on Windows to install WSL. In this case, Ubuntu is selected as the distribution ("-d" argument).

Command 1 WSL installation

```
wsl --install -d ubuntu
```

Once the installation of WSL is done, an Ubuntu terminal will show up prompting to create a username and a password. Once it is set up, it is highly recommended to restart the computer to prevent any unexpected behavior of WSL.

The next step is to install the required tools and environments on WSL. The first is going to be the Node.js. More precisely, NVM. As the NVM will be installed from a server, client URL (cURL) is going to be needed as well, so first that needs to be installed using Command 2.

Command 2 Curl installation

```
sudo apt-get install curl
```

Next, the NVM can be installed with the first line of Command 3 that is utilizing the command-line tool that has been installed with the previous command. To verify that the installation was successful, the second line of Command 3 can be used, which checks if NVM exists on the system.

The second command from Command 3 should print “nvm” if it works properly. It might be, that the terminal should be restarted and the command should be tested again to make it work.

Command 3 NVM installation and testing the success of the installation

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh | bash

command -v nvm
```

NVM can be then used to install Node.js. The “--lts” argument in the command of Command 4 indicates that the long-term support (LTS) version should be installed.

Command 4 Installing the latest version of Node.js

```
nvm install --lts
```

Next, the Solana CLI can be installed. The first line of Command 5 will install the release v1.10.8, but it can be replaced with “stable”, “beta”, “edge” or any other software version if another version is needed. If the output of the installation prompts to update the PATH environment variable, then the recommended command below the prompt should be used. The success of the installation can be tested with the command in the second line of Command 5. If some commands throw errors unexpectedly, the terminal might need to be restarted.

Command 5 Installing Solana CLI and verifying installation

```
sh -c "$(curl -sSfL https://release.solana.com/v1.10.8/install)"

solana --version
```

Next, Metaplex should be installed. Git, Node.js, Yarn and TS-node are needed before getting the Metaplex installation done. However, Git usually comes with the Linux system, and Yarn comes with the Node.js installation. The command from Command 6 can be used to verify that they are installed.

Command 6 Verifying yarn and git installation

```
git –version
```

```
yarn --version
```

Node.js can be used to install TS-node and typescript using the commands from Command 7. The "-g" argument means that it will be installed globally not just for a particular project. It is the recommended way to have the installation done.

Command 7 Install TS-node and TypeScript

```
npm install -g typescript
```

```
npm install -g ts-node
```

Then the next step is to install Metaplex using the commands from Command 8. First, the metaplex repository needs to be cloned from GitHub – preferably into the root directory. Then the dependencies need to be installed using Yarn by either specifying the directory in the same command or previously getting into the directory of the repository (using the cd command). The application can be tested only after the NFTs are deployed and certain configurations coming after this step are done.

Command 8 Install Metaplex

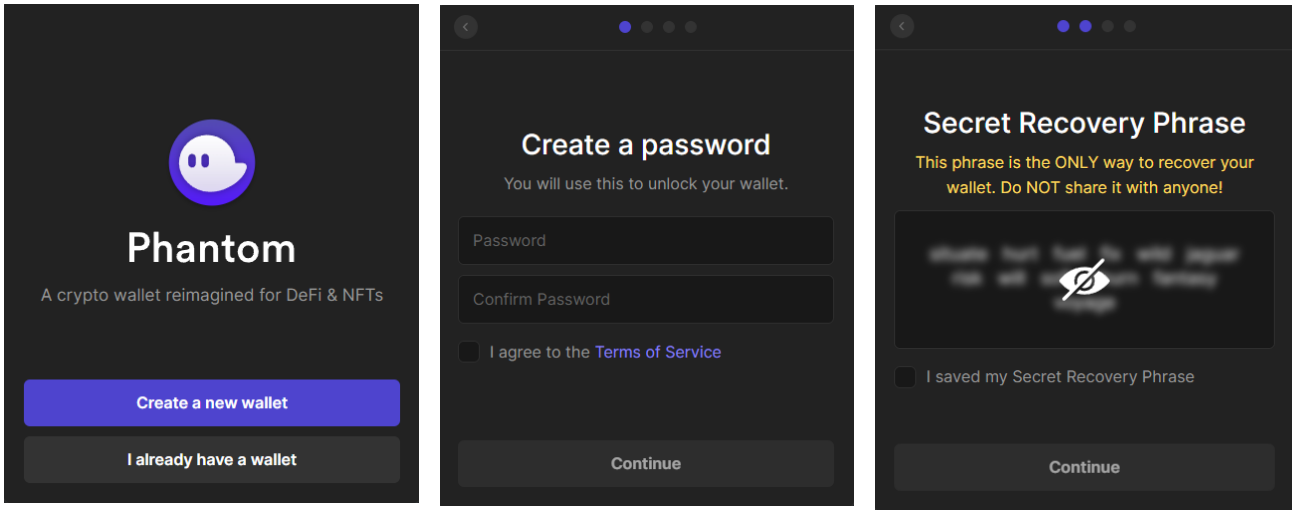
```
git clone https://github.com/metaplex-foundation/candy-machine-ui ~/candy-
machine-ui/
```

```
yarn install <directory of the repository>
```

5.2 Setting up the wallet(s)

The Phantom wallet setup process involves very few steps, and it does not requires any personal information. First, the Phantom extension needs to be set up in the preferred browser. Once the extension is added to the browser, a page will pop up (see Figure 6), where the wallet can be created. Then it asks for a password that can be used to access the wallet later. The Secret Recovery Phrase is crucial to be securely stored in the third step because that is the only way to recover or change the password set in the previous step. Once the box ”I saved my Secret Recovery Phrase” is checked, the wallet can be opened using the shortcut Alt + Shift + P, but it can be pinned as well in the browser header so it will show up next to the address bar.

Figure 6 Phantom extension set up



This wallet will be used to test the minting functionality of the website that will be built for the NFTs. However, for the deployment of the NFTs to Arweave, a wallet needs to be created locally, as the transaction will have a cost. But since everything will happen on the devnet, these costs are not real. The wallet can be created using Solana CLI with the command from Command 9. The argument “-o” is the equivalent of “--outfile” and the path defined points to the directory where the generated keypair --that represents the wallet-- should be stored.

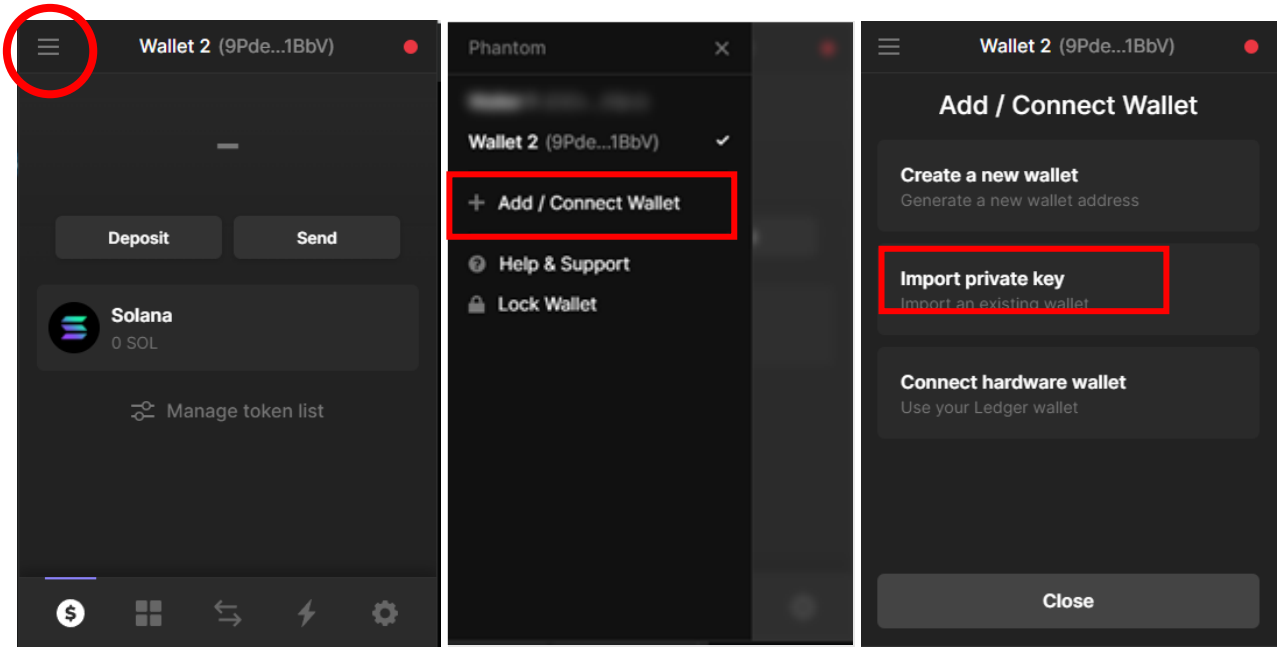
Command 9 Create local wallet

```
solana-keygen new -o ~/.config/solana/id.json
```

After the command is used, it will prompt to enter a passphrase. Once it is entered, it can be approved by pressing the enter key. It will ask to confirm the passphrase by entering it again. When it is entered, a seed phrase assigned to the wallet will be printed as output. It is important to keep that seed phrase safe. The public key in the output represents the wallet address.

Alternatively, the local wallet can be imported into the Phantom wallet application by clicking on the hamburger menu on the wallet (in the top left corner) and selecting the “Add / Connect Wallet” and then selecting the “Import private key” as the Figure 7 illustrates.

Figure 7 Importing the private key of the locally created wallet



The private key can be retrieved from the file for which the path has been defined in Command 9. This is an optional step as the wallet is entirely manageable from the command line as well, however, it is easier to keep track of the balance this way.

5.3 Deploying NFT Collection to Arweave

When designing an NFT collection, there must be an attribute that is shared among all pieces, which will determine the theme of the collection. In this case, the collection will be about funny looking trees that have different hairstyles and sunglasses. There is a layer for the background, the base of the tree and the things which will be changing are stacked on top of the base layers as Figure 8 illustrates.

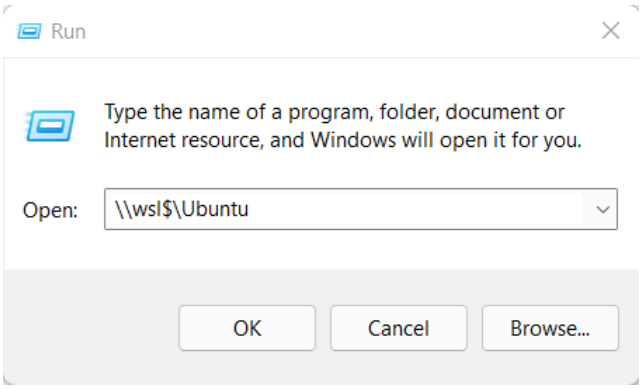
Figure 8 Stacking layers to create a variation for the collection

There are some variations of features that are rarely rendered, which increase the value of the NFTs. When the collection will be rendered, these layers will be weighted so that not every feature of a kind will appear with the same frequency.

5.3.1 Generating NFTs

For the generation of the NFTs, HashLips Art Engine is used as the base for the code. This takes care of the generation of the NFTs. Each NFT should have their metadata which will be also generated with the images. Its source code can be found on GitHub at the following link: https://github.com/HashLips/hashlips_art_engine. The repository can be cloned or forked so there will be an own copy of that repository where the changes can be pushed freely. Once it is cloned using WSL, it can be found in the file explorer by opening the Run window and entering "\\wsl$\Ubuntu" as Figure 9 shows.

Figure 9 Open WSL file system in the file explorer



The cloned repository located on WSL can be opened in Visual Studio Code (or in any other code editor). The repository contains a folder named "layers". Its content needs to be replaced with own images of layers to be used for the NFT creation. The layers should be grouped by features into separated folders. It is important to give these folders descriptive names.

The layers can be weighted using a naming convention. The filenames should consist of the filename followed by a hashtag sign. After the hashtag, numbers should take place indicating the rarity of a layer. These values can be defined on a scale of 1 to 100. The larger the number, the layer will more likely occur in the generation process. It is not advisable however to use too specific numbers, but 1, 10 or 100 instead. The weighting of the layers is not necessary, but either all layers in the same folder should be configured like this or no layers should be weighted at all, which means that all layers can have equal occurrence.

In the “src” folder, there is a file called config.js. This file needs to be customized. As this code can be used to generate NFTs either for Ethereum or Solana, this needs to be specified, because it affects the metadata created for the NFTs.

Program code 1 Code snippet from the config.js to for the metadata generation

```
const network = NETWORK.sol;

const namePrefix = “Name of the NFT collection";
const description = "This is a demo";
const baseUri = "ipfs://NewUriToReplace";

const solanaMetadata = {
  symbol: "max. 5-6 characters",
  seller_fee_basis_points: 1000,
  external_url: "the website for the NFT",
  creators: [
    {
      address: "Wallet address",
      share: 100,
    },
  ],
};
```

The first line of Program code 1 determines the network. It has been set to the Solana network in this case. If Ethereum would be used, the “`network`” variable should be defined as “`NETWORK.eth`”.
In that case, the “`solanaMetadata`” object could be disregarded. However above it, the “`namePrefix`”, “`description`” and “`baseUri`” should be defined in both cases, although in the case of Solana the “`baseUri`” is not relevant, and the other two variables can be blank. However, in “`solanaMetadata`”, there are certain fields that are important to define.

The “`symbol`” should be kept short, it can be for example the initials of the collection name specified. The “`seller_fee_basis_points`” indicates how many percentages are desired from the secondary sales. A thousand points is equivalent to ten per cent. The value cannot be higher than one thousand (10%). The “`external_url`” points to any related website. It can be the website of the collection as well.

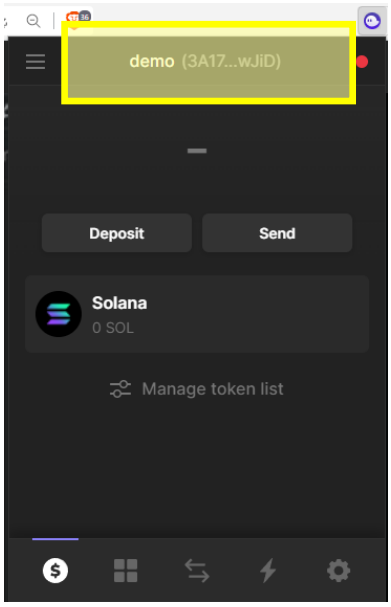
The “`creators`” is an array that defines the creators of the NFTs. That means that there could be multiple objects defined there. The “`address`” is the wallet address of the creator. The wallet address can be retrieved from the local wallet created previously using the command from Command 10 or it can be copied from the Phantom app as well by clicking on that area that is

highlighted on Figure 10. The "`share`" is the percentage value of the earned SOL by an NFT sold. If there are multiple creators defined, the sum of "`share`" values must always make up 100.

Command 10 Get the wallet address (public key)

```
solana address
```

Figure 10 Wallet address



The next snippet of code from the config.js (Program code 2) defines the folders of features in the "layers" folder of the project. Each folder contains images of layers that are different variations of the same feature. The layers will be stacked according to the order of those folders defined in the "`layersOrder`" array.

The array size can be increased or decreased depending on the number of folders (features) in the "layers" folder. Figure 8 illustrates three layers stacked on top of one another. The folders that hold those three layers could be called "Tree" (green layer), "Glasses" (orange layer) and "Headscarf" (purple layer). The layers should follow this order when merging them into one image by the configuration. The "`growEditionSizeTo`" defines the number of NFTs to be generated.

Program code 2 Layer configurations

```
const layerConfigurations = [
    {
      growEditionSizeTo: 100,
      layersOrder: [
        { name: "Tree" },
        { name: "Glasses" },
        { name: "Headscarf" }
      ],
    },
];
```

To generate images with better resolution, the “width” and “height” attributes in the “format” object need to be increased. Also, there is an object in config.js called “background” for which the attribute “generate” can be set to “false”, because in this case there are some custom backgrounds to be used, otherwise, the code would generate a simple background with a random color which would be redundant in this case.

Before starting the generation, some modules are required to be installed. That can be done using the first command from Command 11. It is recommended to run the commands in the Ubuntu terminal, in the project’s directory. The command will manage the installations based on the configurations of the project’s “package-lock.json” file.

The next command can be used to start the generation process. If changes in the configuration need to be made, it is possible to alter the configuration of the collection, therefore it is recommended to try the generation with a small number of pieces first (by setting the “`growEditionSizeTo`” in the config.js to a smaller number). The second command from Command 11 can be used to start the application and generate the images and the metadata. If the generation was successful, the output on the terminal should be similar to Figure 11. It can be seen from this output that ten images have been generated and the numbering starts from 0 which indicates that the Solana network was set in the config.js. In the case of Ethereum, the numbering would start from 1.

Command 11 Installing modules and testing the application

```
npm install
npm run generate
```

Figure 11 Build output in the terminal – NFT generation

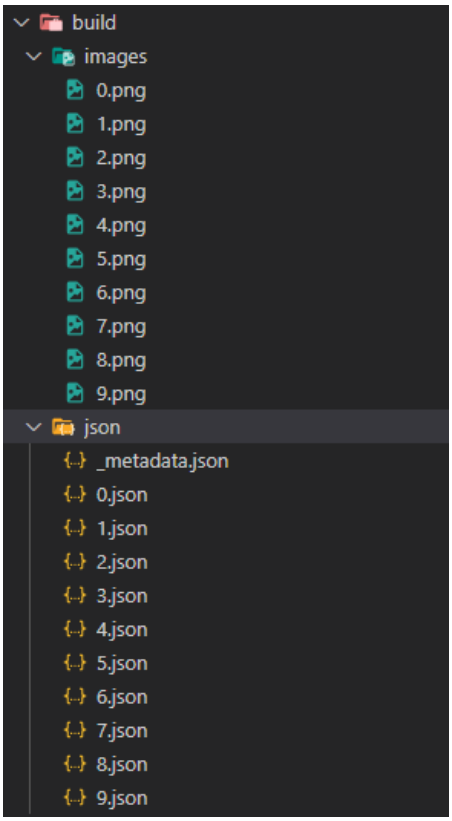
```
~/NFT-project$ npm run generate

> hashlips_art_engine@1.1.1 generate
> node index.js

(node:3597) [DEP0147] DeprecationWarning: In future versions of Node.js, fs.rmdir(path,
(Use `node --trace-deprecation ...` to show where the warning was created)
Created edition: 0, with DNA: 5d2977fe30828028a7c22ce68f40a0b145bf3d3f
Created edition: 1, with DNA: c38c02da58065572f202c90b3ceaa1eebc21dd05
Created edition: 2, with DNA: 5eb6c1ed0fafdadf37bb7b439bd1ddb5a44f82d1
Created edition: 3, with DNA: 1f20f1056daf6c7d2564c82f1b04134bfaab22f7
Created edition: 4, with DNA: 23ca2acf5c27cdb249f8ae035790cc4e6fab4134
Created edition: 5, with DNA: 96c3dabfc52a4fda2a98bb628280be0033a809b5
Created edition: 6, with DNA: ff3974e2f5153b8d4935176024ddf1c50f6f9e07
Created edition: 7, with DNA: 78ea54906c305a4961178f5bce69c0baad1c6a95
Created edition: 8, with DNA: 95f92894291dbbd6a81cf93f4be24abb6b8050f3
Created edition: 9, with DNA: cdb175d2cf4cc9f5dcafd921d3b5a2cedda9e6c8
```

The files have been generated into the “build” folder of the project. The metadata files got saved into the “json” folder and the actual images into the “images” folder as Figure 12 shows the file structure. Each image has a corresponding JSON metadata in the “json” folder with a matching filename.

Figure 12 Generated files in the build folder



5.3.2 Uploading NFTs to Arweave

The next step is to upload the NFTs to a storage where they can be retrieved from on minting. For the deployment of the NFTs The Candy Machine will be used. A config file is needed for the deployment first. An example file can be found in the "metaplex" repository that has been set up previously. The file can be read using the command from Command 12.

Command 12 Reading the example config file

```
cat ~/metaplex/js/packages/cli/example-candy-machine-upload-config.json
```

The content of the file should be similar to Program code 3. This is an example of the configuration. The most important attributes in this configuration are:

- "`price`", which determines the price of each NFT. As this is just a test, the price is not relevant.

- "`number`", which indicates the number of NFTs in the collection. In this case, this value will be set to 10, because that is the maximum amount that can be included in a collection deployed to Arweave on devnet.

- "`gatekeeper`", which determines the address of the provider network. This configuration helps to disable bots to do mints, by setting up a gatekeeper challenge when the user clicks on the mint button. This configuration however is not that relevant at this point so it will be set to null.

- "`solTreasuryAccount`", which will be changed to the wallet address.

- "`goLiveDate`", which is the launch date of the NFTs, when people can start to mint. The format is "`dd MMM yyyy hh:mm:ss Z`" (for example: 01 Jan 2021 00:00:00 GMT)

- "`storage`", which determines where the NFT will be stored.

Program code 3 Candy Machine configuration

```
{
  "price": 0.01,
  "number": null,
  "gatekeeper": {
    "gatekeeperNetwork": "ignREusXmGrscGNUesoU9mxfds9AiYTezUKex2PsZV6",
    "expireOnUse": true
  },
  "solTreasuryAccount": null,
  "splTokenAccount": null,
  "splToken": null,
  "goLiveDate": null,
  "endSettings": null,
  "whitelistMintSettings": null,
  "hiddenSettings": null,
  "storage": "arweave",
  "ipfsInfuraProjectId": null,
  "ipfsInfuraSecret": null,
  "awsS3Bucket": null,
  "nftStorageKey": null,
  "noRetainAuthority": false,
  "noMutable": false
}
```

After the file named “config.json” with the modified content of Program code 4 is saved into the root directory of the metaplex repository, the NFTs with their metadata need to be copied into a folder called “assets”, which should be located in the same directory as the “config.json” file making sure that each image has its own pair of JSON file with the corresponding metadata. The next thing is to change the cluster to devnet because the output of the first command from Command 13 indicates that currently the mainnet is set. To switch to the devnet, the second command from there should be used. The deployment is not free, so SOL needs to be “airdropped” into the wallet. That can be done using the third command from Command 13. The command can be used whenever the balance gets lower than the required fund for certain transactions.

Command 13 Installing modules and testing the application

```
solana config get

solana config set --url devnet

solana airdrop 2
```

At this point, everything is set up to deploy the NFTs to Arweave using the commands from Command 14. The first step is to navigate to the correct directory, and then run the command for

the deployment. In the second command used for the deployment, the devnet environment is specified (using the "-e" option), followed by the path to the keypair of the wallet, and the last argument-value points to the config-path to be initialized on Arweave (option "-cp") followed by the path pointing to the assets folder where the NFTs are saved. To verify the upload the same command can be used with the slight difference of changing the "`upload`" to "`verify_upload`".

The same applies in the case the config if it needs to be updated or modified for any reason – for example, because of changing the launch date ("`goLiveDate`"), except that the "`upload`" has to be replaced by "`update_candy_machine`".

Command 14 Deploying NFTs to Arweave

```
cd metaplex

ts-node ~/metaplex/js/packages/cli/src/candy-machine-v2-cli.ts upload -
e devnet -k ~/.config/solana/id.json -cp config.json ./assets
```

If the deployment of the collection was successful, the output will include the address of the Candy Machine (highlighted in Figure 13) created. That transaction can be looked up in the Solana Explorer using the address of the Candy machine. This address will be needed for the website as well.

Figure 13 Output of the deployment

```
Beginning the upload for 10 (img+json) pairs
started at: 1651475933420
initializing candy machine
Candy machine address:  DYnp6ii8cNbrbTSGYCV3ig1dDvYL28qMA2BPVNJ4PgfL
Collection metadata address:  BCmnNLCQgw72MyXm9eu4vJdvUjBX7RiNBdjyQdY5vuaE
Collection metadata authority:  3A17iAjFAQCuAU81QjSkmyvNrvq95RuTXsUb4M7DwJiD
Collection master edition address:  4jcaUkd48FGSizjSLkt9KA8F8wNP9x3W2CQsK3tb7mgx
Collection mint address:  AAR2tfn7c6aPmgfwBkX69mUzBjy9DUnNgdixCqi93QLJ
Collection PDA address:  149mbiKuZCEaEKw9vuZj23tdJ5HhGeTft5dEjRckwjcx
Collection authority record address:  9a84LywNwJZQqqgtZkFjgoNWBDur2Yj6WY5WNM3CzQSG
Collection:  {
  collectionMetadata: 'BCmnNLCQgw72MyXm9eu4vJdvUjBX7RiNBdjyQdY5vuaE',
  collectionPDA: '149mbiKuZCEaEKw9vuZj23tdJ5HhGeTft5dEjRckwjcx',
  txId: '2csk1iBqDDiyU4Qm2cARncNNa4etgLZk5Yhrthx5qGFgBr6sz7zv3MgANhukcNQRKdwpUpKkeMowByCKHcks4KFT'
}
```

This address is stored in a file called "devnet-temp.json" that has been generated during the deployment. It is located in the ".cache" folder that can be found in the root directory of the metaplex repository. That file contains besides the address and information about each NFT

deployed, such as links that provide the metadata assigned to each NFT respectively. These JSON objects have links (as values of “image” attributes) pointing to the actual NFT images stored on Arweave.

5.4 Creating the Website

The next step is to create a website from where the deployed NFTs can be minted, so users can connect their wallets and purchase NFTs. The minted NFTs on successful transactions should be displayed. The minting functionality would be available when the configured launch data has come. Until that, a timer will show how much time is left until the launch.

If the cloned “candy-machine-ui” repository is opened, there can be found a file named “.env.example”. The following variables should be defined in that file:

- REACT_APP_CANDY_MACHINE_ID, which can be retrieved from the generated “devnet-temp.json” file which can be found in the “.cache” folder of the metaplex repository. This is sensitive information; thus, this is configured externally and not in the application itself. It can be easily accessed using the command from Command 15,

- REACT_APP_SOLANA_NETWORK, which is set to “devnet”,

- REACT_APP_SOLANA_RPC_HOST, which can be set to “https://metaplex.devnet.rpcpool.com/”.

The file can be renamed simply to “.env”.

Command 15 Get the Candy Machine ID

```
cat ~/metaplex/.cache/devnet-temp.json
```

The modules and packages for the application need to be installed next using the first command from Command 16. Then the application can be launched on the local server using the second command. It is recommended to use the WSL terminal to avoid errors, which is available from Visual Studio Code as well.

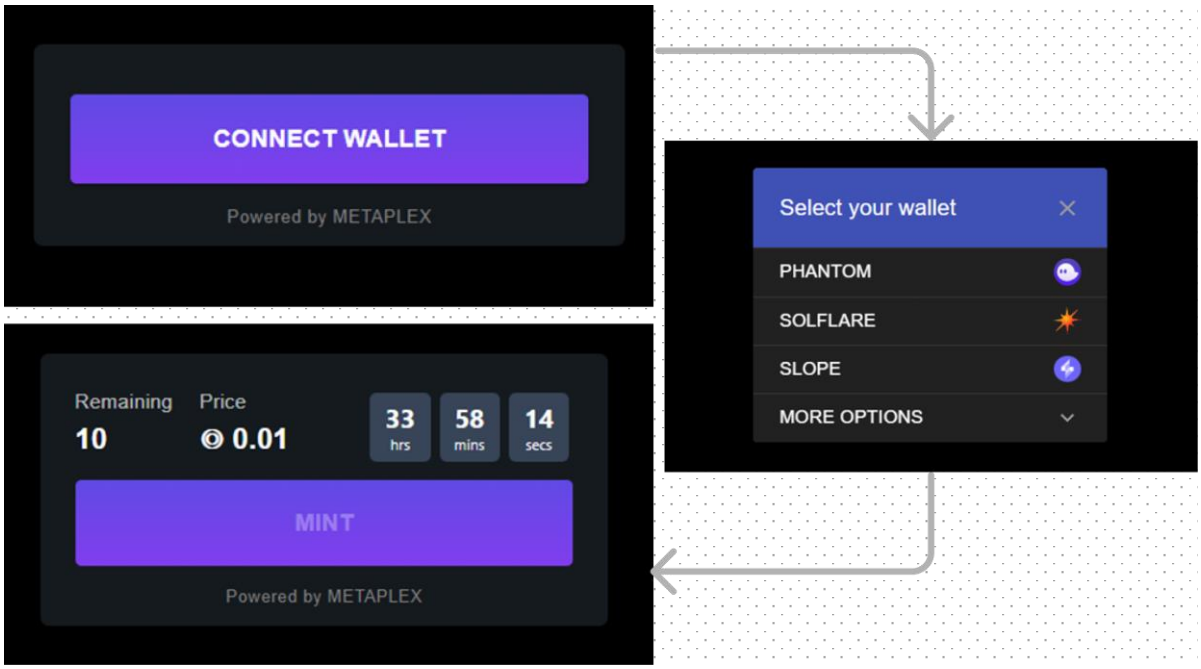
Command 16 Install modules and packages and launch the application

```
yarn install
```

```
yarn start
```

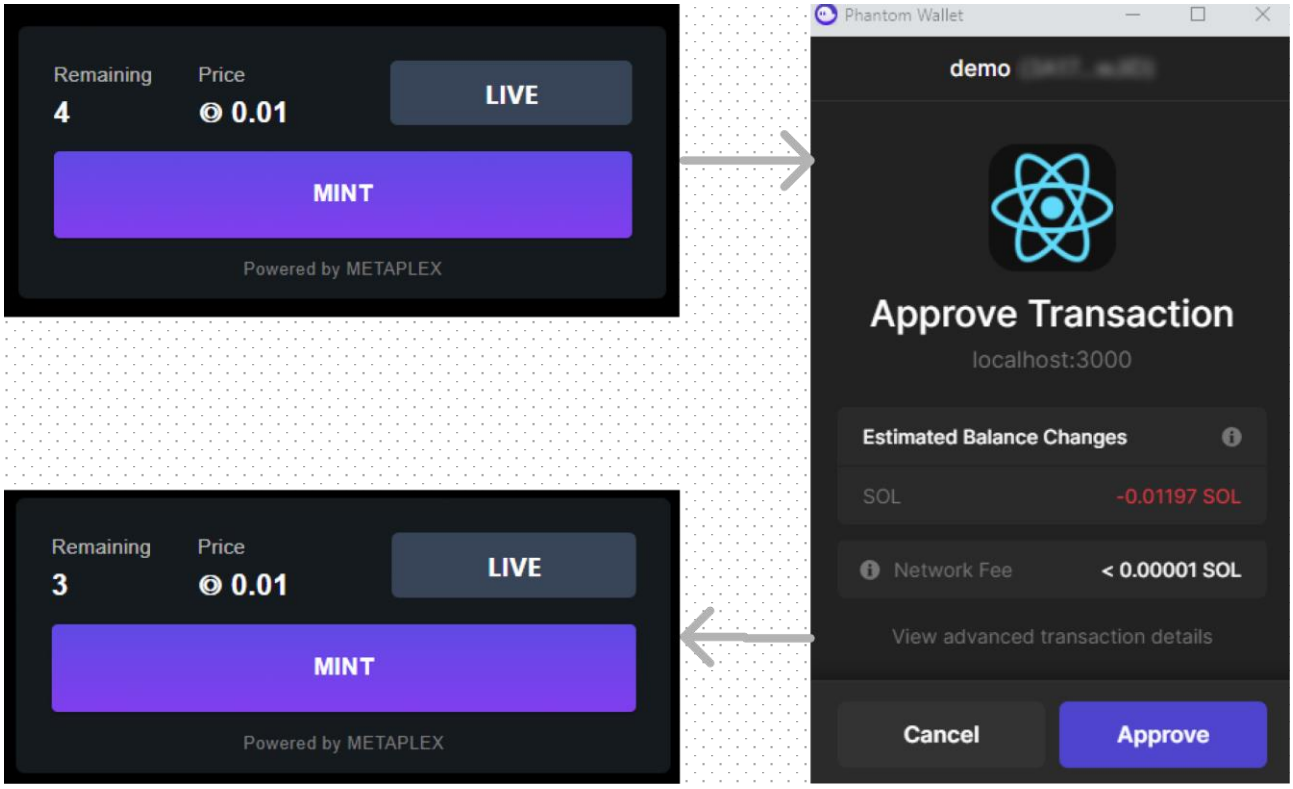
At this point, the UI has several elements as Figure 14 shows. It has a “Connect Wallet” button, by clicking the button, the wallet type can be selected on a pop-up window. After connecting the wallet, the “Mint” button will be visible with additional information about the collection, such as the number of remaining NFTs, the price of one NFT and a countdown indicating how much time has left until the NFTs can be minted. This information is retrieved from the deployed “config.json” of the created Candy Machine, which is identified by the id configured in the “.env” file.

Figure 14 Features



When the launch date has come, the mint button will be enabled, and the countdown will disappear, and the Mint button will be enabled. Once the Mint button is clicked, the Phantom wallet will ask the user to confirm the transaction and show the estimated price. On successful mint, the number of NFTs left will decrease by one as Figure 15 shows.

Figure 15 Minting



5.4.1 Customizing the web application

At this point, the most essential functionalities are implemented with the help of the predefined UI elements of The Candy Machine. The goal is to make further improvements and add new features to the application. For example, users can connect their wallets, but they cannot disconnect them so that could be added. The official repository of Solana has a component called “WalletMultiButton”, which can be used for this purpose. It can be installed using the command from Command 17.

Command 17 Install WalletMultiButton component

```
yarn add @solana/wallet-adapter-material-u
```

It can be imported as the following snippet shows. It has to be nested into different context providers, which are responsible for establishing the connection, which requires several additional packages to install, that can be found in the repository of Solana at the following link: https://github.com/solana-labs/wallet-adapter.

Program code 4 Candy Machine configuration

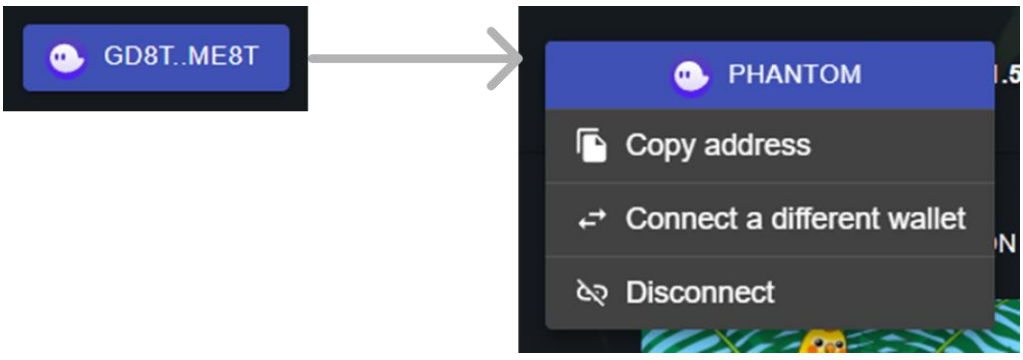
```
import { WalletMultiButton } from "@solana/wallet-adapter-material-ui";
```

Program code 5 Example of rendering the WalletMultiButton component

```
    return (
        <ConnectionProvider endpoint={endpoint}>
            <WalletProvider wallets={wallets} autoConnect>
                <WalletModalProvider>
                    <WalletMultiButton />
                    <WalletDisconnectButton />
                    { /* App components*/ }
                </WalletModalProvider>
            </WalletProvider>
        </ConnectionProvider>
    );
```

The button shows the shortened wallet address. By clicking on it, a dropdown gives options for disconnecting the wallet, copying the address, or connecting a different wallet as Figure 16 illustrates.

Figure 16 WalletMultiButton UI



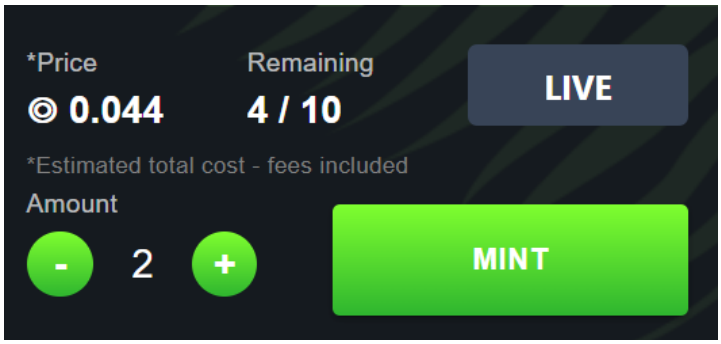
Once the wallet is connected, the balance can be also retrieved and displayed. The app can make HTTP requests to Solana nodes to receive information about the wallet by passing its public key as Program code 6 shows. The balance is expressed in lamports, which needs to be converted to SOL.

Program code 6 Getting balance information

```
(async () => {
      if (wallet?.publicKey) {
        const balance = await props.connection.getBalance(wallet.publicKey);
        setBalance(balance / LAMPORTS_PER_SOL);
      }
)
```

The minting functionality works only for one mint at a time so it would be nice to enable users to mint multiple NFTs at once. A component is created for this purpose, which consists of buttons, that can be used to set the value of the counter – by decreasing or increasing the value. The minimum value is 1 and the maximum value of the counter depends on the number of remaining NFTs. This component receives information about the number of remaining NFTs and the price of one from its parent component and then the counter component sends back the value of the counter that has been set and the calculated cost based on the counter. At this point, the minting panel look as it is illustrated in Figure 17 – when the minting is enabled.

Figure 17 Minting panel UI



The next improvement is to display the NFTs from the collection owned by the user whose wallet is connected. The wallet address can be used also to retrieve the metadata of the NFTs owned by the wallet user. The URI from that metadata – highlighted on Program code 7 – holds the metadata of the corresponding NFT in a JSON object, which can be retrieved by a GET request to get the URL that points to the image representation of an NFT stored on Arweave.

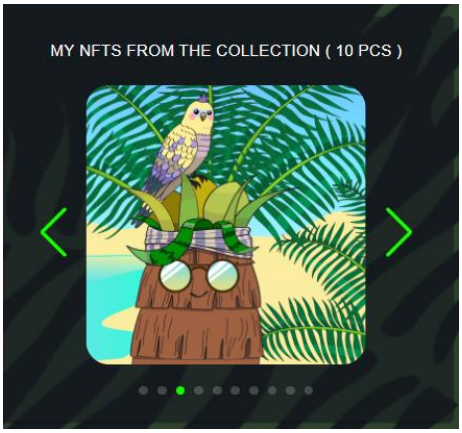
Program code 7 Metadata holding URI to retrieve NFT image and metadata


```
▼Array(20) i
  ▼0: MetadataData
      collection: undefined
    ▼data: MetadataDataData
      ▶creators: (2) [Creator, Creator]
        name: "Treedom #8"
        sellerFeeBasisPoints: 1000
        symbol: "T"
        uri: "https://arweave.net/[redacted]"
      ▶[[Prototype]]: Data
      editionNonce: 251
      isMutable: 1
      key: 4
      mint: "[redacted]"
      primarySaleHappened: 1
      tokenStandard: 0
      updateAuthority: "3A17iAjFAQCuAU81QjSkmyvNrvq95RuTXsUb4M7DwJiD"
      uses: undefined
    ▶[[Prototype]]: Data
```


These URIs are also included in the “devnet-temp.json” file located in the previously mentioned “.cache” folder that has been created when the NFTs have been deployed to Arweave. Certain attributes of the metadata, that have been set in the configuration of the NFT generating code can be used to identify the NFTs in the wallet that belongs to the collection the website was made for.

Then a customized carousel component called “Swiper” is used to display the images (NFTs) on the webpage. If the mint has been completed, the page reloads and shows the newly acquired NFTs as well. Figure 18 shows what it looks like.

Figure 18 NFTs of the collection owned by the wallet user rendered on the page



The next step is to categorize the minted NFTs so their values will be determined by the categorization. As it was mentioned in chapter 5.3, the rarity can determine the value of the NFT. But there are multiple features to consider, and their combinations determine the overall rarity of each NFT. The rarity assessment is important information for the buyer; therefore, it should be displayed for each minted NFT which belongs to the collection. To be able the determine the rarity of each feature, the metadata of the NFTs will be needed which holds the attributes that have been configured during the compilation of the NFTs. But this time this metadata needs to be retrieved from Arweave. It can be retrieved from the same URI which was used for the GET request previously to get the image itself for the NFT. Program code 8 shows an example of the received metadata.

Program code 8 Metadata retrieved by GET request


```
▼attributes: Array(10)
  ▶0: {trait_type: 'Background', value: 'background02R'}
  ▶1: {trait_type: 'Branches', value: 'branches'}
  ▶2: {trait_type: 'Coconuts', value: 'coconuts'}
  ▶3: {trait_type: 'Parrots', value: 'captainL'}
  ▶4: {trait_type: 'Leafs', value: 'leafs'}
  ▶5: {trait_type: 'Monkeys', value: 'albinosmilingR'}
  ▶6: {trait_type: 'Treetrunk', value: 'treetrunk'}
  ▶7: {trait_type: 'Glasses', value: 'glassesL08'}
  ▶8: {trait_type: 'Headscarf', value: 'headscarfL06'}
  ▶9: {trait_type: 'Mouths', value: 'mouth01'}
    length: 10
  ▶[[Prototype]]: Array(0)
  description: "This is a tree collection with funny parrots and monkeys"
  edition: 5
  external_url: "https://www.treedomnft.com/"
  image: "https://www.arweave.net/Sm4PjoZmTN1U6oQC1eK6SmHda-Cmj8ULCgb4NhMjvVc?ext=png"
  name: "Treedom #6"
▶properties: {files: Array(1), category: 'image', creators: Array(1)}
  seller_fee_basis_points: 1000
  symbol: "TDN"
```


It is important to mark the layers such that it indicates the rarity. Each feature on an image gets a score based on the combination. Although it might be that two image gets the same score still one is more valuable because certain features are considered more special than the other so that must be taken into account as well during the categorization.

The code presented in Program code 9 retrieves first all NFTs from the connected wallet. This metadata is filtered so only the NFTs which belong to the collection will be processed when

iterating through the filtered data and calling a function which does the GET request using the URI that is passed to the function. The “NFTsfromCollection” variable will have a list of objects holding the image and rarity related information for each NFT that was found in the wallet and belongs to the collection.

Program code 9 Filtering and processing metadata of NFTs from connected wallet

```
const collectNftsFromWallet = async () => {
  const nftsmetadata = await Metadata.findDataByOwner(
    props.connection,
    anchorWallet!.publicKey
  );
  let NFTsfromCollection = Promise.all(
    nftsmetadata
      .filter(
        (nft) => nft.data.name.includes("Treedom") && nft.data.symbol == "TDN"
      )
      .map((nftData) => getNFTs(nftData.data.uri))
  ).then((value) => {
    return value;
  });
  setNFTs(await NFTsfromCollection);
};
```

The function in Program code 10 uses the URI to get the metadata from which the image and the attributes can be retrieved. These attributes are sent to another function which evaluates the NFT based on those attributes.

Program code 10 Using URI for GET request to get the metadata of the current NFT

```
const getNFTs = async (uri: any) => {
  let response;
  try {
    response = await axios.get(uri);
  } catch (error) {
    console.log(error);
  }
  let rarity = await analyseAttributes(response?.data.attributes);
  let image: string = response?.data.image;
  return { image, rarity };
};
```

The function in Program code 11 initializes a dictionary where scores will be stored for features, such as "Background", "Parrots", "Monkeys", "Glasses" and "Headscarf". The scores are on a scale of 0 to 3 in general, but those vary slightly by feature depending on the number of variations in one feature. The rare attributes include “R” in their names and legendary ones include “L”. “Blank” can be in the case of monkey or parrot which means that those do not appear on the image if it is

marked as “blank” which get score 0. In all other cases, the minimum score is 1. These scores are summarized using a function which receives an array of scores as a parameter from the dictionary. Then these scores are used for the final assessment of the NFT which is not included in the Program code 11 as it is too long and not relevant to understanding the overall process.

Program code 11 NFT rarity assessment

```
const analyseAttributes = async (attributes: any) => {
  let traits = ["Background", "Parrots", "Monkeys", "Glasses", "Headscarf"];
  let scores: any = {};
  let category = "";
  await attributes.map((nft: any) => {
    if (traits.includes(nft.trait_type)) {
      if (nft.value === "blank") {
        scores[nft.trait_type] = 0;
      } else if (nft.value.includes("R")) {
        scores[nft.trait_type] = 2;
      } else if (nft.value.includes("L")) {
        scores[nft.trait_type] = 3;
      } else {
        scores[nft.trait_type] = 1;
      }
    }
  });
  let sum = getSum(Object.values(scores));

  --evaluation based on scores and custom rules set--

  return { scores, category };
};
```

The function returns the scores and the category as the result of the assessment. The scores will be needed to display the scores given to all features. By hovering the mouse over the image, the rarity and the scores are displayed on the image as Figure 18 shows. Only the legendary NFTs have maximum score. Below it, the categories are: “super rare”, “rare”, “above average” and “average”.

Figure 19 Displaying rarity on the image overlay

6 Results

The thesis provides solutions responding to the research questions, that can serve as a guide for developers or anyone who is interested in creating their own NFT project. The knowledge base helps to understand better why blockchain has a significant role in decentralization and how NFTs fit into this context. The thesis helps to understand what NFTs are, how they differ from other tokens and why they are so popular these days. The comparison of Ethereum and Solana networks gives an idea of what these networks can offer, how they work and what are the aspects to consider when choosing a network.

Solana is quite a new network, and it is continuously being improved and developed. Thus, some configurations might change over time and the official documentation available might lack certain updates, therefore it requires extra time and research to find solutions and workarounds on certain issues that people are not aware of yet in a wide range. Having that said, the Solana based Metaplex protocol provides very powerful tools such as The Candy Machine that significantly improved the effectiveness of the development work. By customizing the base code of the application, new features were added, which helped me to understand better the code base of the basic UI included in the Metaplex repository.

HashLips Art Engine makes very efficient the generation of NFTs with their corresponding metadata. The NFTs can be deployed to a decentralized data storage called Arweave.

The actual result of the thesis is a website which is deployed on the devnet and running on the local server, meaning that the NFTs cannot be bought for real. The users can connect their wallets, and then mint, if the launch date has come. Meanwhile, a countdown indicates the time left until the exact launch date. The application can be accessed from GitHub at the following link: https://github.com/Eszter96/NFT_minting_site. The project could be taken to a new level by deploying the application to a remote server and the NFT collection to the mainnet, so the NFTs could be bought for real tokens.

7 Summary

The evolving blockchain technology and its popularity motivated me to dive deeper into this field. The demand for blockchain developers is continuously increasing and, in my opinion, this tendency will last for a long period of time. In the blockchain world, non-fungible tokens have a huge potential, because of their diversity which derives from the fact that almost everything can be turned into NFTs. I was interested in how one can successfully create an NFT collection, and then create a website where these NFTs can be minted.

The research questions were practice-based therefore they were meant to be answered in the implementation part of the thesis by providing a description of the process of how to generate an NFT collection programmatically, and how to create a website which can be used to buy those NFTs based on best practices.

The development work helped me to get a better understanding of the workflow of setting up a working environment and launching my own NFT collection. By observing the source code of the tools that I was using, I got a better insight into advanced solutions and best practices. During the development of the project, it became clear that developing a code for NFT generation and minting requires advanced skills due to the complexity of the configurations that the deployment and the transactions require. Therefore, the thesis puts high emphasis on powerful tools, such as the HashLips Art Engine and The Candy Machine, that can genuinely support the development work in the case of NFT projects by providing a strong foundation to build upon.

As there are many concerns about the ecological impact of the computing power that it takes to run a blockchain, the collection could have a use case that could mitigate those harmful effects. For example, the NFTs could represent real trees, which could be adopted by buying an NFT from the collection.

References

Avyan. (n.d.). *Solana vs Ethereum: A Detailed Comparison*. Retrieved from CoinMarketCap: https://coinmarketcap.com/alexandria/article/solana-vs-ethereum-a-detailed-comparison

Barker, C. (2021). *Getting Started with Solana Development*. Retrieved from Solana: https://solana.com/news/getting-started-with-solana-development

BuiltIn. (n.d.). *Blockchain Technology Defined*. Retrieved from Built In: https://builtin.com/blockchain

Coinbase. (n.d. -a). *How to set up a crypto wallet*. Retrieved from Coinbase: https://www.coinbase.com/learn/tips-and-tutorials/how-to-set-up-a-crypto-wallet

Coinbase. (n.d. -b). *What is a non-fungible token (NFT)?* Retrieved from Coinbase: https://www.coinbase.com/learn/crypto-basics/what-are-nfts

CoinMarketCap. (n.d.). *Countries Which Allow Cryptocurrency As Legal Tender*. Retrieved from CoinMarketCap: https://coinmarketcap.com/legal-tender-countries/

Cointelegraph. (n.d.). *A beginner's guide to the different types of blockchain networks*. Retrieved from Cointelegraph: https://cointelegraph.com/blockchain-for-beginners/a-beginners-guide-to-the-different-types-of-blockchain-networks

Cookbook, S. (2022a). *Accounts*. Retrieved from Solana Cookbook: https://solanacookbook.com/core-concepts/accounts.html#facts

Cookbook, S. (2022b). *Programs*. Retrieved from Solana Cookbook: https://solanacookbook.com/core-concepts/programs.html#facts

ELEV8. (2020). *8 Things You Should Know About Crypto Networks*. Retrieved from Elev8: https://www.elev8con.com/8-things-you-should-know-about-crypto-networks/

Frankenfield, J. (2021a). *Bitcoin Definition*. Retrieved from Investopedia: https://www.investopedia.com/terms/b/bitcoin.asp#toc-who-is-satoshi-nakamoto

Frankenfield, J. (2021b). *Fungibility*. Retrieved from Investopedia: https://www.investopedia.com/terms/f/fungibility.asp

Frankenfield, J. (2022a). *Cryptocurrency*. Retrieved from Investopedia: https://www.investopedia.com/terms/c/cryptocurrency.asp

Frankenfield, J. (2022b). *Smart Contracts*. Retrieved from Investopedia: https://www.investopedia.com/terms/s/smart-contracts.asp

GeeksForGeeks. (2022). *Advantages and Disadvantages of Blockchain*. Retrieved from GeeksForGeeks: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-blockchain/

Hayes, A. (2022). *Blockchain Explained*. Retrieved from Investopedia: https://www.investopedia.com/terms/b/blockchain.asp

IBM. (2021). *Opaque Data Types*. Retrieved from IBM Documentation: https://www.ibm.com/docs/en/informix-servers/12.10?topic=edt-opaque-data-types

IBM. (n.d.). *What is blockchain technology?* Retrieved from IBM: https://www.ibm.com/topics/what-is-blockchain

Investopedia. (n.d). *Blockchain*. Retrieved from Investopedia: https://www.investopedia.com/blockchain-4689765

Modderman, G. (2022). *NFT scams: How to avoid becoming a victim*. Retrieved from Cointelegraph: https://cointelegraph.com/explained/nft-scams-how-to-avoid-becoming-a-victim

Oridoc. (2021). *What is NFT / non-fungible token?* Retrieved from Oridoc: https://www.oridoc.com/what-is-nft-non-fungible-token/

Radocchia, S. (2018). *3 Innovative Ways Blockchain Will Build Trust In The Food Industry*. Retrieved from Forbes: https://www.forbes.com/sites/samantharadocchia/2018/04/26/3-innovative-ways-blockchain-will-build-trust-in-the-food-industry/?sh=27483dfe2afc

Reinicke, C. (2021). *1 in 10 people currently invest in cryptocurrencies, many for ease of trading, CNBC survey finds*. Retrieved from CNBC: https://www.cnbc.com/2021/08/24/1-in-10-people-invest-in-cryptocurrencies-many-for-ease-of-trading.html

Schmidt, J. (2022). *What Is An NFT? Non-Fungible Tokens Explained*. Retrieved from Forbes: https://www.forbes.com/advisor/investing/nft-non-fungible-token/

Sharma, R. (2021). *Bit Gold*. Retrieved from Investopedia: https://www.investopedia.com/terms/b/bit-gold.asp

Sharma, R. (2022). *Non-Fungible Token (NFT) Definition*. Retrieved from Investopedia: https://www.investopedia.com/non-fungible-tokens-nft-5115211

Solana. (n.d. -a). *Accounts*. Retrieved from Solana Documentation: https://docs.solana.com/developing/programming-model/accounts

Solana. (n.d. -b). *Native Programs*. Retrieved from Solana Documentation: https://docs.solana.com/developing/runtime-facilities/programs#system-program

Solana. (n.d. -c). *Solana Clusters*. Retrieved from Solana Documentation: https://docs.solana.com/clusters

Solana. (n.d. -d). *Transactions*. Retrieved from Solana Documentation: https://docs.solana.com/developing/programming-model/transactions

Solana. (n.d. -e). *Token Program*. Retrieved from Solana Program Library: https://spl.solana.com/token

Solidity. (n.d.). *Solidity*. Retrieved from Solidity: https://docs.soliditylang.org/en/v0.8.13/

Trends, M. (2022). Retrieved from Analytics Insight: https://www.analyticsinsight.net/ethereum-or-solana-which-is-better/#

Yaffe, L. (2018). *Stateful vs. Stateless Blockchain Contracts*. Retrieved from Medium: https://medium.com/coinmonks/stateful-vs-stateless-blockchain-contracts-bcd1b0c25ff

Yakovenko, A. (2019). *Tower BFT: Solana's High Performance Implementation of PBFT*. Retrieved from Medium: https://medium.com/solana-labs/tower-bft-solanas-high-performance-implementation-of-pbft-464725911e79

Annex 1: Material management plan

Notes for the content and practical part of my thesis were created and stored in OneNote, which is available from the following link: https://hameenamk-my.sharepoint.com/:o:/g/personal/eszter19100_student_hamk_fi/Ejjc03Aru3dPorL1ZOoFeZsBQivRhZDvUWVl6BM3nDtlFw?e=cJI5Uf. A copy of its content is saved in Word format to my computer. The codebase for the application is available on GitHub and it can be accessed through WSL on my computer. The layers for the NFTs and other assets for the website are stored locally.