REVOLUTIONIZING EVENT TICKETING: A SECURE AND TRANSPARENT APPROACH USING NFTS AND BLOCKCHAIN

Theodor Tudose[1], Costin Carabaș[2], Nicolae Țăpuș[3]

The traditional approach to ticket management facilitates access to events such as conferences, workshops, concerts, seminars, films, with disadvantages such as proneness to fraud and forgery. Moreover, they do not benefit from the oversight of secondary markets, which could lead to more expensive tickets through fraud. Digital tickets, in the form of QR, do not solve the aforementioned problems. Furthermore, they have no sentimental value after the event, while paper tickets might be considered by some to be a collector's item, they are not really eco-friendly and can easily get damaged.

This paper proposes a solution based on NFTs (Non-Fungible Tokens) and Blockchain technology. Because an NFT is unique, transparent, and cannot be counterfeited or duplicated, it can be tracked on the secondary market and easily regulated through the use of smart contracts. Through its design and features, an NFT can be used as a form of admission to the event and can enhance the attendee experience by offering special rewards or access to exclusive content

Keywords: blockchain, NFT, smart contracts, e-tickets

1. Introduction

In recent years, there has been an exponential surge in the popularity of Blockchain and Crypto Currencies.[1] Blockchains validate transactions, digital identities, money transactions, and secure voting systems. Because of its constant development, Blockchain technology has given rise to a multitude of decentralized applications and smart contracts, which have significantly transformed the landscape of online business practices and brought the Web3 concept closer to our everyday lives. The blockchain we will use throughout this paper, where we will explore the limitations of current implementations and evaluate the proposed solution, is MultiversX.

[1]MSc Student, National University of Science and Technology POLITEHNICA Bucharest , Romania, e-mail: theodor.tudose@stud.acs.upb.ro

[2]Lecturer, National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: costin.carabas@upb.ro

[3]Professor, National University of Science and Technology POLITEHNICA Bucharest, Romania

An important characteristic of blockchains is their immutability. Once a transaction is recorded on the blockchain, it becomes impossible for any network participant to alter its outcome.

1.1. Problem Statement and Objectives

Traditional ticketing systems have encountered several **limitations** that this paper seeks to resolve. `Ticket Fraud and Counterfeiting`: Due "to the emergence of scalpers and secondary markets"[2] Traditional paper or digital tickets are susceptible to fraud and counterfeiting, as they can be duplicated or forged easily. This leads to revenue loss for event organizers and attendees. `Opaque Ticket Ownership`: In traditional systems, it's often challenging to establish the legitimate owner of a ticket, especially when tickets are resold or transferred. `Verification of Ticket Authenticity`: Event attendees and organizers often struggle to verify the authenticity of tickets, leading to delays and confusion at entrances. `Limited Ticket Transferability`: Traditional ticket systems might have restrictions on transferring tickets between individuals due to logistical challenges or rules imposed by organizers. `Dynamic Ticket Features`: NFT ticketing systems offer the potential for introducing dynamic features to tickets. For instance, the system can be programmed to automatically offer early bird discounts, loyalty rewards, or access to exclusive content. These programmable conditions enhance the overall ticketing experience for both event organizers and attendees. `Traceability and Accountability`: The blockchain ledger in the NFT ticketing system provides an audit trail of all ticket-related transactions. This traceability ensures accountability throughout the lifecycle of a ticket, from initial purchase to final entry into the event.

The main focus of this paper is to provide a solution to the problems associated with conventional ticketing platforms, by minting and selling tickets through the transparency and security of a smart contract.

Moreover, we developed a feature that can help the organizers of free events to limit the absentees, thus limiting unnecessary expenses. When a participant signs up for a free event, he locks up some tokens(fee for absence) in a smart contract. The tokens will be unlocked only if the organizers acknowledge its presence at the event.

The outcomes of this project include a stable, fault-tolerant, efficient and user-friendly ticketing platform. The application also provides room for incorporating additional functionalities. Furthermore, the paper also includes a thorough performance evaluation for varying load scenarios and a detailed report outlining the design and implementation procedures.

Overall, this paper aims to contribute valuable insights, knowledge, and a practical solution to the domain of event ticketing, demonstrating the benefits of NFTs and blockchain technology in revolutionizing ticketing systems.

2. State of the art

Non-fungible tokens or NFTs, are unique digital assets that represent ownership or proof of authenticity of a specific item, artwork, collectible, or piece of content on a blockchain network. Unlike traditional cryptocurrencies, like Bitcoin, Ethereum or Solana, which are interchangeable and can be exchanged for one another, NFTs are indivisible and cannot be exchanged on a one-to-one basis due to their unique characteristics. Each NFT is distinct and holds a specific value based on its uniqueness and scarcity, making it suitable for representing and trading one-of-a-kind or limited-edition digital creations. In addition to the features of a fungible token, an NFT contains metadata and unique attributes, as can be seen in the figure below.[3]

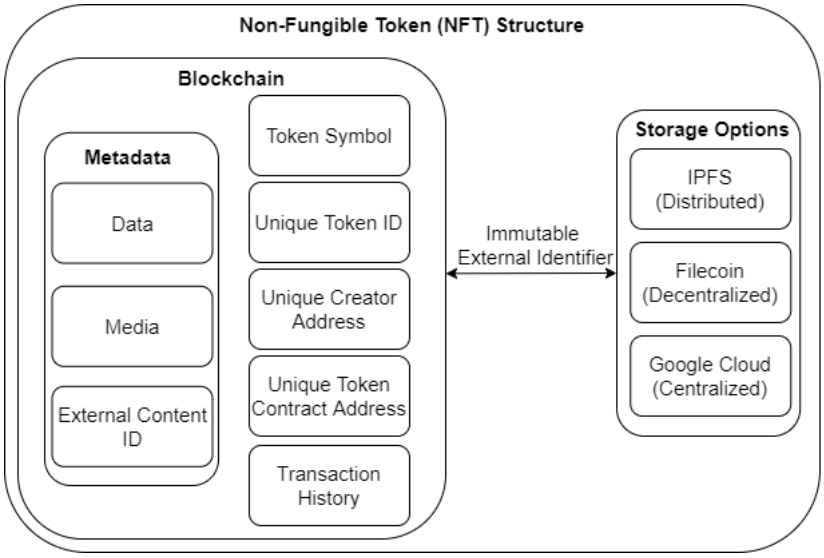

Fig. 1. Structure of an NFT[4]

Figure 1 illustrates the structure of an NFT and its key components. The blockchain serves as the foundational element, housing distinctive features that set each NFT apart. Each NFT is associated with a unique contract address within a smart contract that cannot be replicated, which is necessary to ensure the authenticity of the smart contract users interact with. Moreover, the deployer of the smart contract is identified by a unique address, which is referred to as the Unique Creator Address. A specific token symbol might be designated, though the distinct token contract address remains essential to distinguish the NFT from other collections. Furthermore, NFTs are individually recognized by unique token IDs, generated during minting. The transaction history of each NFT smart contract is permanently stored on the underlying blockchain, which can be used to track and trace NFTs. Lastly, NFT metadata can be stored on-chain or off-chain.

NFTs are created and transferred through a process known as minting. Minting involves the creation of a unique token on a blockchain network, typically using smart contracts. If we want to create an NFT collection we have to interact with a platform or marketplace that supports this operation, providing the necessary information and metadata about the asset, including its name, description, and any additional attributes. Once the minting is complete, the NFT is assigned a unique identifier, representing ownership and authenticity of the asset, and becomes part of the blockchain network.

To transfer an NFT from one party to another, the owner initiates a transfer and create a transaction on the blockchain. In this transaction, the owner specifies the recipient's wallet address and provides details about the NFT being transferred. The blockchain network then verifies the transaction and updates the ownership records accordingly. Through this process, transparency, immutability, and security are ensured in the transfer of NFTs.

Current event ticketing solutions have made significant advancements, leveraging technology to enhance the ticketing experience for both event organizers and attendees. Mobile ticketing has become increasingly popular, allowing us to conveniently access our tickets via our smartphones, eliminating the need for physical tickets. Personalized ticketing options enable us to choose specific seats or packages tailored to their preferences. Furthermore, event ticketing solutions now offer features like real-time ticket availability updates, interactive seating charts, and integrated payment gateways for smooth transactions. In paper [5], authors demonstrate the usefulness of NFTs to tokenize digital goods, prevent fraud, and improve control over secondary market transactions. Moreover, authors [6] propose a `fair price ticketing curse` which occurs when an event organizer sells tickets at prices that do not correspond to underlying demand conditions and does not want resellers to profit from resale opportunities.

Integration with event management systems enables organizers to streamline the entire event planning process, from ticket sales to attendee registration and data management. These solutions have significantly improved the efficiency, accessibility, and overall experience of purchasing and managing event tickets, benefiting both event organizers and attendees alike.

Authors [7] propose a solution by implementing radio-frequency identification (RFID) and blockchain technology to optimize the services applied in the multi-sport event.

3. Designing a Secure Event Ticketing System

This section describes the solution that addresses the problem by presenting its architecture and key components.

Figure 2 briefly explains the interaction between a user and the Smart Contract. A user can be either an event organizer or a potential participant. Through the GUI platform, each of them can call certain endpoints of the

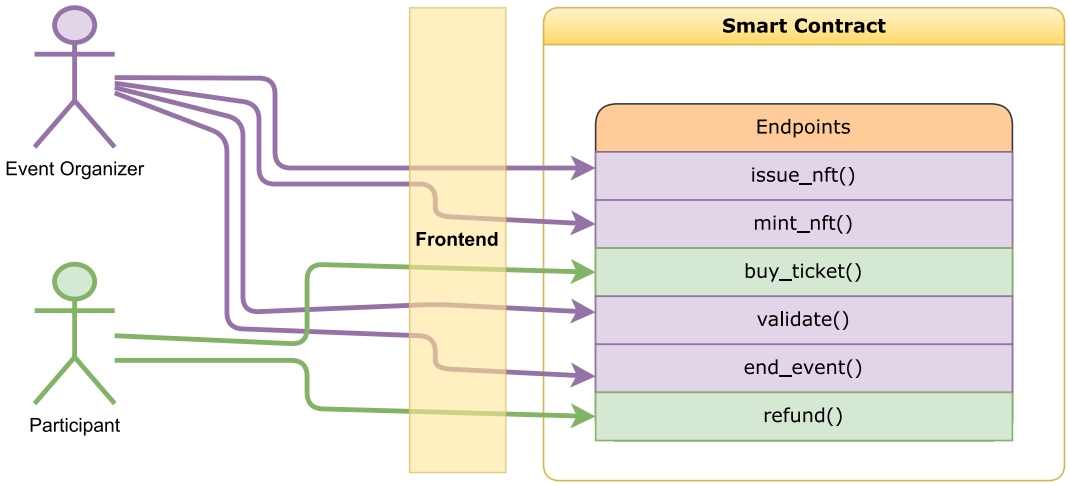

FIG. 2. Interaction diagram between user and Smart Contract

smart contract. An event organizer can create an event, in which case he will create an NFT collection and add the "tickets" to it. He can validate a participant's ticket when he attends the event. At the end of the event, the end_event endpoint will be automatically triggered. A participant can buy a ticket and if he can no longer attend the event and certain criteria established by the organizer are met, he can refund his ticket.

A favorable scenario for this application would go as follows: EO (Event Organizer) wants to organize a free admission conference (CON).

(1) EO creates the collection while specifying certain details like the name of the event, a token name for the collection, the value of the refundable attendance fee (F), the number of tickets he wants to mint, the start date and end date of the event, etc.
(2) The Smart Contract (SC) will mint the specified number of NFTs and assign them to EO's wallet.
(3) P (Participant) will access the platform and specify he wants to attend CON.
(4) P wants to buy a ticket and if there are any tickets available for CON, a value of F EGLD will be transferred from P's wallet to SC and an NFT will be transferred from EO's wallet to P's wallet.
(5) P presents the ticket to attend the event and EO marks him as an attendant.
(6) EO ends the event and because P was validated, he receives his EGLD back from SC.
(7) P keeps the ticket in his wallet as a collectible.

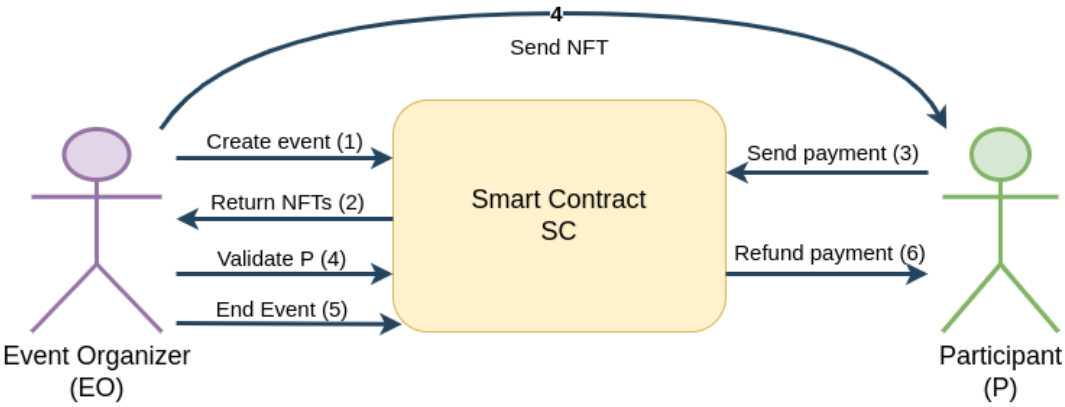

FIG. 3. The flow of an ideal scenario

4. Implementing NFT-Based Ticketing on the Blockchain

Our solution consists of 2 main components: the *Smart Contract*, which facilitates the user-blockchain interaction and also serves as a backend for this project, being the core component of our application, and the *frontend*, the component with which the users will directly interact.

The Smart Contract uses `Rust` and the MultiversX framework. Rust is a systems programming language known for its focus on safety, performance, and concurrency. It aims to provide a balance between low-level control over hardware resources and high-level abstractions that prevent common programming errors like null pointer dereferences and race conditions. Rust's ownership system and borrowing rules enable memory safety without the need for a garbage collector, making it suitable for tasks where performance and reliability are crucial, such as systems programming, embedded development, and creating reliable software with concurrent operations.[8]

The MultiversX framework is a collection of libraries, templates, and guidelines that we use to streamline the development process. The `multiversx_sc` library contains the smart contracts development kit and provides a complete smart contract build solution.

We opted for `TypeScript` to power the front end, a strategic decision that perfectly aligns with the advanced capabilities of the MultiversX Framework. TypeScript's integration within the MultiversX ecosystem offers an array of libraries that facilitate the User - Smart Contract interaction through REST API calls. Typescript blends the benefits of JavaScript with static typing which boosts code reliability, minimizes errors, and makes it a perfect tool for the frontend part of the project.

4.1. Smart Contract features

The first step in transacting tickets is to create the tickets themselves. To generate NFTs we will use 2 endpoints of the Smart Contract: `issue_nft` and

`mint_nft`. The former will be used to create an NFT collection with a certain name and ticker. When issuing a collection, the combination of the name and ticker helps to ensure the uniqueness of the collection and to distinguish one collection from another. Apart from the parameters required to create the collection, this function must also receive the event-specific p arameters. These are the event price, the start date, the end date, and the last refund date. If the organizer does not wish for the ticket to be refundable, then he can skip this last step. And the refund date will be automatically set to the current date. When the method is called it performs the following steps:

(1) It retrieves the EGLD amount paid by the organizer in order to start the creation of the collection.
(2) The ticket price and the other event-specific details are stored inside separate Storage Mappers.
(3) The token name is stored inside a Storage Mapper.
(4) The token (collection) is issued using the following parameters:
    - EsdtTokenType::NonFungible specifies that the token is non-fungible.
    - The payment amount specifies the price of issuing the collection.
    - The token name and ticker are specified
    - The initial nonce is set to 0. This specifies that the nonce of the first NFT minted within this collection will be 0.
    - No additional NFT roles are specified
(5) The token is stored inside an Array type Storage Mapper.

The `mint_nft` endpoint will be used to create an NFT inside the collection and transfer it to the event organizer's wallet address. When the method is called it performs the following steps:

(1) Acquires the organizer's address and the current block epoch.
(2) Another function is used to create the NFT token and acquire the transaction details.
(3) The token's attributes are retrieved based on the token's nonce.
(4) Associates the token name with the organizer's address in a Storage Mapper in order to create a database with who created each token.
(5) Transfers the created token to the organizer's wallet.

Figure 4 shows the flow of the solution when creating a collection.

Creating these two endpoints was one of the first milestones of this application as minting NFTs represents the main functionality of the platform.

We implemented a function that allows participants to purchase NFTs, via the `buy_nft` endpoint. This endpoint sends a random NFT from the intended collection, from the Organizer's wallet to the caller's wallet of the function (the Participant), and transfers the EGLD from the caller's wallet to the smart contract's wallet. Figure 5 shows the flow of a participant buying a ticket.

The `end_event` endpoint is triggered at the end of the event, specified by the Organizer in the first step of the process. The function iterates through

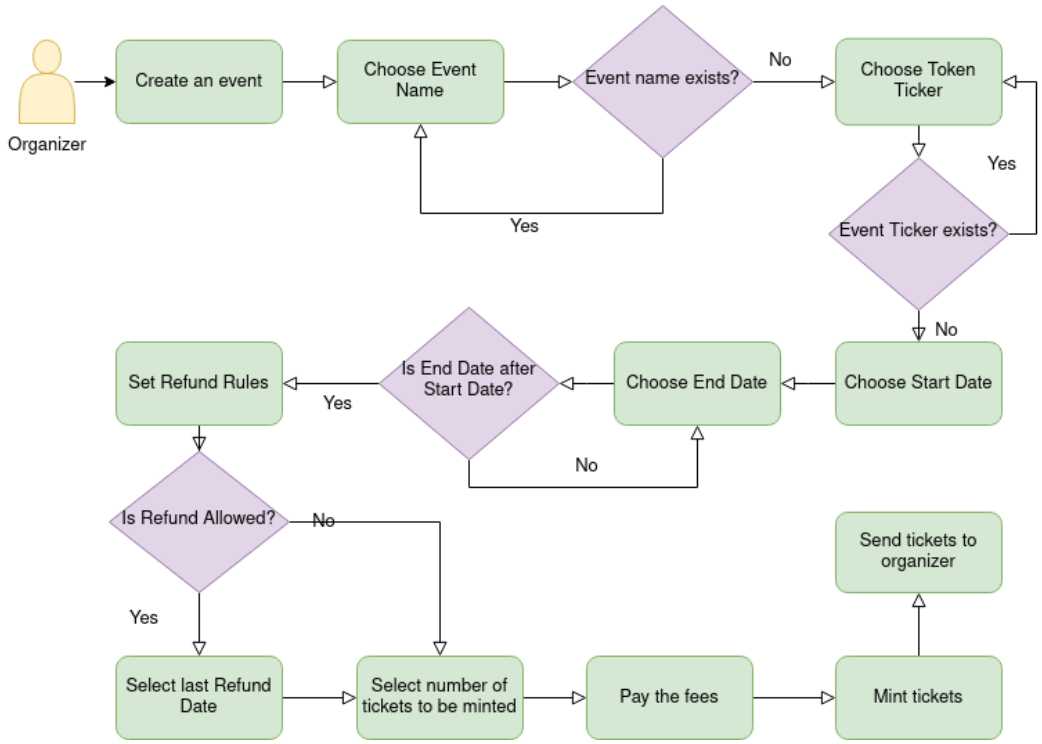

FIG. 4. The flow of minting NFTs

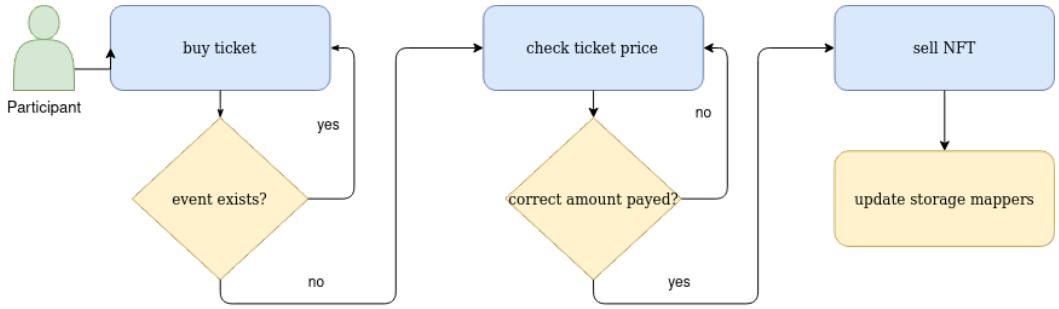

FIG. 5. The flow of buying a NFT

all of the participants and if the participant is marked as validated, then the smart contract returns refunds the EGLD paid and lets him keep the ticket as a potential collectible. If the participant has not been validated, then the EGLD paid by him is transferred to the organizer. Figure 6 shows the flow of ending the event.

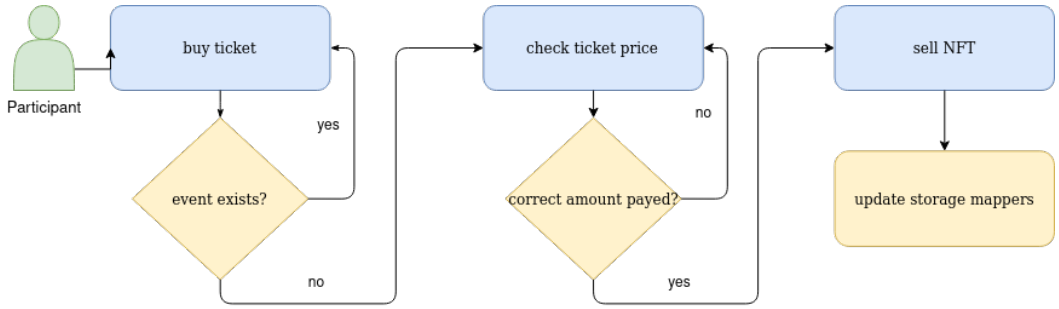

FIG. 6. The flow of ending the event

In the case of a paid event, like a Concert, no participant will be marked as validated so, at the end of the event, all the EGLD paid for the tickets will go to the Organizer.

The `refund` Endpoint is called by a Participant. If the Organizer chose a refund date and that date has not yet been reached, then the Participant is eligible for a refund. The function checks if the Participant has sent the ticket, if the return date has not passed, and then transfers the EGLD from the Smart Contract to the Participant. The last step executed by this function is to update the Storage Mappers and to remove the participant from all of the structures.

The `cancel_event` Endpoint is meant to be used if the Organizer can no longer host the event. In this case, the Organizer will call the Endpoint, specifying the event name as a parameter and after the function checks that the address that called the contract is the real organizer of the event, it will return all the EGLD from the Smart Contract to the Participants, while letting them keep their Tickets as collectibles. This is the main difference between the Refund and the Cancel Event operations. The Refund operation transfers back the NFT to the organizer, while the Cancel Event operation lets the Participant keep their NFT. The last step of this process is to burn the unsold NFTs by sending them to an address that cannot be accessed by anyone. In the end, we can clear the associated storage mappers.

Another difficulty encountered at this endpoint was that, on Collections with a lot of NFTs, the cancel event operation would fail due to the MultiversX Smart Contract API limitations. This means that one transaction cannot make more than 100 built-in function calls. In our case, if we want to trigger the `cancel_event` action on a collection that has more than 100 unsold NFTs, then the transaction will fail and return a "*max calls reached: too many built-in functions calls*" error.

The best solution for this problem would be to increase the number of transactions and decrease the number of API calls, to a maximum of 100 per transaction. For example, if I have a collection with 250 unsold NFTs that I want to cancel, I would first call the `cancel_event` endpoint that would refund the EGLD back to the non-validated participants followed by another endpoint that would clear the storage mappers and burn the unsold NFTs. This last endpoint would be called 3 times, in separate transactions, from the front end, in order to not cross the 100 API calls limit. Each call would burn a maximum of 100 NFTs.

5. Evaluation and Performance Analysis

After rigorous testing, we have concluded that the execution time of a transaction does not represent a good metric for our project because all of the requests made by our web platform are handled by the MultiversX API, and measuring the performances of the MultiversX platform is not part of this paper' objectives. Because of the introduction of Smart Contract API limits in the Polaris release, a metric that better aligns with our project's implementation is the cost of a transaction. During the preparation of our paper, the limitations were: maximum 100 API calls per transaction, maximum 250 transfers per transaction, maximum 1500 read operations per transaction.

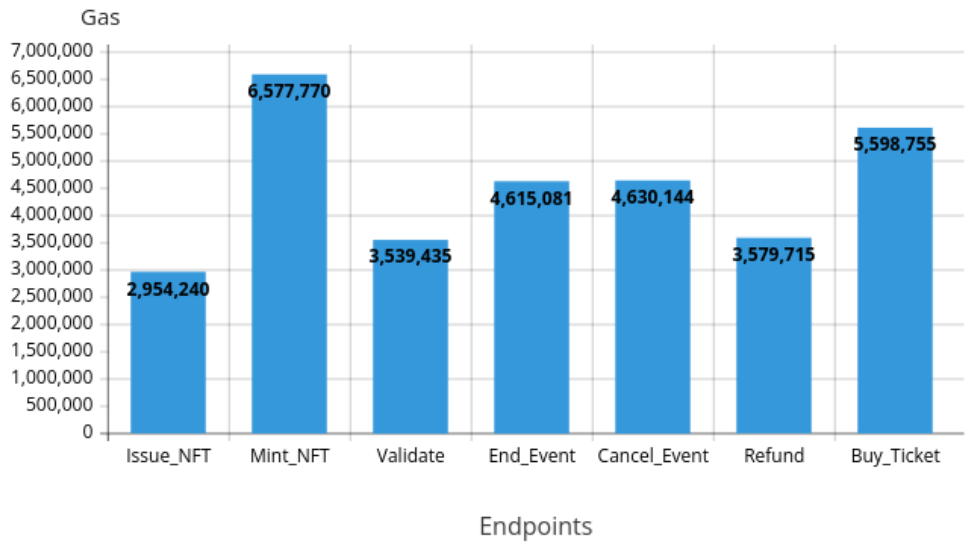

Fig. 7. Gas cost of each Operation

Figure 7 shows the cost comparison of each functionality. Because the gas cost is influenced by the length of the arguments, all of the endpoints were called for an event named "Neversea", with the ticker "NVS" on a freshly deployed contract with empty storage mappers. To better illustrate the discrepancies between the endpoint calls, only 3 tickets were minted for this collection.

As we can see in Figure 7, other than `mint_nft` that makes 3 API calls, to mint 3 NFTs, the most expensive endpoint is the `but_ticket` endpoint. This is because this endpoint calls the `sell_nft` API function that wraps 2 transactions. First, the Smart Contract sends the NFT to the caller, then the Caller sends the EGLD to the Smart Contract. This shows that the number of operations is a very important factor when determining the final gas cost. To further strengthen this claim, the next figure outlines the number of tickets minted by a single API Call, reported to the gas cost.

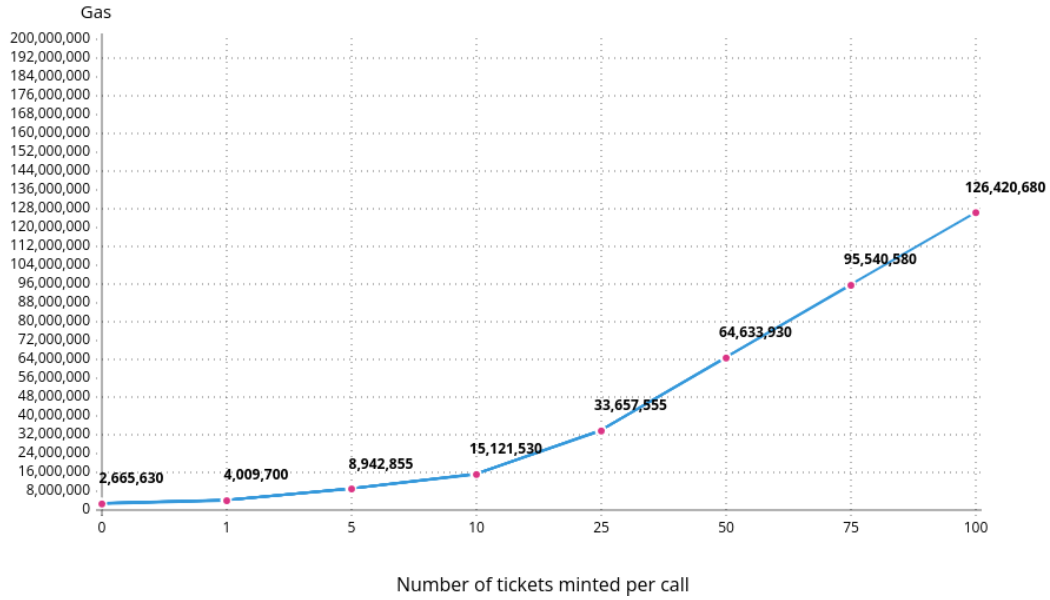

FIG. 8. Gas cost per number of tickets minted

While at first it may seem that the cost increases exponentially, and the minting operation is a very expensive one, we have to keep in mind that simply calling the endpoint and making some API read calls inside the function, without minting a single NFT, cost more than half of the gas needed to mint 1 NFT.

6. **Conclusions**

In conclusion, this paper has investigated the progress and implementation of a ticketing system based on NFTs, that uses the MultiversX blockchain for its advantages. As we navigate the digital age, our platform represents a promising avenue for enhancing security, transparency, and ownership in the ticketing industry. Organizers can effortlessly distribute their tickets more securely, while also using the facilities provided by NFTs.

The results show that we have successfully created a platform that can be easily used to organize small to medium-sized events, with a simple, user-friendly interface that bonds harmonically with the MultiversX platform. In

the case of events with over 350 tickets, some further development is required, in order to surpass the MultiversX API limitations.

To scale our solution, we can modify the application in order to surpass the MultiversX API limitations and handle large events. Moreover, we can implement an identity management system to remove the privacy challenges mentioned by authors [9].

As we navigate further into the digital age our solution can easily be adapted for other domains, aside from events, that now require tickets, like transportation, cinema, or even real estate. It can become a global railroad ticketing platform, a cinema tickets marketplace, or even a property renting platform where one property owner would issue an NFT that would represent the leasing agreement. On the same note, it could be used as an accommodation marketplace like Booking, in which people could have their room access keys in the form of NFTs.

Acknowledgement

This work was supported by a grant from the National Program for Research of the National Association of Technical Universities - GNAC ARUT 2023

REFERENCES

[1] *Ismail, Leila and Materwala, Huned.* A review of blockchain architecture and consensus protocols: Use cases, challenges, and solutions. *Symmetry*, 11(10), 2019.

[2] *YuanJiang, Yuan and Zhou, Ji Ting.* Ticketing system based on nft. In *2022 IEEE 24th International Workshop on Multimedia Signal Processing (MMSP)*, pages 01–05, 2022.

[3] MultiversX. Nft & sft tokens • multiversx docs, 2021.

[4] *Musamih, Ahmad and Dirir, Ahmed and Yaqoob, Ibrar and Salah, Khaled and Jayaraman, Raja and Puthal, Deepak.* Nfts in smart cities: Vision, applications, and challenges. *IEEE Consumer Electronics Magazine*, PP:1–14, 10 2022.

[5] *Regner, Ferdinand and Schweizer, André and Urbach, Nils.* Utilizing non-fungible tokens for an event ticketing system. In *Blockchains and the Token Economy: Theory and Practice*, pages 315–343. Springer, 2022.

[6] *Courty, Pascal.* Ticket resale, bots, and the fair price ticketing curse. *Journal of Cultural Economics*, 43(3):345–363, 2019.

[7] *Nugraha, Aji and Daniel, Debby Ratna and Utama, Anak Agung Gde Satia.* Improving multi-sport event ticketing accounting information system design through implementing rfid and blockchain technologies within covid-19 health protocols. *Heliyon*, 7(10), 2021.

[8] *Klabnik, Steve and Nichols, Carol. The Rust programming language.* No Starch Press, 2023.

[9] *Feulner, Simon and Sedlmeir, Johannes and Schlatt, Vincent and Urbach, Nils.* Exploring the use of self-sovereign identity for event ticketing systems. *Electronic Markets*, 32(3):1759–1777, 2022.