UNIVERSITÀ DEGLI STUDI DI
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea in Ingegneria Informatica

Blockchain and Decentralized Applications Project Work

Dynamic Tomatoes NFT

Prof.
Giuseppe Aceto

Studente
Simone Montella, M63001566

Indice

Introduction

Dynamic Tomatoes NFT is a blockchain-based project, a decentralized application that simulates the growth of virtual tomato plants depending on real-world weather conditions. Each tomato is represented as an ERC-721 Non-Fungible Token (NFT) that can evolve through multiple growth stages, with on-demand growth transitions dependent on temperature and humidity data obtained, on-chain, from real-world weather APIs.

The project aims to demonstrate the concept of dynamic NFTs that change over time based on external data that could be acquired, for example, from IoT sensors.

The implementation uses the Ethereum's Sepolia testnet and Chainlink Functions for the oracle services. Additionally, a React-based web platform was implemented, providing a user-friendly interface for minting, viewing, and interacting with the NFTs.

This document summarizes the project objectives, implementation details, challenges encountered, and potential future enhancements.

The project code can be found on GitHub

Related Works

Recent advancements in blockchain technology have enabled the tokenization of real-world assets, including food products, for enhanced supply chain transparency and traceability. By leveraging dynamic NFTs (dNFTs), it becomes possible to track the origin, storage conditions, and journey of food items, ensuring authenticity and increasing consumers trust. While the current project demonstrates the potential of evolving dNFTs in response to external data, similar mechanisms can be applied to real-world use cases, such as improving food safety and optimizing logistics in the agri-food industry. Studies about blockchain leveraging, specially about tokenizing and interacting with real-world objects are taking always more hold, examples are at [1] [3] [4]

Chapter 1

Project Overview

1.1 Objectives

The primary objectives of this project are:

1. To implement blockchain-based digital-twins (dNFTs) of tomato plants
2. To interact with real-world data through Chainlink ecosystem
3. To create a mechanism for NFTs to evolve based on real-world weather conditions
4. To develop a user-friendly frontend interface for interacting with the NFTs

1.2 Tools and Tech Stack

The project, which implementation will be discussed in detail later (3), was developed using various tools and technologies:

- **Blockchain Platform**: Ethereum Sepolia
- **Smart Contract Development Environment**: Hardhat
- **Weather Data Providers**: OpenWeatherMap, wttr.in
- **On-Chain Data Integration**: Chainlink Functions
- **Metadata Storage**: IPFS and Pinata
- **Front-End Development**: React with Typescript
- **Chain Interactions**: Alchemy, Ethers, Wagmi, Viem
- **IDEs**: Remix, Visual Studio Code

1.3 Code Structure

The project code, available in a GitHub repository, is structured as follows:

- /**contracts**: Solidity code for the smart contracts, it contains the main contract (DynamicTomatoes.sol), and other test contracts used to get-in-touch with Chainlink and the ERC-721 Standard
- /**scripts**: Some useful JavaScript scripts to interact with Sepolia, Chainlink, IPFS (Pinata), and weather data providers
- /**test**: JavaScript testing scripts using Hardhat's testing tools
- /**app**: Front-End codebase
  - /**src**: WebApp implementation with React and Typescript
    - /**chain**: Blockchain interaction logic
    - /**components**: Some useful React components
    - /**pages**: Application pages
    - /**theme**: Theme configuration
  - /**secrets-manager**: Simple WebService endpoint to send private OpenWeatherMap API key to the Chainlink DON

Chapter 2

Chainlink Functions and Sepolia Testnet

2.1 Chainlink Functions

Blockchain networks, by design, are isolated systems that cannot natively access external data. This limitation, known as the "oracle problem" [2] significantly restricts the potential applications of smart contracts. Chainlink emerged as a solution to this fundamental challenge by creating a secure bridge between blockchains and external data sources.

It uses a decentralized approach to delivery external data into the blockchains: rather than relying on a single point of failure, it distributes the responsibility of the data retrieve/delivery process across a network of independent node operators who each retrieve and deliver

data independently.

The Chainlink Functions service enables smart contracts to execute JavaScript code off-chain through a decentralized network, the Decentralized Oracle Network (DON): when a smart contract needs to access external data or perform complex computations, it submits a request to the Functions Router contract, containing the JavaScript code to be executed, along with any necessary parameters.

Another key feature of the Chainlink Functions is the secured approach to secrets management: API keys and other sensitive credentials are encrypted using the DON's public key before transmission.

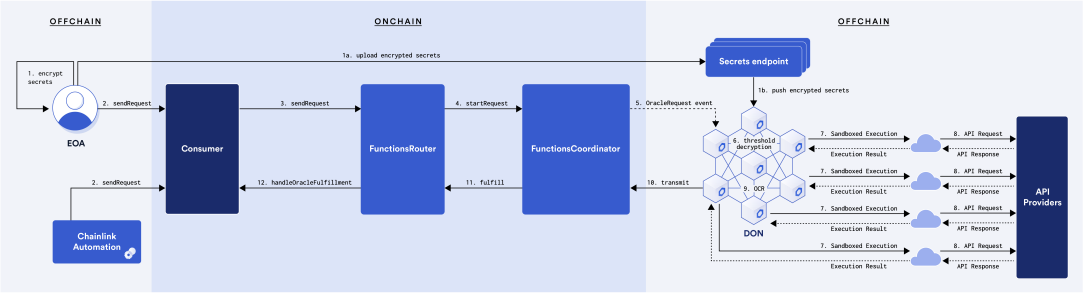

Figure 2.1: Chainlink Functions service architecture

In this project Chainlink Functions plays a central role in enabling the connection between tomato NFTs and real-world weather conditions. The process of the data retrieve/delivery is discussed in detail later in this document 3.1

2.2 Sepolia Testnet

The blockchain platform chosen for this project was Sepolia. Sepolia is one of Ethereum's main testnets, designed to faithfully simulate the mainnet environment while maintaining a lighter and more efficient infrastructure. It operates on a Proof-of-Stake (PoS) consensus mechanism, which reduces energy consumption and ensures faster finalization times for transactions: the average block time is around 12 seconds, which is very similar to Ethereum's mainnet. This makes it particularly suitable for testing smart contracts intended for the mainnet without facing the congestion or high costs associated with testnets like Goerli, which is going to be fully replaced by Sepolia.

Sepolia supports all features of the Ethereum Virtual Machine (EVM), allowing smart contracts to be easily migrated from the testnet to the mainnet and viceversa, and also allows developers to use the trusted largely-adopted Ethereum libraries, both on-chain and off-chain: also the off-chain tools which are able to interact with the Ethereum blockchain can be adopted to interact with the Sepolia chain as well.

To facilitate development, Ether on Sepolia (SepoliaETH) holds no real economic value and can be obtained for free through public faucets, like the ones offered by Google Cloud or Chainlink. This allows for large-scale testing without any real-world costs, which is particularly beneficial for projects that require numerous on-chain in-

teractions, such as the dynamic NFTs approached in this project. Furthermore Sepolia is the only Ethereum testnet allowed to interact with the Chainlink ecosystem, and this, along with the chance to get-in-touch with the Ethereum ecosystem for the first time, was one of the other reasons that made me chose this network.

Chapter 3

Implementation

3.1 Smart Contract

The core of the project is the **Dynamic Tomatoes** smart contract, which defines the on-chain backend logic of the entire system of the tomato plants digital twins. It contains all the features that follow:

- ERC-721 standard for NFT (using OpenZeppelin's implementation)
- Chainlink Functions Client for oracle integration
- Growth progression logic based on weather conditions
- Ownership management for administrative functions
- Event emission for frontend notification system and debugging

These functionalities were implemented through dedicated functions,

such as those for NFT minting and metadata upload, those for interacting with the Chainlink DON, and those handling the growth mechanism. Additionally, key variables were used, including:

- Chainlink interaction parameters (e.g., FUNCTIONS_ROUTER, DON_ID)
- Growth mechanism conditions (e.g., minimum and maximum temperature and humidity requirements)
- Mappings to maintain the state of NFTs (e.g., ownership, growth stages)

As also listed above, i found useful defining and emitting custom events to notify the frontend application, discussed in 3.2, and to monitor the processes executed on-chain by the smart contract: for example, i decided to define custom events about the growth process, like *TomatoGrowthRequest*, *TomatoGrown* and *TomatoGrowthFail*.

3.1.1 Growth Mechanism

The main functionality of the entire system is the **growth mechanism**, which can be resumed by the following steps, also figured in the sequence diagram 3.1:

1. The interested user, owner of an NFT tomato, interact with the frontend application to initiate a growth request

2. The frontend application contacts the secrets manager backend service to:

    - Generate an encrypted secret containing the OpenWeather API key
    - Upload the secret to Chainlink's DON
    - Return the slotID and version number of the uploaded secret

3. Once the slot ID and version number are known, the frontend application interacts with the smart contract by calling the *growthRequest* function, passing:

    - The tomato's ID
    - The secrets slotID and version number

4. The smart contract prepares and sends the weather data request to Chainlink, also storing it in a custom mapping (associating the request ID with the tomato's ID).

5. Chainlink processes the request executing the custom JavaScript code provided within the request (the one interacting with the OpenWeatherMap API) and sends back the code response to the smart contract

6. The received data (weather conditions) is processed and based on the (weather) predefined requirements the tomato either grows or remains unchanged. A corresponding event is emitted to notify the frontend application (and the user).

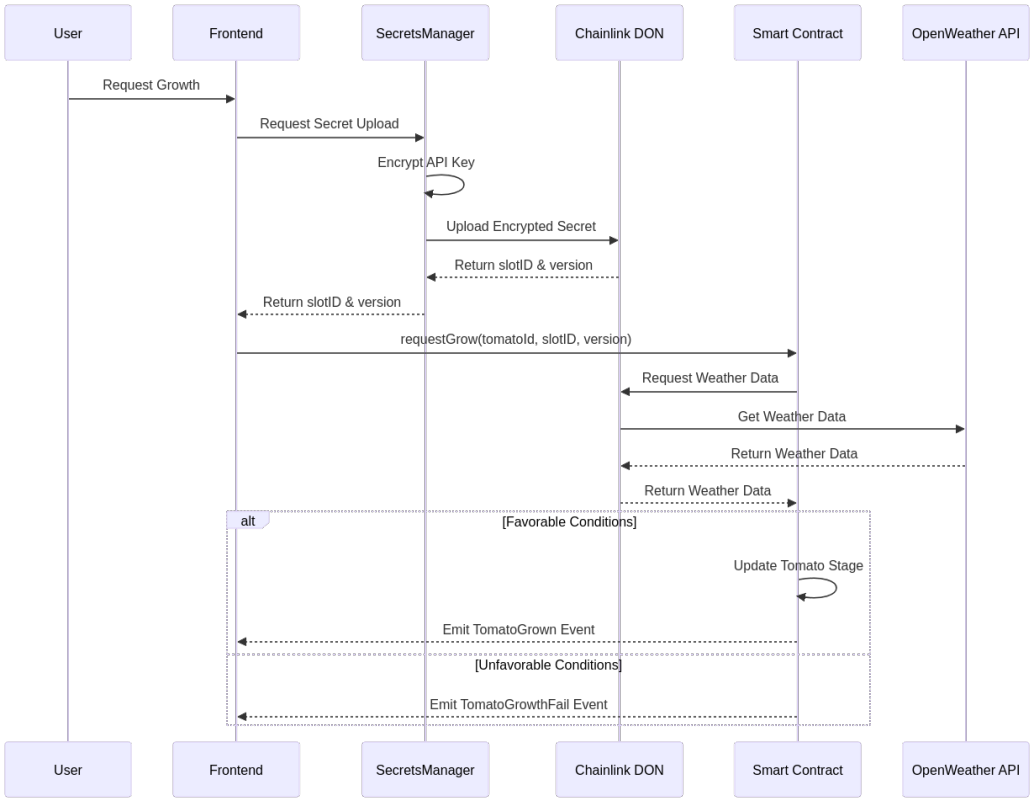

Figure 3.1: Growth mechanism sequence diagram

3.2 Front-End Application

The frontend application serves as the user interface for interacting with the Dynamic Tomatoes ecosystem. It is built with React and TypeScript, leveraging modern web3 libraries such as Wagmi, Viem, Ethers, and AlchemySDK to provide a seamless connection between users and the blockchain.

The application basically distributes its functionalities across two main pages:

- The HomePage, which displays a list of owned tomatoes.
- The Tomato Page, which contains detailed information about each tomato.

Through an intuitive interface, users can:

- Connect and disconnect their Ethereum (Sepolia) wallet.
- "Plant" (mint) new tomato NFTs.
- View a list of owned tomatoes, including their current growth stages, images, and other relevant details and actions.
- Request growth updates, refresh metadata, and access the OpenSea NFT page.

- Track each tomato's history by listing and monitoring emitted events, viewing related transactions, and accessing transaction details with direct hyperlinks to Etherscan.

- Get notified about live events on the entire system (like new mintings or stages updates)

Chapter 4

Conclusion

4.1 Challenges

This project was an opportunity to me to get-in-touch for the first time with the blockchain world, and i really enjoyed working on it since i finally had the chance to put my hands on something i had only learned about in theory or come across online. Even if i found the programming part not difficult at all, i still ran into some challenges i had to figure out, for example i had to solve:

- **Data Encoding/Decoding Errors**: Oracle responses initially failed within the DON, requiring investigation through Etherscan to identify the issue: the problem was how i encoded/decoded the weather data got from the OpenWeatherMap API. At first i tried to work with plain alphanumeric weather data strings but it was too expensive (logically and economically) and subject to

errors, then i chose to move on JSON formatted weather data, but also in this case the on-chain decoding was too expensive, and then i finally came accross the Ethers ABI data encoding tools that i could easily decode on-chain with the standard ABI decoding tools by Solidity.

- **Debugging**: The blockchain development environment doesn't offer the same tools and opportunities to code that we have in the "real" world of the largely known technologies, but on the other side there are a lot of other capabilities that i didn't find elsewhere, like the possibility to inspect, step by step, any transaction (the contract calls) i made interacting with the smart contract after having deployed it. I found it very useful, challenging and formative since i had to navigate into the blockchain to identify runtime errors in my code. I also integrated into the code some other useful tools for debugging like the Debug event to log some messages and a contract version code variable.

- **Library Version Conflicts**: Since i tried to automatize some developing process like the deployment step, some testing stuff, and mainly the secrets management part, i had to interact with the Hardhat environment and with some Chainlink toolkits but there were some conflicts between the two because of the misalignment of some libraries like the Ethers one. I avoided these problems, after a while, taking advantage of the useful powered

tools of package managing offered by Node.js (NPM), who allowed me to work with different versions of the same packages at the same moment

- **Metadata Handling**: I initially uploaded to IPFS just the plain images of the tomato plants figuring the various growth stages, but inspecting the NFTs there were - of course - no raw informations, into the public metadata, about the tomato's growth stage and this was exactly because i was naively uploading just the plain images. I easily solved moving on the JSON metadata standard format that allowed me to include also raw data with the images.

4.2 Future Directions

Since this project was made for educational purposes, it comes with relatively basic functions and logic, but it could serve as a good foundation for creating realistic applications implementing agri-food on-chain digital twins. The concept of dynamic NFTs responding to real-world data opens numerous possibilities for agricultural supply chain tracking, product authenticity verification, and consumer engagement.

A significant enhancement would be modifying the API integration to receive data directly from IoT sensors positioned in actual agricultural fields. This would transform the project from using generic

weather data to leveraging real-time, hyperlocal measurements from sensors, temperature monitors, and humidity gauges. Such integration would create a more accurate digital twin representation and could provide valuable insights for actual agricultural production, bridging the gap between virtual NFTs and real-world farming practices.

From a technical perspective, migrating to Ethereum Layer 2 solutions would significantly reduce transaction costs while improving scalability. This would make the application more accessible to users and enable more frequent interactions with the digital twins.

Bibliografia

[1] Ricardo Borges, Dos Santos, Rodrigo Palucci Pantoni, and Nunzio Marco Torrisi. Blockchain Tokens for Agri-Food Supply Chain. *Journal of Engineering Research and Sciences*, 2(2):15–23, 2023.

[2] Giulio Caldarelli. Understanding the blockchain oracle problem: A call for action. *Information*, 11(11), 2020.

[3] Noah Habtemichael, Hendro Wicaksono, and Omid Fatahi Valilai. Nft based digital twins for tracing value added creation in manufacturing supply chains. *Procedia Computer Science*, 232:2841–2846, 2024. 5th International Conference on Industry 4.0 and Smart Manufacturing (ISM 2023).

[4] Karan Singh Thakur, Rohit Ahuja, and Raman Singh. Iot-gchain: Internet of things-assisted secure and tractable grain supply chain framework leveraging blockchain. *Electronics*, 13(18), 2024.