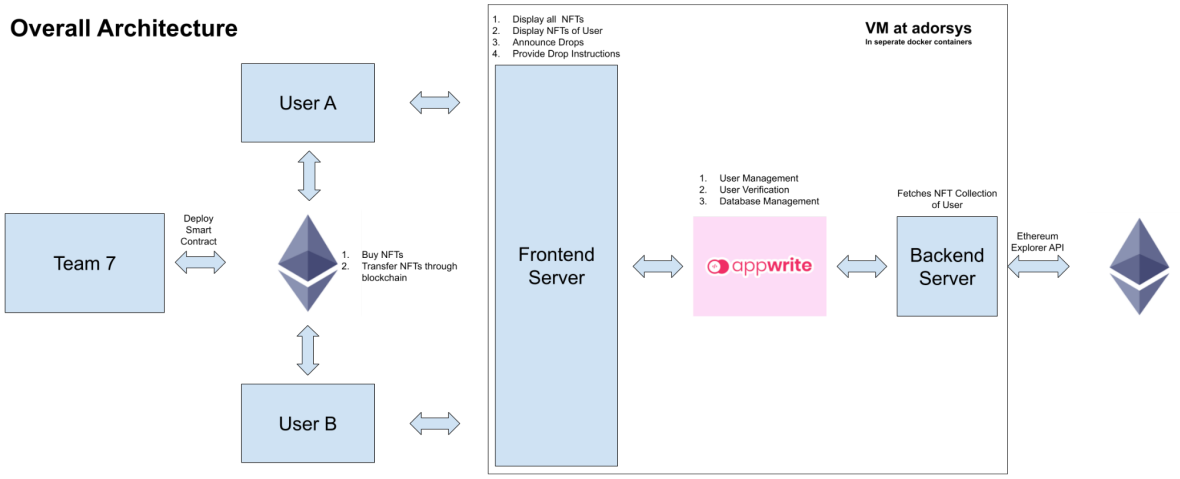
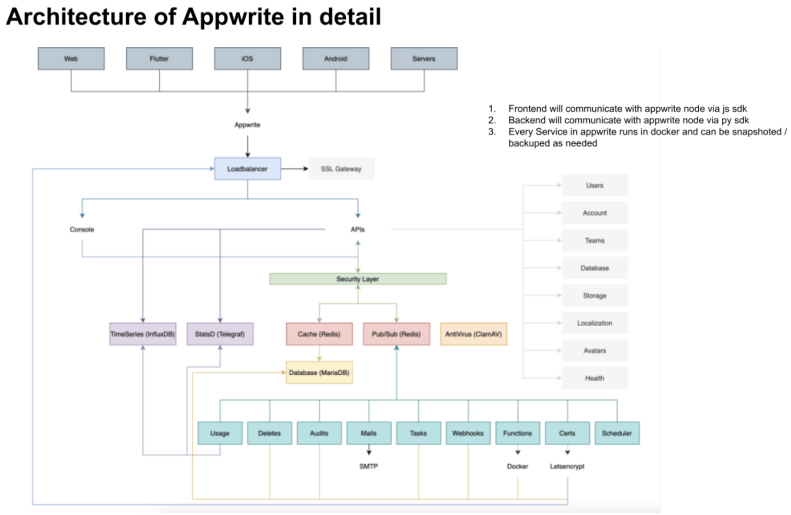
Software Architecture

Runtime Components





https://docs.google.com/drawings/d/1bIWAVe0IjYw1AKSm3l9tGMDktJxQ8krB4e8Bm88Yrh4/edit

Code Components

Yet it’s too early for us to define all of the code components because the project contains a large number of components with different code components in each category.

Technology Stack Summary

Our web application features:

Frontend

The client side facing part of the app will be programmed using the React framework. We decided to use JavaScript as a language. ESLinter is deployed to run as a commit hook with Git to check for simple code smells and code formatting that is uniform for Frontend.

Material UI is used as the library for the graphical representation and interaction components of the side.

Backend

AppWrite is used as a Backend-as-a-Service solution which runs custom services inside Docker containers. Python was selected as the programming language.

The backend development writes software in containers with Python. The AppWrite SDK has a Python API. A Code Formatter *Black* is used together with Linters *Flake8* and *Bandit*.

Python's *FastAPI* is the option to use when AppWrite would not provide a certain functionality that is needed.

Blockchain

The project is based on Ethereum which uses the crypto currency ETH (ether).

Solidity is used as a feature-rich programming language which supports OOP principles.

The team uses a fresh MetaMask wallet (payment address) to transact test currency to buy NFTs in the test net.

The Kovan testnet will be used.

Database

Maria DB (a relational database), because of good support by AppWrite

Runtime

Besides Github, we preferred to have a simple external server computer to host our application. It could use a virtual machine that is not specified yet.

Technology Stack Explanation

Frontend

The frontend controls the user interaction and displays UI for better usability.

React JS is a popular website framework that allows for well encapsulated (well-structured) generation of HTML controlled by JavaScript code. Via a poll, JavaScript was prefered by most Frontend developers as the language in combination with ReactJS.

Material UI has been proposed without any objections. It's rich palette of pre-made but well customizable components allows us to concentrate on the relevant part. It should provide a well responsive user experience.

Backend

Our service features are programmed with **Python** because it is easy to use and has a rich ecosystem.

AppWrite is an open-source user management and authentication system and a backend abstraction with abstractions for databases, localization, mail notification, security features, even antivirus software and an own console. Programming a user management system is a cumbersome and uninspired task. To keep the focus on our features, instead of reinventing the wheel, we use a well-received open-source project that allows us to run our own service components in a containerized environment.

FastAPI is a modern common Python web framework, which is made for speed.

Blockchain

Security critical procedural steps and rules for handling (NFT) transactions are programmed as Smart Contracts that are run by nodes in a blockchain network.

We are using **Solidity** as our programming language for the smart contracts since it is one of the most active and maintained ones available. Solidity supports a lot of features like contract inheritance and the creation of libraries.
We decided to use **Kovan** as a testnet since it is one of the most popular ones (which means there's a lot of community support in the case of questions or issues), it is not vulnerable to spam attacks since test-Ether need to be requested, and the faucet doesn't involve any social media interaction. We are going to use **MetaMask** as a wallet to store our test-currencies, since it has an appealing user interface and is straightforward to use.

Database

A database is needed to persistently store account information and, importantly, the data that is referenced by the block chain.

Maria DB is a relational database that can be accessed with SQL. Our main reason for Maria DB is first-class support by AppWrite and it's similar in nature to the formerly proposed Postgre DB. Note, the selection is not based on the analysis of our NFTs data requirements. But since AppWrite provides database abstraction, an exchange of the NFT database is less effort when needed later.