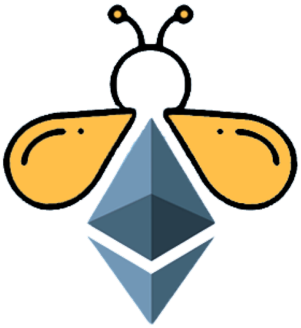
Telecommunications
S8 PROJECT



Eirbmon

Students :
Antoine Perfettini | aperfetti001*
Guillaume Sauvadet | gsauvadet*
Martin Héliot | maheliot*
Karima Squalli Houssaini | ksquallihou*
Lucas Fauchereau | lufauchereau*
Nicolas Blanc | nblanc003*
Alexandre Baudry | abaudry005*
Nadia Hilmi | nhilmi*
*@enseirb-matmeca.fr

Supervisor :
Joachim Bruneau-Queyreix

BORDEAUX INP

Contents

Abstract

Overall, this project consists of the creation and development of a Decentralized Application (DApp) combined with a video game. In the game, the player can capture and train Eirbees which are NFTs, whereas the web site will allow players to see, sell and buy their NFTs on a marketplace using Metamask and an Ethereum blockchain. To do so, the *Agile/Scrum method* was chosen to carry out this project.

1 Introduction

In the search for transparency, security and decentralization, the blockchain technology is now a centerpiece in the field of innovation and design of modern projects. As the 21st century has witnessed the rise of crypto-currencies and the craze of many investors, NFTs have followed suit.

NFTs or "Non Fungible Tokens" are a direct result of the blockchain and are wholly based on this technique. Unlike crypto-currencies, this "Token" is unique and cannot be reproduced. It is then possible to own virtual objects in multiple sectors. From art to video games or even sports, those unique digital objects can retain an important monetary value.

A real economic market that follows the same laws as those of the traditional market then becomes available. Indeed, the market of supply and demand makes sense in this universe because it is governed by many factors of scarcity, NFT are limited in quantity. The video game industry tries to take advantage of these innovations by proposing contents where the player can be the owner of the elements that surround him.

The Eirbmon project, which is part of the blockchain world, is no exception to the rule. In a playful way, it introduces to the user the concept of NFT and the monetary value associated with it.

2 Project architecture

Since the project was based on a "Pokémon" type of game where the exchange of creatures was done through the use of a blockchain and a marketplace, the team was divided into four pairs. For all groups a part of the project was assigned : the blockchain and smart contract creation, the server and the website database management, the website on the client side supervision and finally the creation of the video game.

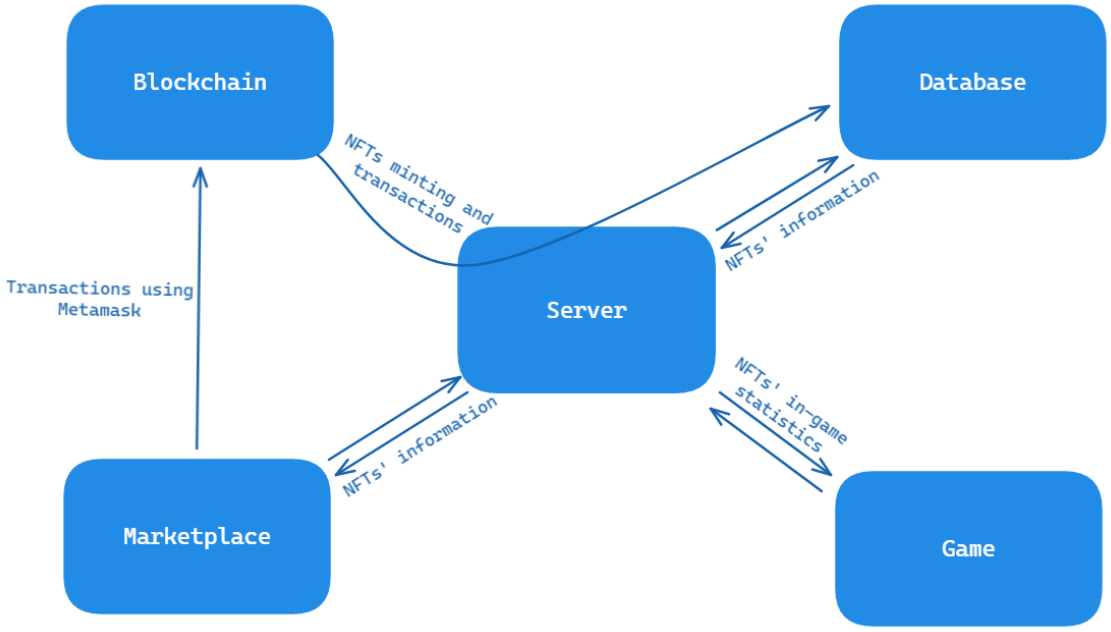


Figure 1: General architecture of the project

As shown in Figure 1, the server is the centerpiece of this project as all the other parts are communicating with it. Moreover, it can be seen that almost every communication passes through the server, with one exception, being the interaction between the marketplace of the website and the blockchain. Indeed, this direct link is handled entirely by Metamask, and this specificity will be explained further on this report.

3 Project management

The final expectations of this project were flexible. The concept was clear, but the specifications were not fully defined, allowing us to express ourselves on certain aspects of the project, notably the game design. As objectives could change with the progress and advancements, the Agile/Scrum method was used to properly carry out the project.

The Agile/Scrum method has a lot of features that fit with the lineaments of Eirbmon. First of all, this type of management is designed for teams with around 9 members. Then, the short term of the final render required the short sprint of the scrum method. In fact, one week-short sprint were planned throughout the project. These sprints permitted a quick reorganization of the team if an issue or an unexpected situation occurred. Throughout the weeks, two or three meetings were scheduled to ensure the pace of each sprint, solve possible difficulties and overall project progress.

To keep track of the monitoring and the traceability of all tasks during each sprint, the online tool *Trello* was used. Each task can be validated (or not) in order to follow the activities and the results over all the duration of this project and for all parts (Figure 2).

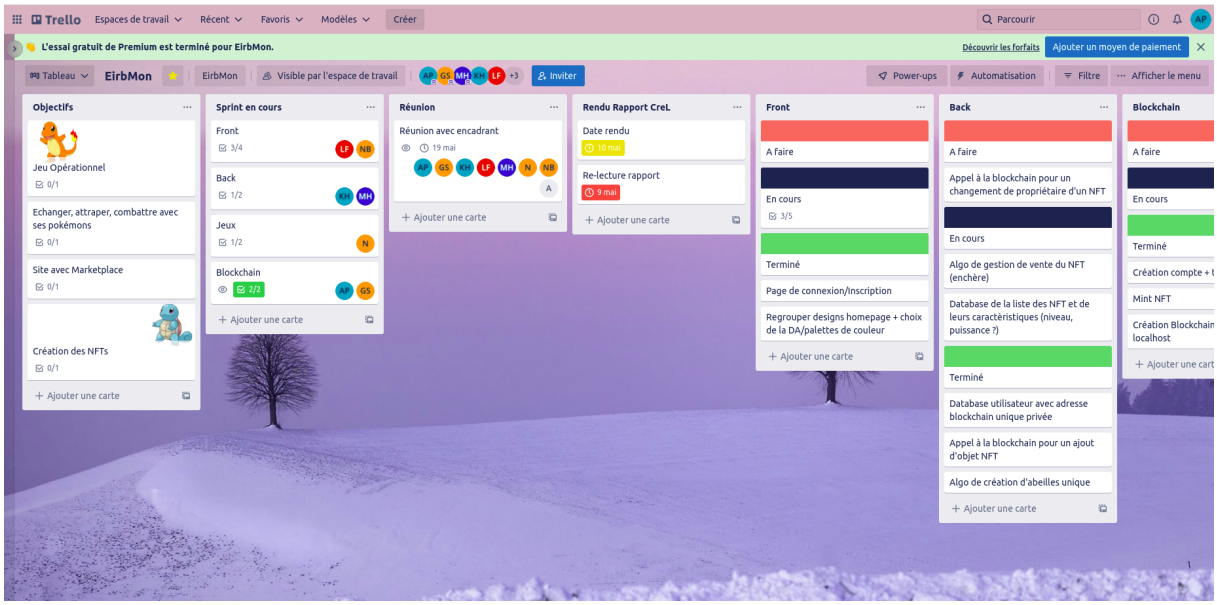


Figure 2: User interface of Trello

4 Game

4.1 NFT generation

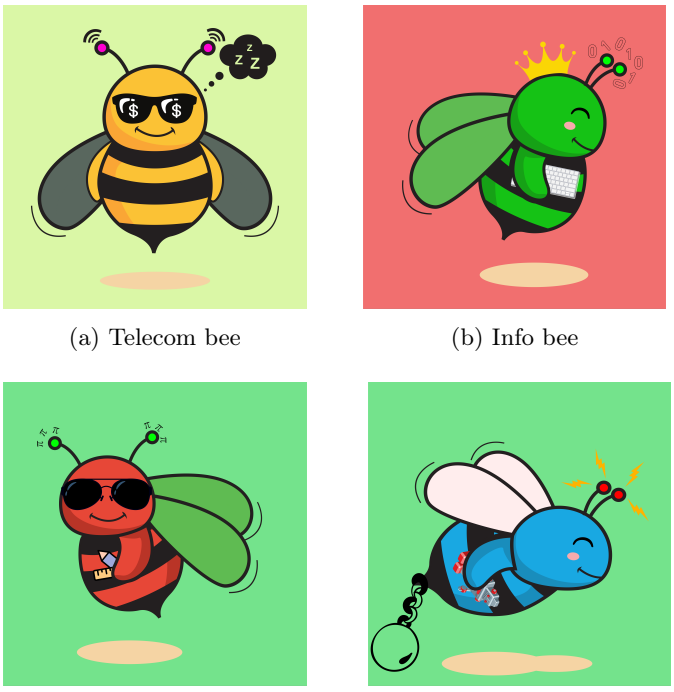
According to the definition of an NFT, it has to be identified as unique. Many unique images had to be generated. A program using *NodeJs* was conceived and used to generate 4000 different drawings, namely 1000 bees for each major in Enseirb-Matmeca.

The *SVG* picture format allows to dive into the code of an image and modify it with many programming languages. Based on only four drawings, 4000 were created, by modifying colors and adding accessories.

Each of the 4000 SVG files (about 120 kilobytes for an image) was hosted on a server based in the US and the URLs were stored in the database to make sure the game and the website can access these images.

To create a difference between each Eirbee in their in-game combat statistics, an index called potential was created. This property of the NFT is created randomly along with the NFT, following a Gaussian distribution, in the same way as the IQ. In addition to the cosmetic features, this combat statistic adds real market value to each Eirbee.

Figure 3 shows four examples of bees that were generated with this program.



(a) Telecom bee (b) Info bee

(c) Matmeca bee (d) Elec bee

Figure 3: Example of generated Eirbees

4.2 Game concept

The game allows players to use their NFTS which are their Eirbees. Indeed, the goal of the game is to make their Eirbees progress so that they increase in value, and therefore the players can sell them at a better price or exchange them with more powerful Eirbees. (Figure 4).

(a) Fight scene (b) Open world

Figure 4: Video Game

In order to be able to play, the player must own at least one Eirbee and therefore buy some in the marketplace. The player will have to move around the map and try to find other Eirbees in the long grass in order to fight them. Then, thanks to the fights, the player's Eirbees will gain skills: they will increase their level and unlock new attacks.

The player may also rarely encounter special Eirbees in the long grass that he will be able to capture. If the player succeeds in capturing the Eirbee, then it will be added to his inventory.

During a fight, the player can switch his fighter and choose another one of his Eirbees. If the player encounters an Eirbee and doesn't want to fight, he has the possibility to run from the fight.

Technically, a "master account" owns all the Eirbees at the time of minting the NFTs. As soon as a user buys an NFT that is supposed no-one owns, a transaction is performed between the master account and the user. The transaction is similar when a user captures an Eirbee, except that the purchase price is set to 0 ETH.

The game was created using *Unity* software development tool and *C#* language. In order to integrate the game on the website, the WebGL Build Support module was used. The game is built thanks to this module then using a reactjs file the game is displayed on the website.

5 Developing smart contracts for NFTs

A blockchain is a technology that makes transmits possible and storage of information with transparency and security and does this without any central part. It is based on distributed algorithms, thus each user owns a part of the blockchain. This technology enables the storage of tokens such as crypto-currencies or NFTs. NFTs (Non Fungible Tokens) are cryptographic tokens, they differ from the crypto-currencies such as bitcoins by their uniqueness.

Everything that is located on a blockchain needs smart contracts to be interacted with. Smart contracts are code written in *Solidity* that are deployed to a blockchain using *Truffle* or *Hardhate* for example and gives permission to interact with elements of the blockchain. The first one that was required was a contract in order to *mint* our NFTs. Minting an NFT corresponds to the process of writing its metadata to the blockchain and giving it an owner. Each NFTs' metadata is stored on *IPFS (InterPlanetary File System)*, which gives access to the data everywhere. At first, a private and local blockchain was used for development and testing using *Ganache*. However, for all users to be able to interact with the same blockchain, it was decided to move everything to a public blockchain where all our NFTs and smart contracts are accessible. The testing blockchain *Ropsten* was then chosen to work with.

Once all 4000 NFTs were minted, a marketplace was needed to be able to sell and buy them. Thus, a smart contract describing this marketplace was written to let users sell their NFTs for any price they want. Buyers can check every available item at one point in time in the marketplace and decide which NFT to buy.

Then, to interact with our newly generated smart contracts, the *Web3* [3] library in JavaScript is used. It gives the possibility to connect to any given blockchain and execute transaction on it.

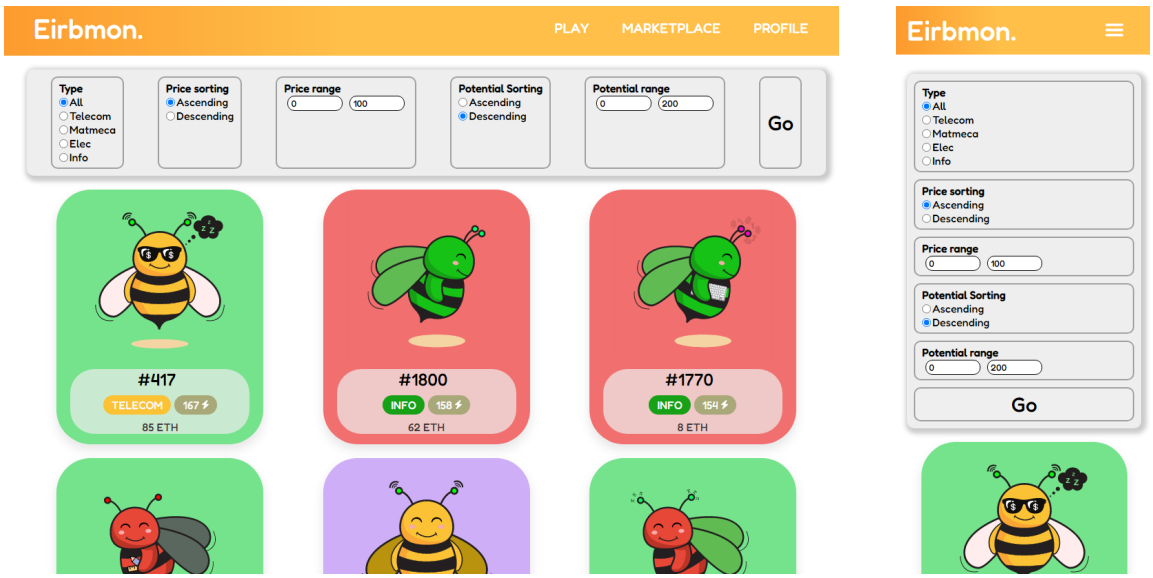
Deploying smart contracts or creating transactions cost money. Indeed, every transaction has a gas fee that needs to be paid in order for it to be approved. Thus, as *Ropsten* is a blockchain used for testing, crypto-currencies can be acquired by requesting *faucets* to send a limited amount of ethers to a chosen wallet. This principle is great for development purposes, but as a lot of NFTs were required for the project, the gas fees added up to a consequential amount whereas the faucet could only give a certain amount of ethers per day.

6 Web application

To access the game, the user must first purchase an Eirbee, following the trend of crypto games such as *CryptoKitties*. Thus, the user requires access to a platform to exchange NFTs with cryptocurrency. A website was therefore developed to meet this need and two teams worked on it, one on the server side and the other on the client side.

6.1 Client-side application

It was decided what features the website would include at the beginning of the project. A marketplace page where the user could sort the Eirbees according to their price, type and potential, is crucial. The user could also access his profile in order to view his collection and resell some of his Eirbees. It was important to make a mobile version of our website, so another version was also developed for this platform, using *CSS Media Queries* (Figure 5).



(a) Computer version (b) Mobile version

Figure 5: Computer and mobile versions of the website

In order to develop the application on the client side, the *Vue.js* framework was used because it allows to have an essential modularity for the creation of the marketplace. To access the data on the server, a single HTTP request of type GET was used (this topic will be explained further in section 6.2.1), which returns an array of Eirbees objects. The display of each of these objects is then realized in a dynamic way. Also, the *Metamask* browser extension was used in order to have access to its cryptocurrency wallet. This way, it was easier to get the user's address and perform the transactions. Indeed, *Metamask* acts as an intermediate between our client application and the blockchain. Each transaction a user could want to do will be signed by *Metamask* itself, with the approval of the user, and sent to the blockchain.

To split the website into different pages, the *Vue-router* library for *Vue.js* was used. Its purpose is to divide the website into different URLs, with each URL corresponding to a different page with specific components. The specificity of this library is that some parts (some components) of the page remain "mounted" on the page when the URL changes, so that it does not have to be reloaded. In the case of the Eirbmon website, the header and the footer stay the same for each page, as they do not have to change: for performance purposes, they do not reload. When a page needs data from the server, the component that needs this data will make an *Axios* request at mount time and insert the fetched data as props. Another advantage of using this library is that the pages browsing is done dynamically, to avoid loading time while browsing between pages, thanks to caching techniques performed by the library.

Nevertheless, a problem had to be dealt with, namely that the video game described in the previous section is hosted on a web page using *React.js* (another library for building dynamic web pages, quite similar to *Vue.js*), as it seemed very difficult to host it with *Vue.js*. The solution chosen was to use two different servers: one server with *React.js* on a specific port, and another server with *Vue.js* on a different port. The chosen solution was the simplest because it was easy to connect the two sites. One of the disadvantages of this technique is that there is a loading time on the change of pages between the *Vue.js* website and the *React.js* website, but this loading time is negligible. The other drawback is that if this project were to be deployed on the Internet (and not only hosted locally as was done for the

whole project), it would cost a considerable among of money since two different servers would have to be rented.

6.2 Server-side application

6.2.1 Web Server

To access web pages of any web application, a *Web Server* that handles the Hyper Text Transfer Protocol (HTTP) requests sent by the HTTP client is required.

The two request methods that were used in our application are the GET method which was used to retrieve information from the given service (the Marketplace or the NFT characteristics for instance) with no other effect on the data, and the POST method, which was used to send the data to the server (such as the user's wallet).

The Web Server that was used to retrieve information from the database is *NodeJs*. It is an open-source and cross-platform runtime environment for executing JavaScript code outside a browser.
NodeJs was used because of its numerous benefits, such as the easy scalability, the fast suite (all asynchronous operations, such as reading, writing, can be done quickly, in the database and network connection), and finally because of its straightforwardness (NodeJs is easy to learn and to use).[1]

To make the client application communicate with the server through the HTTP protocol and to make HTTP requests from NodeJs, the Javascript Library *Axios* was used.

6.2.2 Database

As a database is necessary for any interactive website, designing it is the most crucial part of development. For this end, MongoDb was used, it is a general-purpose database that has provided many benefits to our application thanks to its scaling capabilities and flexible schema for storing data. Indeed, each record in a MongoDb database is a document described in BSON, a binary representation of data, instead of storing it in tables of rows or columns, which is done in SQL databases. Information is then retrieved in a JSON format by the server. Hence, MongoDB falls into the document database category, which is part of the more prominent NoSQL databases family (i.e non-relational database or databases that stores data in a format other than relational tables). [2]

More specifically, the *Mongoose library* was used for MongoDB. It is a NodeJs-based Object Data Modeling (ODM) library for MongoDB that allows a specific schema at the application layer, so that there is a consensus on what the data model should look like, and that every document in the collections created contains the exact same fields.

In this project, two collections were required: one for the users and one for all the NFTs generated. The first one only contains two fields: the user wallet address and the list of NFTs that he owns, whereas the second one contains many metadata describing the bees. Range of metadata are wide it can be their colors and accessory list, but also their potential, their image link or the wallet address of the owner.

Eventually, a third collection containing the real minted NFT ID had to be created. In fact NFT minting order was empirical, leading to an overlap between real NFT IDs and their document in the database. To fix this overlap, a matching algorithm was used.

Once the two Mongoose models were defined, queries could be run for fetching, updating, and deleting data onto our MongoDB collections that align with the Mongoose model through HTTP requests.

Instead of using a local database, an online one was used so that all users could share a common database, no matter what machine they are running the web application on.
For this purpose, data were stored in *MongoDB Atlas*, a multi-cloud application data platform and one of MongoDB cloud services, which offers 250Mo of free storage.

7 Conclusion

Consequently the Eirbmon project is a DApp (Decentralized Application) that corresponds well with the new technologies. It combines programming fields such as web development, blockchain implementation as well as game development.

The organization project was based on self-management and autonomy and our schedule was nearly always respected, the team did not fall behind throughout the months.

The lack of time to enhance the user experience is the main regret about this project.

List of Figures

References

[1] *Introduction to Node.js*, https://www.geeksforgeeks.org/introduction-to-node-js

[2] *Why Use MongoDB and When to Use It?*, https://www.mongodb.com/why-use-mongodb

[3] *web3.js - Ethereum JavaScript API*, https://web3js.readthedocs.io/en/v1.7.3/

[4] *Vue.js documentation*, https://vuejs.org/guide/introduction.html

[5] *Write deploy an nft - Ethereum Blockchain*, https://ethereum.org/fr/developers/tutorials/how-to-write-and-deploy-an-nft