Development of a Distributed Social Network using NEM2 Blockchain

Author: Aleix Morgadas

Director: Cristina Gómez

5th July, 2019

Barcelona School of Informatics

Universitat Politècnica de Catalunya

Thesis presented for the Bachelor Degree of Computer Science

Speciality in Software Engineering

Abstract

Català

L'objectiu d'aquest projecte és desenvolupar una xarxa social, que anomenarem LogBook.Social, en una plataforma Blockchain utilitzant metodologies àgils i arquitectura evolutiva.

LogBook.Social coloca als creadors de contingut en el centre de la plataforma, empoderant-los amb la monetització del contingut.

Durant el projecte, s'analitzarà quins son els inconvenients de la tecnologia per satisfer els requeriments de negoci i quines alternatives podem trobar.

Castellano

El objetivo del proyecto es desarrollar una red social, de nombre LogBook.Social, en una plataforma Blockchain usando metodologías ágiles y arquitectura evolutiva.

LogBook.Social coloca a los creadores de contenido en el centro de la plataforma, empoderandolos con la monetización del contenido.

Durante el proyecto, se analizará cuales son los inconvenientes de la tecnología para satisfacer los requerimientos de negocio y que alternativas podemos encontrar.

English

The objective of the project is to develop a social network, named LogBook.Social, with a Blockchain platform using agile methodologies and evolutionary architecture.

LogBook.Social puts the content creators in the center of the platform, empowering them with the content monetization.

During the project, the technology disadvantages to satisfy the business requirement will be analyzed and an alternatives will be shared.

Keywords: *social network platform, blockchain, evolutionary architecture, agile methodologies, platform, webapp, distributed system, command sourcing.*

Index Table

1. Introduction

The project is about developing a Social Media platform using the distributed Blockchain technology. The project aims to emulate the scenario of a startup in an early stage that builds a Minimum Viable Product (MVP) as first iteration of a social media platform.

The distributed social media platform, called LogBook.Social, mission is to help Content Creators monetize their content from the people that consume the content. LogBook.Social works as a social media aggregation platform, where users can publish posts, commend them and share some digital assets.

The way we as a platform enables content monetization is through Blockchain technology. Using the native digital asset exchange that Blockchain technology has, we add this unique feature to the social media platform as a method to exchange value between content creators and viewers. Also, it allows us to explore a new business model that does not use advertising as principal driver to generate revenue.

Because of the scenario of being a startup without a lot of resources to invest on the product up-front, we choose two methodologies that helps us minimize the risk and the investment of developing the platform.

From the MVP definition perspective, we use the Lean Startup method. It helps us focus on the minimum required features to define the first iteration of the platform. Even though we aim to define a small set of features, the project should have the vision of becoming the backbone of a large platform.

On the other hand, from a development perspective, we use Kanban as an agile methodology. It allows us to respond to change faster due to the different technology risks and uncertainty. Also, it helps us incrementally create the platform and verify that we are on the good path delivering value to the end user.

1.1. Context

The project is composed of two known platforms, current social media platforms and Blockchain technology. We will analyse them separately to understand where we start from.

1.1.1. Social media platforms

Social media platforms have increased in popularity during the last decade thanks to Facebook, Twitter, Reddit, Snapchat, Youtube... to name some. Part of the success of the social media is due to the common features they all have:

1. Social media are interactive Web 2.0 Internet-based applications.
2. User-generated content, such as text posts or comments, digital photos or videos, and data generated through all online interactions, is the lifeblood of social media.
3. Users can create service-specific profiles, for the website or app, that are designed and maintained by the social media organization.
4. Social media facilitates the development of online social networks by connecting a user's profile with those of other individuals or groups.

This features allow the social media platforms have become a driver for users to share their content through a network of people, instead of them creating this content. It helped the platforms control the information and the user information, a topic that has become controversial in our society during the last two years because of the Facebook–Cambridge Analytica data scandal[1].

After the scandal, some movements had appeared to try to help people be more conscious about the data they shared, and the problematic of this platforms being biased on how they use it for different things. Even with the data scandals, the social media business keeps growing.

According to statista.com[2], see figure 1, that the worldwide revenue from enterprise social networks during the 2018-2019 has been 3 498.7 million U.S. dollars, around a 20% more revenue compared to 2017-2018.

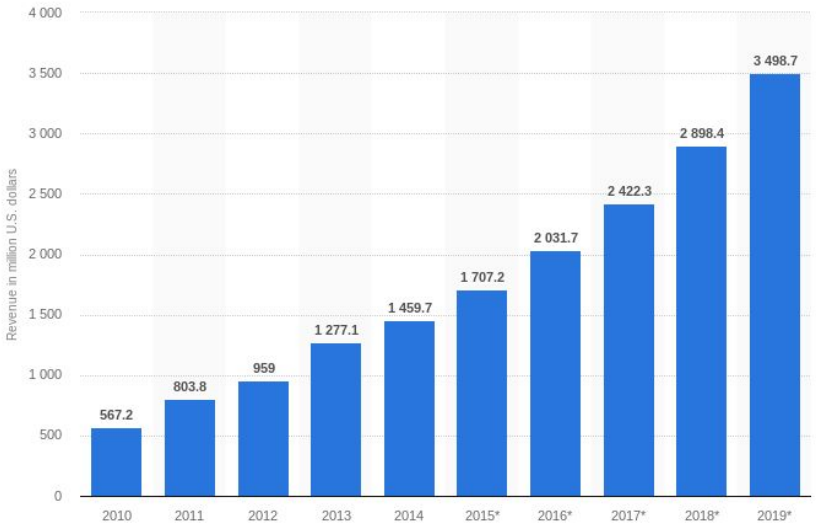


Figure 1 - statista.com
Revenue from enterprise social networks worldwide from 2010 to 2019 (in million U.S. dollars)

The social media platforms are a suitable business that do not seem to stop growing in the next years, based on previous years data.

1.1.2. Blockchain technology

We have another market that has grown fast in popularity during the last two years between companies, governments and users: the Blockchain[3] Technology. Figure 2 shows an increasing search over the Blockchain keyword on Google.

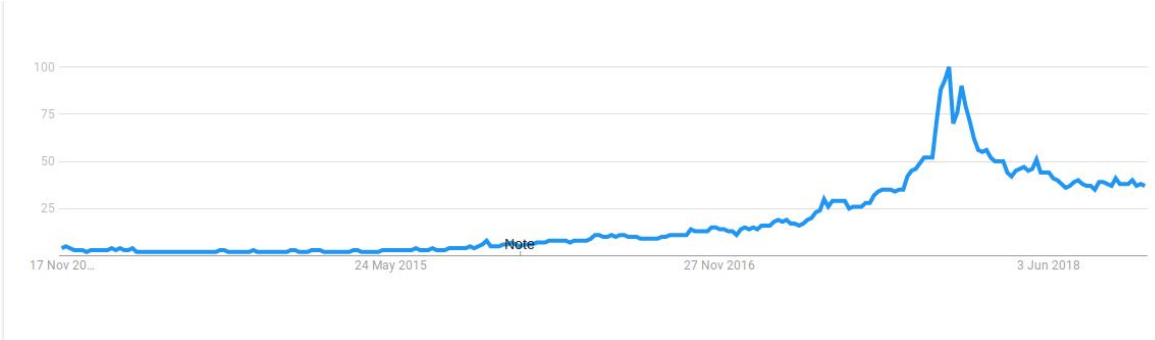


Figure 2 - Blockchain interest over the last 5 years at Google Trends

Blockchain enables us a new way to exchange value between people over the Internet, replacing the intermediaries by a trusted-distributed system that no central authority controls, a decentralized system. Using the Blockchain technology, the platform helps the readers and the content creators exchange digital assets by quality content.

To understand Blockchain technology, we have to see when was it invented and which was the purpose.

The first Blockchain application was made by Satoshi Nakamoto (an unknown person or group of people) at 2009, called Bitcoin[4]. Bitcoin is a decentralized digital currency that enables the transfer of value from user-to-user without a central authority or intermediate.

Bitcoin, as a digital currency, was needed to solve the double-spending issue, among others. In order to do so, Satoshi Nakamoto created a public decentralized ledger that records all bitcoin transactions and stores them into a sequence of linked blocks.

Meanwhile Bitcoin was using the Blockchain Technology to store the transactions, other people found the Blockchain Technology useful for other more broad use-case scenarios.

It is the case of NEM[5] Blockchain. Launched in March 2015, NEM Blockchain aimed to create a simpler Blockchain platform for developers, focusing on the most used use cases and decreasing the complexity on building Blockchain products on top of it though an API.

1.2. Stakeholders

The project has three crucial platform user stakeholders. Each role makes the platform usable by other roles, and we need the three of them to make the platform a success.

On the project team we have the only developer, which is the Bachelor's Thesis Author, Aleix Morgadas, and the Project Director, Cristina Gomez.

- **Content Creators/Publishers**

A publisher is a content creator. They benefit from the platform due to the explicit focus on the content ownership and generate profit for the content creators. They help the platform be content rich and engage with the viewers.

- **Space owners**

The space[1] owners are the ones more interested that the publishers get into their spaces. When the space owner gets enough traffic, *publishers* might start paying to promote their post over the others.

- **Viewers**

A viewer is a user that consumes the content. They might tip the *publisher* or let some comments into the post. A viewer is seen as a potential content creator in the platform when it starts adding comments.

- **Project director**

The project director of the Bachelor's Thesis, Cristina Gómez, will take care of the project follow up, the correctness of the documentation and reviewing the milestone's completes.

- **Development team**

The development team takes care of the project analysis, requirement definition, project estimation, project development and performing the required testing. The development team member is just the author of the current memory, Aleix Morgadas.

1.3. State of Art

1.3.1. Market research

In order to have a more broad vision of the platform, we do a previous research of the existing similar applications in the market. After the research, we can obtain the current state, the weakness and strengths of each platform, and use that information to improve the current project.

[1] A *space* is a place where publishers add their posts related to a specific topic, creating a community in that *space*.

There is one main social media application on top of Blockchain platform that can benefit the current project: steemit.com.

Steemit is built on top of Steem Blockchain. Steem Blockchain is a public Blockchain that takes into account which are the social media platforms needs, like high throughput, network fees to help publishers monetize the content in a specific way.

So, the state of the art has no previous project to reference, there are some proof of concepts published but without a white paper or similar to study as starting reference material. Thus, we analyze the current Blockchain technologies available on the market as their references as starting reference material.

At section 8, Social Network Research, we have analysed in more detail the existing social networks that do not use Blockchain, like Reddit, and analysed Steemit features compared to LogBook.Social.

1.3.2. Current Blockchain technologies

A broad types of Blockchains have been pushed into the market. We can group them in public vs private vs permissioned blockchain and smart contract vs predefined features. Each of those comes with advantages and disadvantages.

A Smart Contract[6] Blockchain provides a programming language to write business logic on chain. It implies more freedom for the developer to write the software, but also implies more bugs on the deeper layer, becoming more vulnerable to on-chain attacks, which might end up causing money being stolen[7] or permanently locked[8].

On the other side, we have the predefined features Blockchains. Those Blockchains provide a predefined and well tested features that can be called via an API. They offer a simpler way to integrate with a Blockchain, at a cost that they are not as customizable as a Smart Contract Blockchain.

On terms of public versus permissionated or private Blockchains. The main difference is that public Blockchains have not have a central authority that manages the system. Instead, they are executed by a huge network of nodes that makes a difficulty for a single party to gain control of the network. Usually, on most public chains, the information stored on the chain is public for all parties.

A permissionated or private Blockchain is often executed by one or multiple parties, allowing them to establish the network rules up-front. This helps companies to agree in a "way" to exchange information between them. They are also used on testing environments before moving the system to a public chain. So, private chains allow the product have more freedom on different aspects of the development and product design.

Ethereum, a public Blockchain with Smart Contract support

Ethereum[9] is the most popular smart contract Blockchain platform released in 2015. It has been adopted by numerous projects during the last two years. Due to its increase of usability, the number of security issues are high since the technical complexity to write robust software has been a problem because of how the Smart Contracts are developed.

Mastering Ethereum[10] book explains how to run Ethereum node, do the basic transactions, understand the basic cryptographic behind and build Smart Contracts and Distributed Applications.

Hyperledger Fabric, a permissioned Blockchain with "Smart Contract" support

Hyperledger Fabric[11] is permissioned Blockchain originally contributed by IBM. It supports a *Smart Contract* like feature, similar to Ethereum, where the developers can add their logic to manage digital assets.

Hands-On Blockchain with Hyperledger[12] book explains the basics to setup Hyperledger networks and write chaincode (specific Smart Contract implementation for Hyperledger).

NEM, a public and private Blockchain with predefined features

NEM, being a predefined feature like Blockchain, offers both options: a private and a public version. NEM focuses on digital assets management and security features to help users protect themselves from common attacks.

NEM offers an API that simplifies the development process. The information available of NEM is just found on the main web page, nemtech.github.io, and there are no reference books.

1.3.3. Putting down available resources and social media together

The resources available about how to use Blockchain are limited, that is why the project needs to relay reference books of similar Blockchains to have a broad technical decisions to take into account while developing the project.

Some of those books are:

- Mastering Bitcoin 2nd Edition - Programming the Open Blockchain
- Mastering Ethereum, by Andreas M. Antonopoulos, Gavin Wood

Some of those technologies group the improvements as Improvement Proposals.

- https://github.com/bitcoin/bips
- https://github.com/ethereum/EIPs
- https://github.com/nemtech/NIP/

An Improvement Proposal is a design document providing information to the community, or describing a new feature for the Blockchain or its processes or environment. The Improvement Proposal provides a concise technical specification of the feature and a rationale for the feature.

That's why we can use those Improvement Proposals as support material, as they share a technical specification of the different features and the reasoning behind why that feature is needed.

Also, another good source of information are personal Blogs of the Blockchain developers, Open Source projects and Blockchain Meetups.

2. Project Scope

In this section, we introduce problem formulation, the scope of the project, which stages it will have and which are the obstacles and the risk that we had predicted in advance. At the end, we propose some alternatives if some risk becomes a problem during the project development.

2.1. Problem Formulation

The problem is formulated from a startup point of view with few resources to develop a Blockchain project. We start with the situation of:

- Only one member in the development team
- Small development budget
- The business hypothesis has to be verified
- Small market niche due to the need of users with previous experience of Blockchain applications
- The technology should allow business swifts

That being said, the project objectives are:

- Create a web application that connects to a NEM Blockchain network to receive the content created by publishers, the digital assets exchanges through the platform and viewer comments.
- Integrate the web application with NEM Wallet, allowing the user use the web application though the Blockchain.
- Usage of good practices to achieve a quality and robustness of the system

On the user point of view, the platform objective is to create a social media platform that aggregates content owned by publishers and help viewers give digital assets for different posts.

2.2. Definition scope

The scope is limited by the previous objective definition and the available time expected to be dedicated for a Bachelor's thesis. Even though being short in time, the project aims to create a Minimum Viable Product instead of a proof of concept

solution, implying to be focused on core features completeness instead of multiple features with almost no impact to the end user.

The platform features are:

- Create a Space. The *Space Owner* creates a "community" where content creators can publish their work.
- Publish a *Post*. The publishers can publish posts into Spaces.
- Publish a *Comment*. The viewers can add a comment to a *Post*.
- Give digital assets. The viewers can give digital assets to *Publishers* for a specific *Post*.

In order to achieve the objective, the work has been divided in the next stages:

1. Problem Analysis and Solution Design
2. Product Development
3. Testing Phase
4. Outcome Analysis

Once the project is finished, it should be ready to be used on production environments.

2.3. Obstacles

1. Wallets integration

The platform relies on third party wallets to interact with the platform. Due to Blockchain is *relatively* a new technology, we might find that the NEM Wallets do not offer the required features to create a complete user experience.

2. NEM API integration

Integrating the NEM API could lead into undocumented scenarios that will need deep research. Those scenarios could be undocumented API endpoints to undefined behaviour on edge case scenarios.

3. NEM as a Service

In case of running a private network of NEM, it could lead into a different scenarios related to sysadmin tasks like setting up a network and configure the security.

2.4. Risks & Alternatives

The problems related to NEM API integration should appear near the beginning of the project. Through an iterative approach, a way to mitigate the integration issues will be performed.

On the other hand, problems related to Wallets integration will be harder to be solved on the platform side, requiring to be in contact with wallet developers in case of a blocking feature. Having this external dependency could lead into a bad situation of the project. In the worst case scenario, the option of forking an open source wallet and applying the required fix will be considered.

For the scenario of running a private version, a way to mitigate the risk is using an existing private cloud managed by a third party for testing purposes.

3. Methodology and Tools

3.1. Working Methodologies

Due to the limited amount of time, the non-fixed ordering feature and the possible unexpected tasks because of a new technology, Kanban agile methodology will be used. Kanban methodology helps taking into account the non-predictable tasks in the development cycle, making them visible for the next iteration, and shipping the feature as soon as it is completed.

Every two weeks, the TFG's director and the developer will review the backlog and set a new task prioritization, taking into account the feedback of the last development cycles and the analysis of the platform usage by the early adopters.

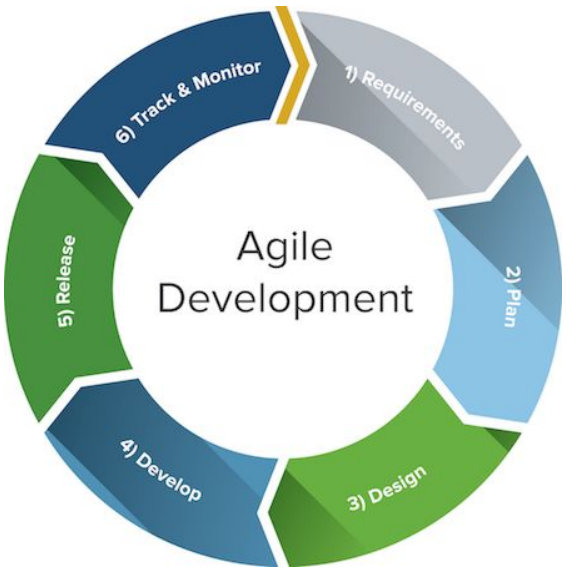


Figure 3 - Agile Methodology Cycle
Font number8.com/kanban-versus-scrum

3.2. Development Methodologies

The project aims to follow the *Extreme Programming Development Methodology* in combination with *DevOps*. As outcome, the project should stay simple and often released to the end user. Thus, *Continuous Deployment* and *Test-Driven Development* practices are applied along all the project.

The usage of Extreme Programming does not imply not doing any up-front design regarding architecture, or applying Continuous Deployment without thinking in the strategy. A proper architecture definition of the properties that the system must have will be done at the beginning of the project, reviewed during the development and analyzed at the final stage to ensure that those properties have been successfully accomplished. In relation to Continuous Delivery, it will be implemented in the very early stage of the project and maintained during the development when the deployment artifacts evolves.

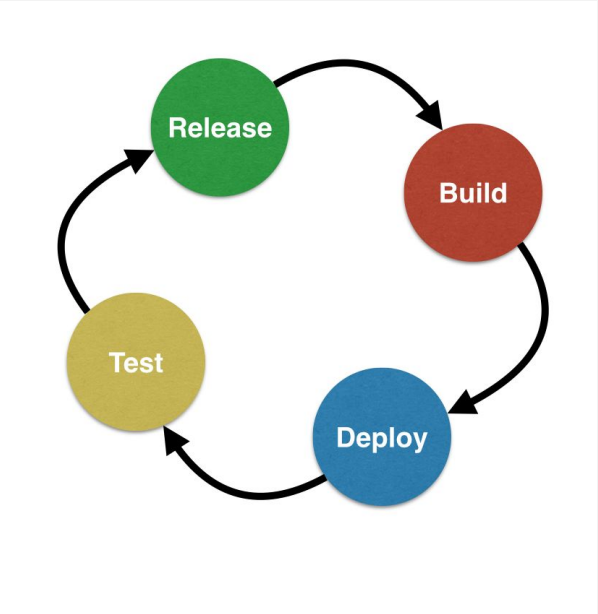


Figure 4 - Continuous Deployment Font blog.arungupta.me/continuous-integration-delivery-deployment-maturity-model/

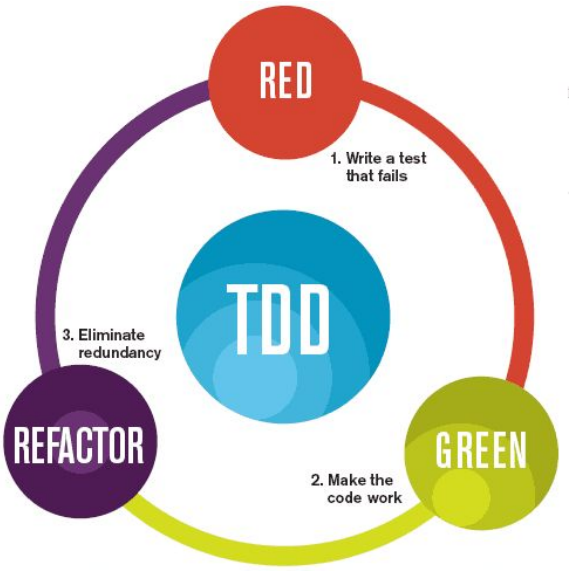


Figure 5 - Test-Driven Development Font medium.freecodecamp.org/isnt-tdd-test-driven-development-twice-the-work-why-should-you-care-4ddcabeb3df9

3.3. Tracking Tools

Git will be used as a version-control tool along with GitHub platform to publish the public project repository. We will use Issue system that GitHub offers as Task Manager for the project, it helps us in having a good control on the project status at all times.

Using the issue system, we will track the progress using the following convention:

- Milestones: Grouping user stories and tasks under a milestone we can track when a feature is complete.
- Project: To have a visual representation in columns about the state of the project with a Kanban-like column configuration.
- Issues: As a way to track the progress on a particular user story, task or bug.

Even though GitHub helps us with the tracking tool, a periodic progress check via email will be done between the project director and the development team.

3.4. Validation and Rigor

As shared on a previous section, the project will use a Test-Driven Development methodology to ensure that all the code is tested. This approach allows us to setup the

Continuous Integration system offered by Travis-CI. Using Travis-CI, we will ensure that the code does not break between different commits as a primary way to ensure that the project has the expected robustness.

On the other hand, we will perform regular checks of the system's behaviour and the completeness of the features with the project director during the sprint if necessary, but in more depth at the end of the sprint.

At the end of each sprint, the problems encountered during the development will be analyzed to find possible similar obstacles in the following sprint and, if so, apply a solution to minimize the risk.

Notice that using Test-Driven Development, I do the design phase during implementation phase. The design phase is guided by Test-Driven Development.

4. Project planning

4.1. Temporal Calendar

The project has an approximate duration of four and a half months. We take into account that its starting date coincides with the GEP starting date, the 18th of February 2019, and defined as end date, the day of the thesis defense, between Monday 1st July and Friday 5th July of 2019.

4.2. Task definition

4.2.1. Project Planning

In the planning phase we elaborate a documentation with the objective of defining which is the context, the state of the art, the temporal planning, the budget and the sustainability of the project.

This documentation is made during the GEP subject.

This phase has a duration of a month and a half, from 18th February 2019 to 29th March 2019.

4.2.2. Analysis and Design

On the analysis phase, we collect the business requirements based on the project goals and we deliver a documentation with the functional and nonfunctional requirements. An outcome of the documentation is a set of User Stories to shape the scope of the features.

Based on the requirements, we analyse the existing tools on the market that benefits us and can help in mitigating the critical system components.

Once we finish the analysis, we create a proposed solution design. The solution design should indicate the amount of components, how they interact with each other, a class diagram and persistent layer concepts, among other architectural artifacts.

The analysis and design phase needs the context made in the project planning and the temporal dedication as well. It should happen after Project Planning phase.

4.2.3. Development phase

The development phase is divided into features instead of architectural layers (for example controller, model, database).

Since we apply Agile Methodology, small interactions are done, and we expect performing the next tasks on each phase:

1. Decompose User Stories into smaller tasks
2. Analysis, design and risk mitigation of the tasks
3. Implementation of the task doing Test-Driven Development
4. Ensuring the Acceptance Criteria of each User Story is accomplished
5. Publish a working application into production environment

Notice that the development phase needs the Analysis and Design phase to be finished since it depends on the User Stories, the architectural overview and the solution design.

4.2.3.1. Setting up the environment

Before actually start developing the platform, we need the Continuous Integration system setup and our local environment setup as well. A week will be dedicated into setup the environment before actually start the first User Story.

4.2.3.2. Content Creation - Spring 1

The first sprint is dedicated to *Publisher* stakeholders. The expected features made here should allow publishers create, visualize and list the content. Due to it requires the integration with two other third party components (the Blockchain itself and the Wallet), we do not overcommit on User Stories in the sprint.

4.2.3.3. Comments, Profile and Digital Assets - Sprint 2

On the second sprint, we dedicate effort into allowing the platform become interactive by the *Viewers*. We allow the *Viewers* to add comments into *Posts*, give *Digital Assets* to the *Publishers* and check all users profiles.

4.2.3.4. Space administration - Sprint 3

The last sprint is dedicated to *Space Administrators*, we introduce the concept of *Space* or *Community* into the platform, having a major new way to structure the information in the platform.

4.2.4. Testing phase

Even though a testing is made during the development phase on each Sprint, we allocate time for in depth platform testing with edge case scenarios, like third party system not available and similar distributed system issues.

When needed, the proper code fixing will be made.

The testing phase, because it handles all the system, needs the development phase be finished.

4.2.5. Final phase

On the final phase, the memoir, the necessary project documentation and a deployable artifact will be finished.

Also, the defense presentation will be prepared in this phase.

Since the artifacts and documents need to be based on the development and testing phase, the final phase is made after testing phase.

4.3. Gantt diagram

The Gantt diagram groups the five phases and breaks them down into smaller workable tasks.

For *Project Planning*, *Analysis and Design, Test* and *Final* phases, the workable tasks are have dependency on the previous task, that is the reason why they appear in form of stairs.

On the other hand, the *Development phase* has a similar structure for all the sprints. A Sprint is divided in four tasks:

- User Stories into Smaller tasks

When a User Story is too big, it is split into smaller tasks that can be implemented with a better defined scope.

- Analysis, Design and Risk Mitigation

For each feature, I analyze how it could the architecture look like and the software design of the different components. I also take into account the different decisions risks and I take the time to mitigate those risks in case they are meaningful.

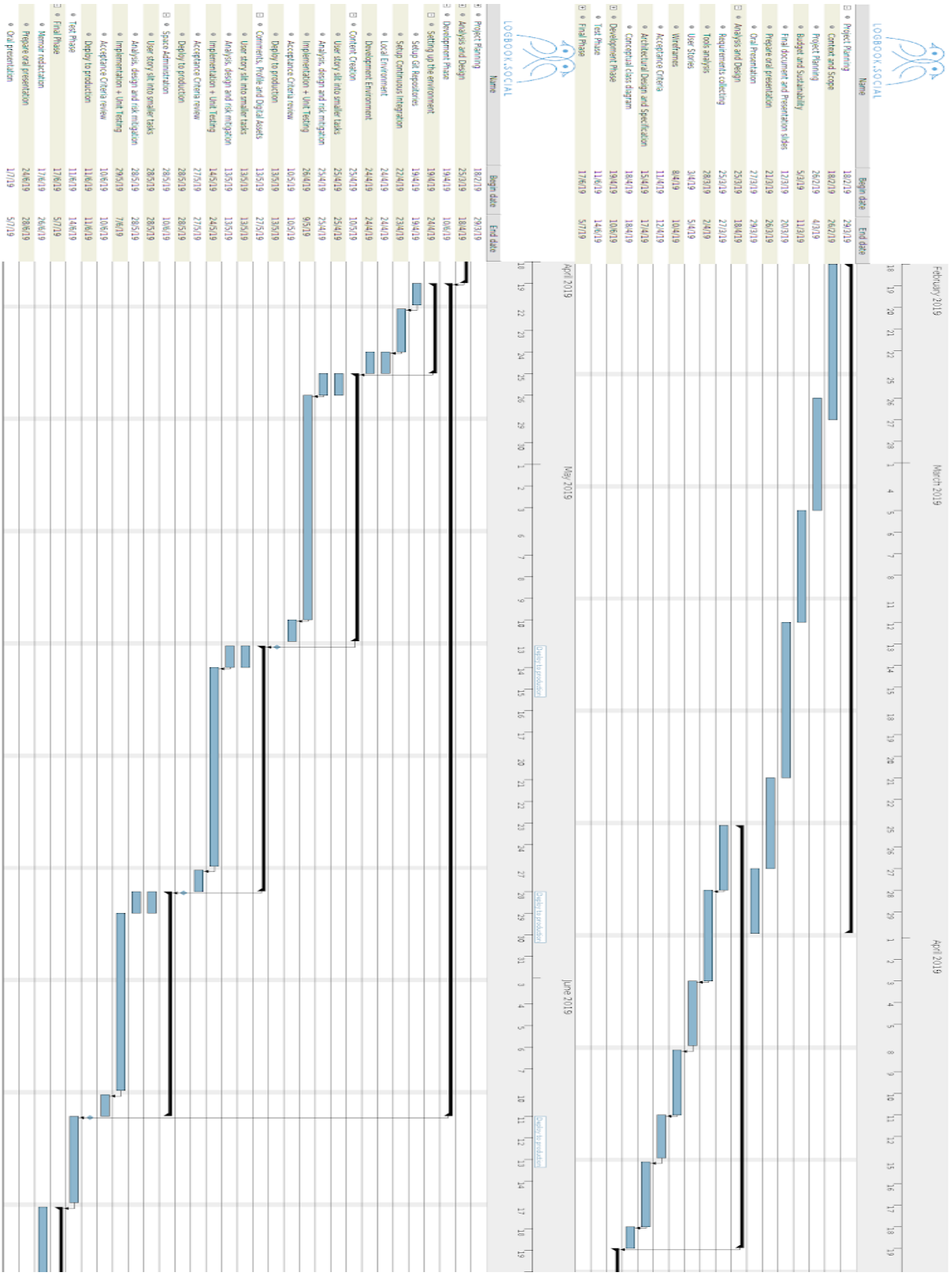
- Implementation + Testing

Almost all the time of a sprint is dedicated into the platform implementation and testing. One user story/task time, I implement the features following the Test-Driven Development.

- Acceptance Criteria Review

Once the sprint is finished, I take the time to analyse the acceptance criteria to determine that all the features are complete and I record the outcomes on this document.

There are different points at time that some tasks are overlapped, it means that some tasks might not require full day dedication and might be combined with other tasks.



4.4. Alternatives and action plan

The project can have multiple time deviation for two main causes:

1. Optimistic task timing causing a delay on finish the tasks
2. Integration with third party issues

4.4.1. Deviation based on a bad planning

There are two main scenarios:

1. A task takes less than expected.

In the scenario of being faster than expected and finish user stories before the end of the sprint, we plan to pull into current sprint the user stories planned for the next one. In case of being the last spring, the remaining time will be dedicated to add more automated testing into the platform.

2. A task takes more than expected.

In the scenario of having troubles in finishing user stories, we consider the stability of the project over unfinished big features. In case of discarding some feature, it should be related with the business impact to the users. We consider more important the features related to the *Publishers*, then the *Viewer* features, and what is related to the *Space Administrators* the less important.

4.4.2. Deviation on integrating a third party component

In case of third party integration issues, we consider the possibility of becoming active on that third party components, because they all are open source project, to unlock the blocking situation that does not allow the project progress.

In case of not finding a solution, we could consider the impact of not integrating a third party component and replace it with a self made tool for testing purposes.

4.5. Resources

To accomplish the project, we need four resource types: personal, Hardware, Software and Cloud.

4.5.1. Personals

At personal resource level, there is one developer with a 30 hours a week dedication during all the project. The developer's responsibilities are performing the planification, analysis and solution design, implementing it, and testing the final

solution to verify that it works properly and that it solves the initial problem definition, alongside preparing the Thesis defense.

4.5.2. Hardware, Software and Cloud

Resource	Type	Utility
Hardware		
Work space	Office Material	
ASUS Laptop	Development tool	
Software		
Google Drive	Office software and Cloud storage	Documentation generation, document and archive storage
IntelliJ IDEA	Development tool	Programming environment
Draw.io	Diagram creation tool	Software architecture design and documentation
GitHub	Git repository	
GitHub Issues	Task management tool	
Cloud		
Digital Ocean	Infrastructure as a Service	Web platform development

5. Social Network Research

The Project features and usability are inspired on different successful social networks, understand and analyse them is required to prevent future problems.

The main social networks analyzed are Reddit, the 3rd most visited website in the USA, and the 6th in the world, a centralized product. And Steemit, a similar equivalent in Blockchain.

5.1. Reddit

Reddit is a Social News and Media Aggregation web page that allows content rating and discussion launched in 2005.

To publish content, Reddit requires a registered member. The content can be links to other webpages, text posts and images. This content, then, can be voted up or down by other members, or replied.

The content is organized by subject into user-created boards called “subreddits”. It helps the content creator to publish his/her posts into the community that it’s more likely to be interested in the content itself.

The subreddits are moderated by Reddit employees and subreddit admins. Reddit’s administrators spend considerable resources on moderating the side.

Registered users can subscribe to subreddits to have a personalization of the content based on their preferences. Reddit suggest new alike subreddits based on those subscriptions.

In 2017, the Reddit included a new feature to publish content into the user profile instead of requiring a community. The new feature was required by the content creators due to the need to find the right place to post their content. Posting on their own profile allowed content creators grow their followers.

In regards to unique visitors, between February 2018 to July 2018, Reddit had a mean of 1.5 billion unique visitors a month.

Concept	**Amount**
People using Reddit (registered and	330 million users

non-registered users)	
How many redditors (registered users)	3.368 million
Number of subreddits*	853 824 subreddits
Number of active communities* (active subreddits)	140 000 communities
Number of monthly pageviews	14 billion
Number of searches	40 million searches
Average visit length	15 minutes
Average number of monthly post submissions**	11 million
Percentage of piracy takedown notices that were rejected by Reddit moderators	81%
Number of comments left on Reddit daily**	2.8 million
Average of daily votes on Reddit**	58 million votes
Reported value of Reddit	$1.8 billion

Table 4..1

5.1.1. Analysing implications/similarities to LogBook.Social

Table 4.1:

* Represents the equivalent of creation of a space. It just happens once per space.
** Represents the posting and commenting. It happens often. This is the **bottleneck** of LogBook.Social.

The percentage of registered users is 1.02% of the total users that visits the website. Meanwhile, the active subreddits is the 16.40% of the total subreddits registered.

354 839 posts are published each day, each of those posts receive an average of 7.9 comments.

Just taking into account the posts made daily, 354 839, it's more than double that NEM can perform within a day, 172 800 transactions per day theoretically, if LogBook.Social was the only application used.

Normalizing all the interactions to *interaction per second*, we have:

- 4 posts per second
- 32 comments per second
- 671 votes per second

It means that NEM should support up to **707 transactions per second** to handle the Reddit traffic.

Definitely, NEM cannot be the platform to run a successful website as Reddit is because of write throughput, at least if the *public chain* is used. In case of reading, LogBook.Social does not have a problem to scale reading because of the usage of common technologies used to scale web applications.

Since NEM public blockchain is not enough powerful, given the threshold of max capacity, an alternative to scale the write operations will be studied with the private ledger option.

Finally, the feature of content personalization in LogBook.Social will not be possible since it does not require the users to be registered in the platform to publish content.

5.2. Steemit

Launched in March 2016, Steemit is a blogging and social networking website that uses the Steem Blockchain to reward publishers.

The platform has a similar editor feature to Medium.com to publish posts. It focuses on the content published by users instead of aggregating news from outside the platform.

Steem blockchain provides a throughput of 1000 transactions per second, more than enough to support a website as Reddit. Also, it has not a fee system for the users that create content. Steem blockchain has been created specifically for not charge the user that create value, so it implies a considerable advantage over his competitors.

The features are similar to Reddit, it allows the user to publish his/her post into a specific community, reply to a post and vote.

5.2.1. Security

On July 14, 2016, Steemit announced on their website that they were hacked. The attack, according to them, has compromised about 260 accounts. About US$85,000 worth of Steem Dollars and Steem are reported to have been taken by the attackers.

5.3. Comparison with LogBook.Social

Feature comparison table:

Feature	**Reddit**	**Steemit**	**LogBook.Social**
Post	Yes	Yes	Yes
Comments	Yes	Yes	Yes
Nested comments	Yes	Yes	No
Up/Down Votes	Yes	No	No
Up vote with a tip	No	Yes	No
Direct tip	No	No	Yes
Enriched editor	Yes	Yes	Maybe
Communities/Spaces	Yes	Yes	Yes
Alias system	No*	No*	Yes
Images	Yes	Yes	No
Links	Yes	Yes	No
Blockchain	No	Yes	Yes
Dedicated Blockchain Incentives**	-	Yes	No
Dedicated Wallet	-	Yes	No

* It has a username system already.

** Dedicated Blockchain Incentives: The fee system takes into account the use case.

6. Specification

The Specification chapter lists the platform features using *Specification By Example* technique to define the business requirements and establishes a *Definition of Done* with a list of criteria which each feature must be met in form of *Acceptance Tests*.

The features might have different wireframes as a visual guide for the representation of the website.

Some features requires the user interaction with the NEM Blockchain in his/her side, the interaction method will be explained in the feature, and marked as external interaction not dependant of the platform per se.

The specification uses NEM Blockchain nomenclature, check Annex I for concepts description.

6.1. Posts

The platform lists the posts for a specific space,

The user lists all posts and a button showing a guide of how to publish a post. It's important to notice that the **publishing a post happens outside the platform**, on NEM Blockchain.

6.1.1. Listing

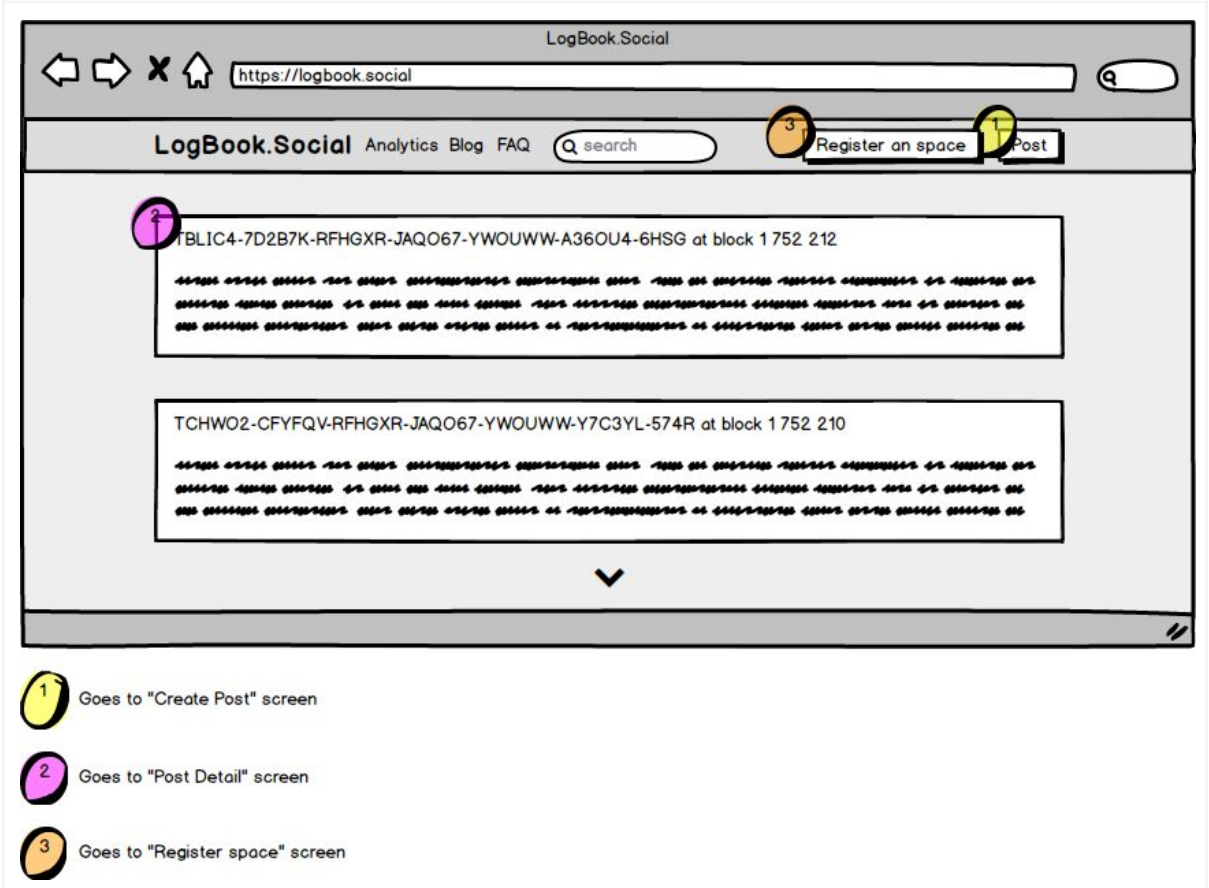
The viewer sees a list of posts of a specific space. Each post has a way to check the source of truth.

Acceptance Testing

```
Feature: List Posts
    In order to view a Post
    As a viewer
    I want to view a list of posts

    Scenario: Exists a set of Posts
        Given a set of Posts previously created
        When a viewer goes to the web application
        Then a list of posts with title, piece of the body text
and the author name are shown.
```

```
    Scenario: No posts
        Given an empty set of Posts
        When a viewer goes to the web application
        Then the application shown an icon informing that no posts
have been created yet
        And a call to action to create one
```

Mockups



6.1.2. Posting

The publisher fills a simple form with the title and content, it generates two possibilities for the user, once he/she has the content, the process of announcing a transaction starts (see 5.6. Announcing a Transaction).

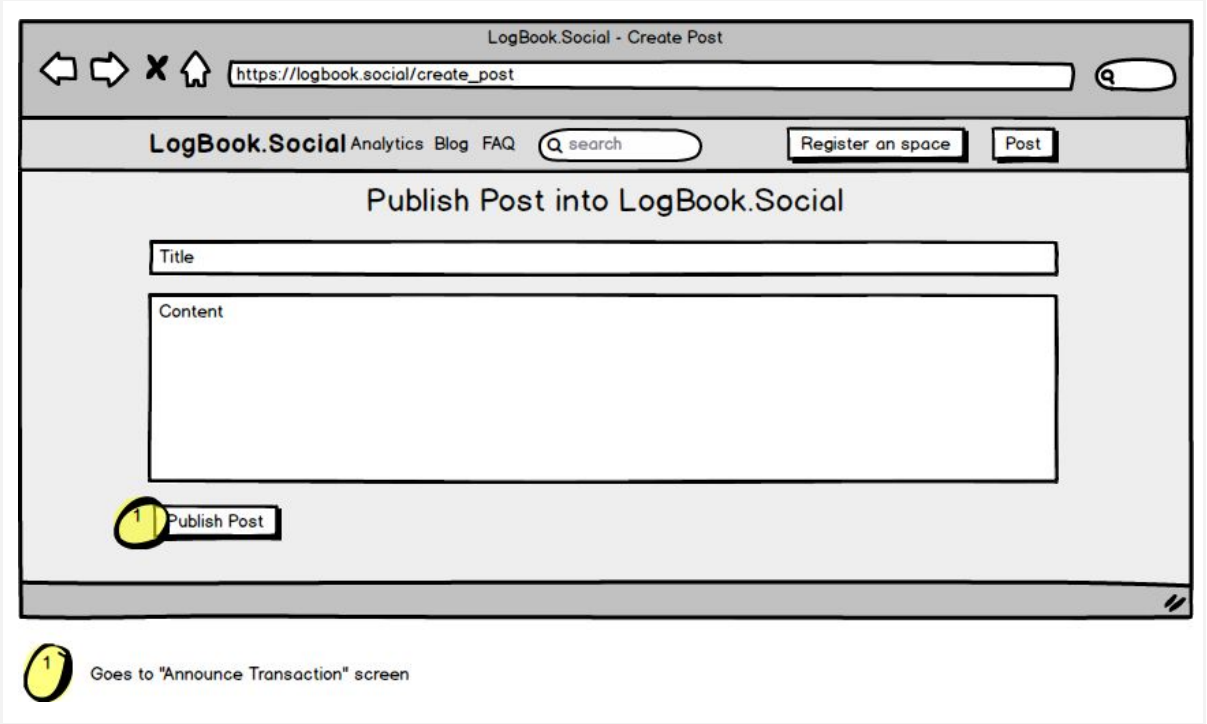
If the publisher has shared its account address, the application will start monitoring the network to provide a nicer user experience, notifying about the state of the

transaction. In case the publisher has not shared its account address, then the application will ask for it.

Acceptance Testing

```
@posting
Feature: Publish a Post
    In order to publish a post
    As a publisher
    I want to know to which NEM Address I have to send the
transaction with the message

    Scenario: Publisher uses a mobile wallet and interacts with
the web
        Given a QR code with the specific address
        When the publisher scans it and sends a transfer
transaction with a plain message
        Then the message with the publisher account address and
the block height the transaction is included in a block appears in
the post list

    Scenario: Publisher uses a desktop
        Given a post transaction
        When the publisher opens the NEM2 Wallet and sends a
transfer transaction with a plain message
        Then the message with the publisher account address and
the block height the transaction is included in a block appears in
the post list
```

Mockups



Abstract of protocol specification

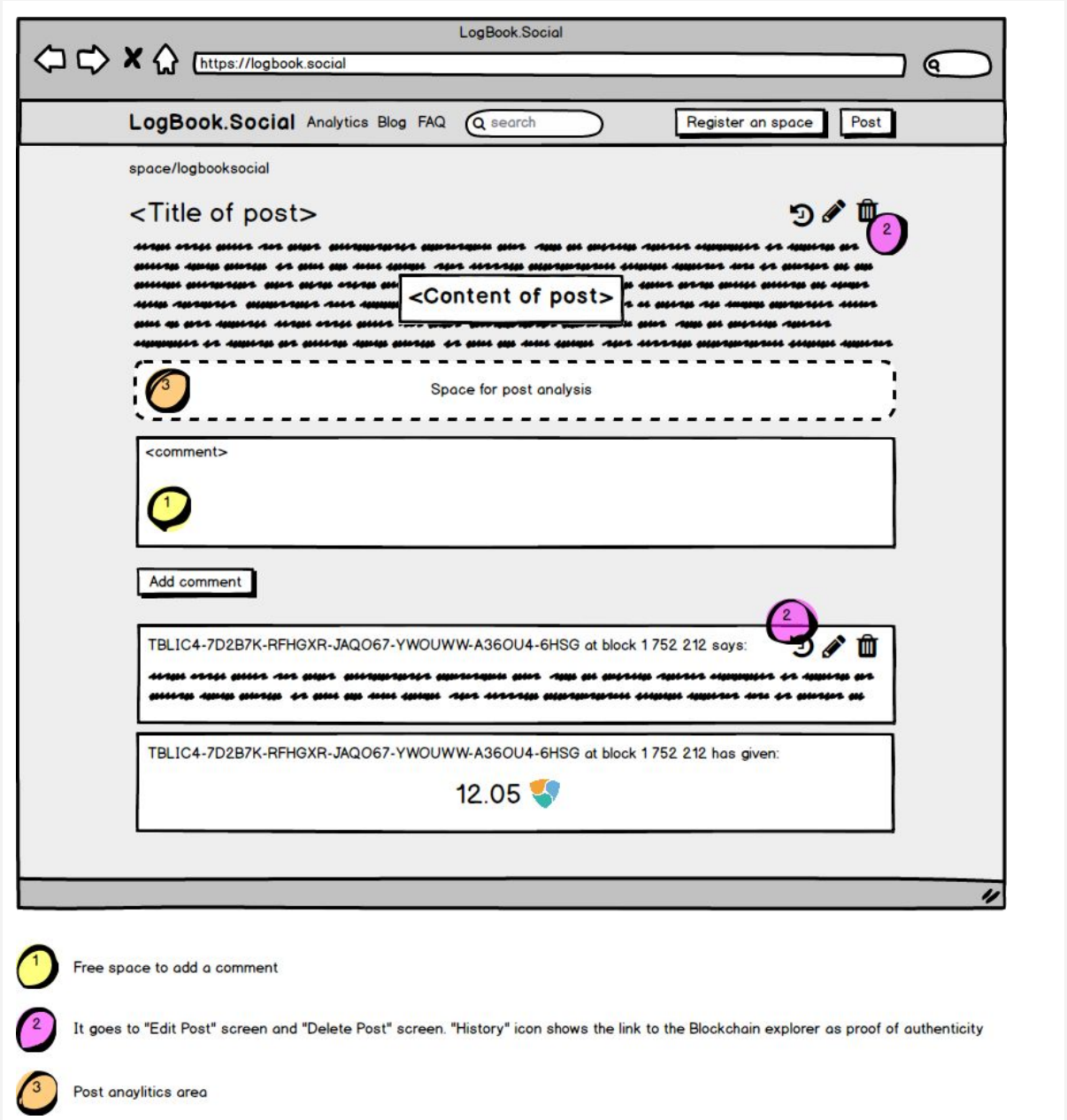
The *publisher* sends a *transfer transaction* with a *plain message* to a *specific address* using a *normal account* or *multisig account*. Then, the platform looks proactively incoming transactions for the *specific address*, if the transaction is a *transfer transaction* or a *multisig transaction* with a *transfer transaction* inside, it gets included in the platform, otherwise it is discarded.

The platform stores the transaction hash to allow viewers to check the post equivalent in the NEM Explorer.

6.1.2. Detail

The Post Detail should display the information the author posted, title, content and the author username, plus the comments and digital assets that the post has received.

Mockups



6.1.3. Editing

The publisher is able to change the content of a post when he/she is the original publisher. Replacing the whole content or adding more content at the end of.

The method to edit the content and publish the changes is the same as Posting.

Acceptance Test

```
@post_editing
Feature: Adding content to a post
```

```
    In order add content to an existing post
    As a post owner
    I want to reference the post in particular and add the text I
want to append.

    Scenario: The signer isn't the post publisher
        Given adding content to a post transaction
        When the signer isn't the post publisher
        Then the transaction is ignored by the system

    Scenario: The signer is the post publisher
        Given adding content to a post transaction
        When the signer is the post publisher
        Then the content is added at the end of the post
```

```
@post_editing
Feature: Replacing the content of a post
    In order to replace the post content
    As a post owner
    I want to reference the post in particular and add the text I
want to replace by.

    Scenario: The signer isn't the post publisher
        Given changing content of a post transaction
        When the signer isn't the post publisher
        Then the transaction is ignored by the system

    Scenario: The signer is the post publisher
        Given changing content of a post transaction
        When the signer is the post publisher
        Then the post content is replaced by the new content
```

6.1.4. Logical Remove

The post author is able to unlist the post using thought a button in the post, then starts the Announce Transaction process (see 5.6. Announcing a Transaction).

Acceptance Test

```
@post_removing
Feature: Removing a post
    In order to remove a post
    As a post owner
    I want to reference the post in particular to be removed.

    Scenario: The signer isn't the post publisher
        Given removing a post transaction
```

```
        When the signer isn't the post publisher
        Then the transaction is ignored by the system

    Scenario: The signer is the post publisher
        Given removing a post transaction
        When the signer is the post publisher
        Then the post is removed
```

6.2. Spaces

6.2.1. Registering

In order to **register a new space**, the *space creator* has to own the space name as NEM Namespace. The process requires the user to introduce the address he/she will be registering the space, a list of namespaces the account owns appears and the user selects the namespace he/she wants to register as LogBook.Social space, then starts the Announcing Transaction process (see 5.6. Announcing a Transaction).

Linking a Space to a *NEM Namespace allows to control the creation and moderation* of spaces via NEM Blockchain instead of control by LogBook.Social. **Keeping the space ownership to the NEM Namespace owner**.

Acceptance Test

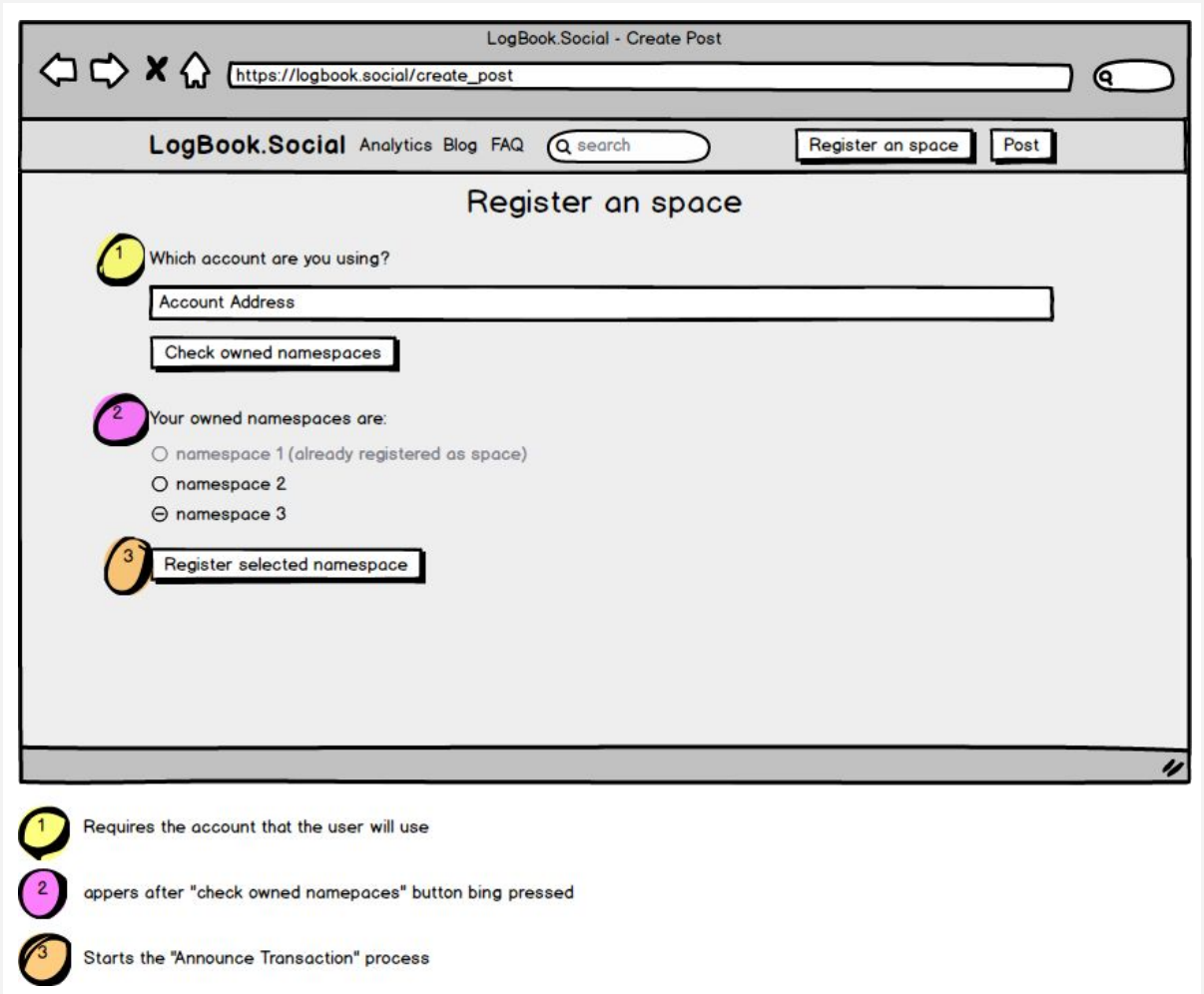
```
@registering
Feature: Checking space availability
    In order to check space availability
    As a space registrator
    I want to know the space availability
        and account with rights to register it

    Scenario: Space is not registered in NEM
        Given a space identifier
        When it's not registered in NEM as namespace
        Then display the availability and the process to register
it

    Scenario: Space is registered in NEM but not in LogBook.Social
        Given a space identifier
        When it's registered in NEM as namespace
            And it's not already configured in LogBook.Social
        Then display the availability
            And the account with register rights
            And a button to continue the registration
```

```
    Scenario: Space is registered in NEM and in LogBook.Social
        Given a space identifier
        When it's registered in NEM as namespace
            And it's already configured in LogBook.Social
        Then it displays that it's already taken
            And shows a button to go to the Space.

    Scenario: Space is registered in NEM
                and it's partial configured in LogBook.Social
        Given a space identifier
        When it's registered in NEM as namespace
            And it has pending steps to be configured in
LogBook.Social
        Then it displays the status of pending to be configured
            And shows a button to continue the registration
process
```

```
@registering
Feature: Registering the space
    In order to register a space
    As a space registrator
    I want to send the required transactions to notify
LogBook.Social
        to register the new space

    Scenario: The space isn't available for registering
        Given an invalid space availability
        """
        See Feature: Checking space availability
        """
        When a space registrator tries to register it
        Then the system ignores the registration process

    Scenario: The space is available
                and the signer isn't the namespace owner account
        Given a valid space identifier
            And the signer isn't the namespace owner account
        When it starts the registration process
        Then the system notifies that the registration isn't
performed by the valid account
            And shows the account address that should be
performing the action

    Scenario: The space is available
                and the signer is the namespace owner account
        Given a valid space identifier
            And the signer is the namespace owner account
        When the space registrator sends the required transactions
        """
```

```
        Check registration transactions specification
        """
        Then the space is available in LogBook.Social

    Scenario: The space registration procedure isn't complete
        Given a space registration procedure that isn't complete
        When a space registrator checks to register it again
        Then it continues from the last step the process is
```

Mockups



6.2.2. Administration

The space administrator is able to remove posts made by publishers and ban users from the posting in its space.

Acceptance Test

```
@space
Feature: Administration
  In order to administrate a space,
  As space administrator,
  I want to remove posts or ban an account

  Scenario: Removing a Post
      Given a Post in a specific space
      When the space admin of that specific space asks for
removal
      Then the post is not shown anymore on the specific space

  Scenario: Banning a User
      Given a user
      When the space admin bans that user for a specific space
      Then all the content of that user is not shown inside the
specific space
```

6.3. Username Alias

The publisher can specify which username based on a namespace he/she owns to be displayed instead of his/her address.

In order to choose the alias, it requires the publisher introduce the address he/she is using. Once the namespaces owned by that address is listed, the publisher selects the namespace he/she wants to use as alias and then starts the Announcing Transaction process (see 5.6. Announcing a Transaction).

Acceptance Test

```
@alias
Feature: Choosing a username based on namespace ownership
    In order to use a username
    As a publisher
    I want to know specify which namespace I own I want to use as
an address alias (username).

    Scenario: Publisher chooses a namespace as alias that he/she
owns
        Given a transaction with the namespace chosen
            And the signer owns that namespace
        When it's received by a specific account of the system
        Then all past posts by the publisher get the username as
```

```
specified
            And future posts as well

    Scenario: Publisher chooses a namespace as alias that he/she
doesn't own
        Given a transaction with the namespace chosen
            And the signer doesn't own the namespace
        When it's received by a specific account of the system
        Then it's ignored
```

6.4. User profile

The user profile shows the list of posts made by that account, the spaces it owns and the amount of digital assets received and given.

Acceptance Test

```
Feature: User Profile
  In order to see a user activity,
  As viewer,
  I want to see its user profile

  Scenario: User exists and has activity
      Given a user with activity
      When a viewer see its user profile
      Then it sees the post created by that user,
      And the spaces he/she owns if so
      And the comments he/she did on posts
      And the amount of digital assets received-given.

  Scenario: User exists and has no activity
      Given a user that exists but without activity
      When a viewer see its user profile
      Then it sees the basic info
      And a message saying it has no activity

  Scenario: User does not exist
      Given a user that do not exist
      When a viewer tries to access that profile
      Then it sees a message informing that user profile do not
exists
```

6.5. Search

Searching has to return results based on Spaces, Users and Posts.

Acceptance Test

```
@search
Feature: Searching
  In order to find relevant content on the site
  As viewer
  I want to search on the side on different and relevant fields

  Scenario: Searching in all relevant fields
      Given a text to search
      When the viewer does the search action
      Then posts, users and spaces list are shown.
```

6.6. Assets

The platform incentivise to spend assets to promote the content or the user give some assets to the content creators.

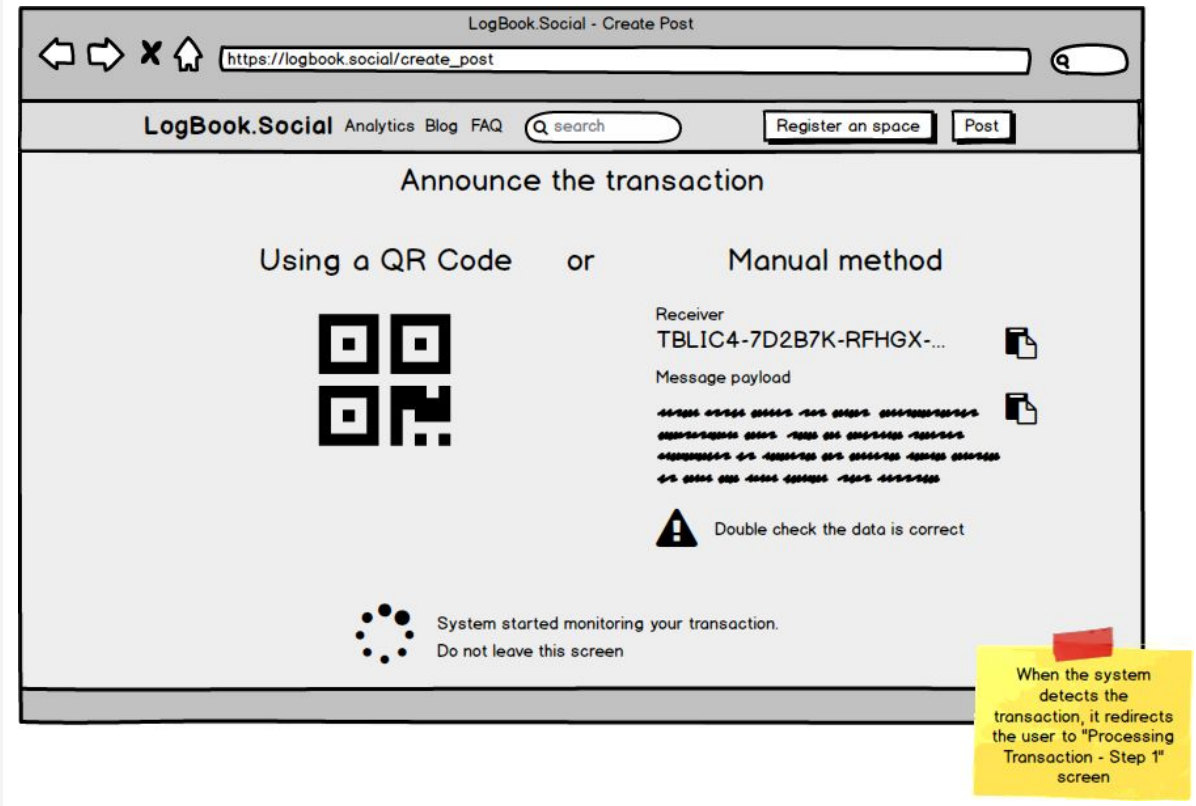
6.6.1. Donating digital assets

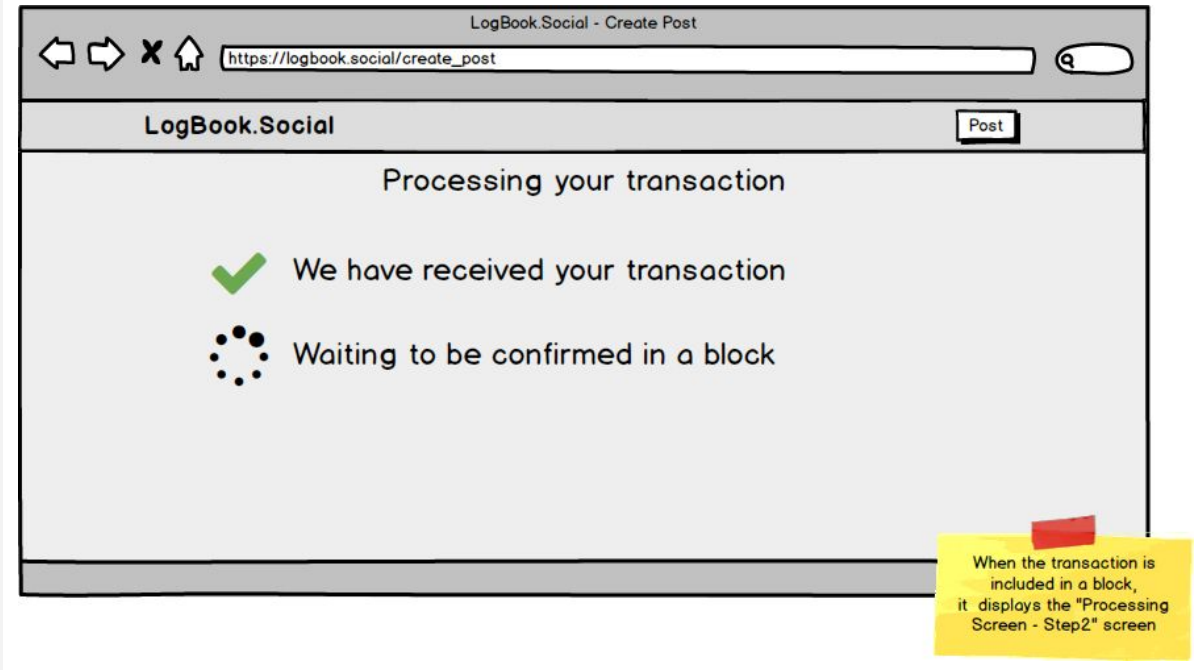
Acceptance Test

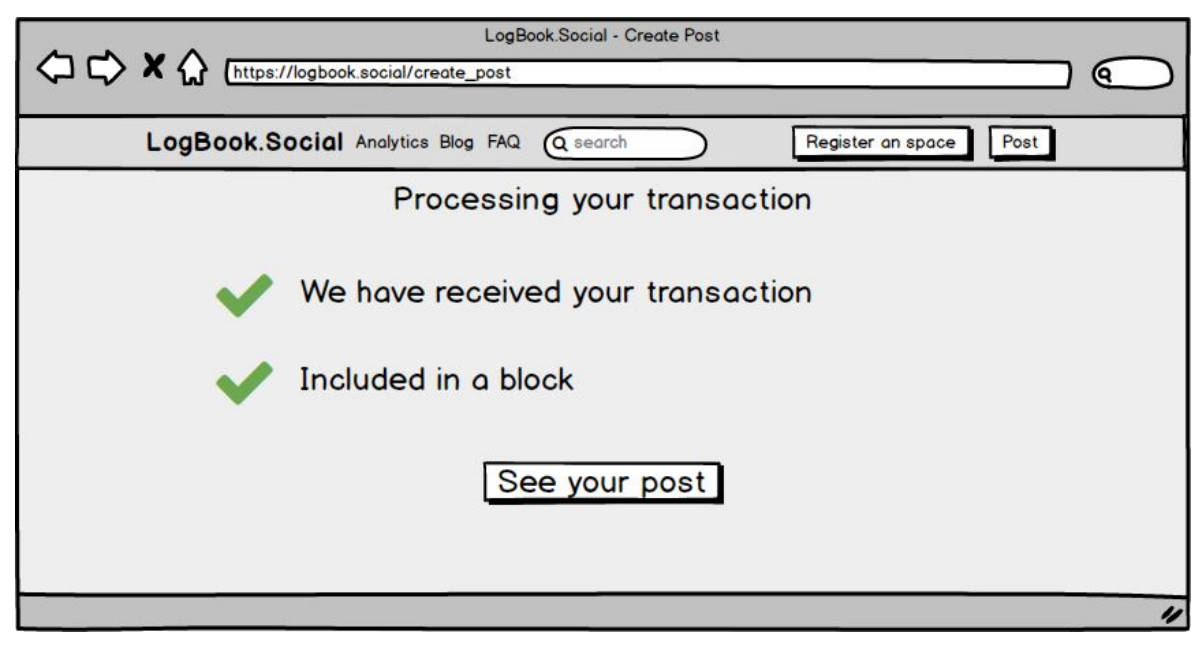
```
Feature: Donate digital asset
  In order to donate digital assets
  As viewer
  I want to click the donate button next to the post

  Scenario: Viewer donated some digital assets to post creator
    Given a digital donation to a post creator
    When it is included on the Blockchain
    Then the post has a reference to the donation
    And the donator has that donation registered to their user
profile
    And the post creator has that donation to their user profile
```

6.7. Announcing a Transaction

The experience of interacting with NEM Blockchain though LogBook.Social webapp is the same for all features.


LogBook.Social - Create Post
https://logbook.social/create_post
LogBook.Social
Analytics
Blog
FAQ
search
Register an space
Post
Announce the transaction
Using a QR Code
or
Manual method
Receiver
TBLIC4-7D2B7K-RFHGX-...
Message payload
Double check the data is correct
System started monitoring your transaction.
Do not leave this screen
When the system detects the transaction, it redirects the user to "Processing Transaction - Step 1" screen


LogBook.Social - Create Post
https://logbook.social/create_post
LogBook.Social
Post
Processing your transaction
We have received your transaction
Waiting to be confirmed in a block
When the transaction is included in a block, it displays the "Processing Screen - Step2" screen



More information about how it works can be found on section 12.3.2. Approaches to create content.

6.8. Conceptual Map

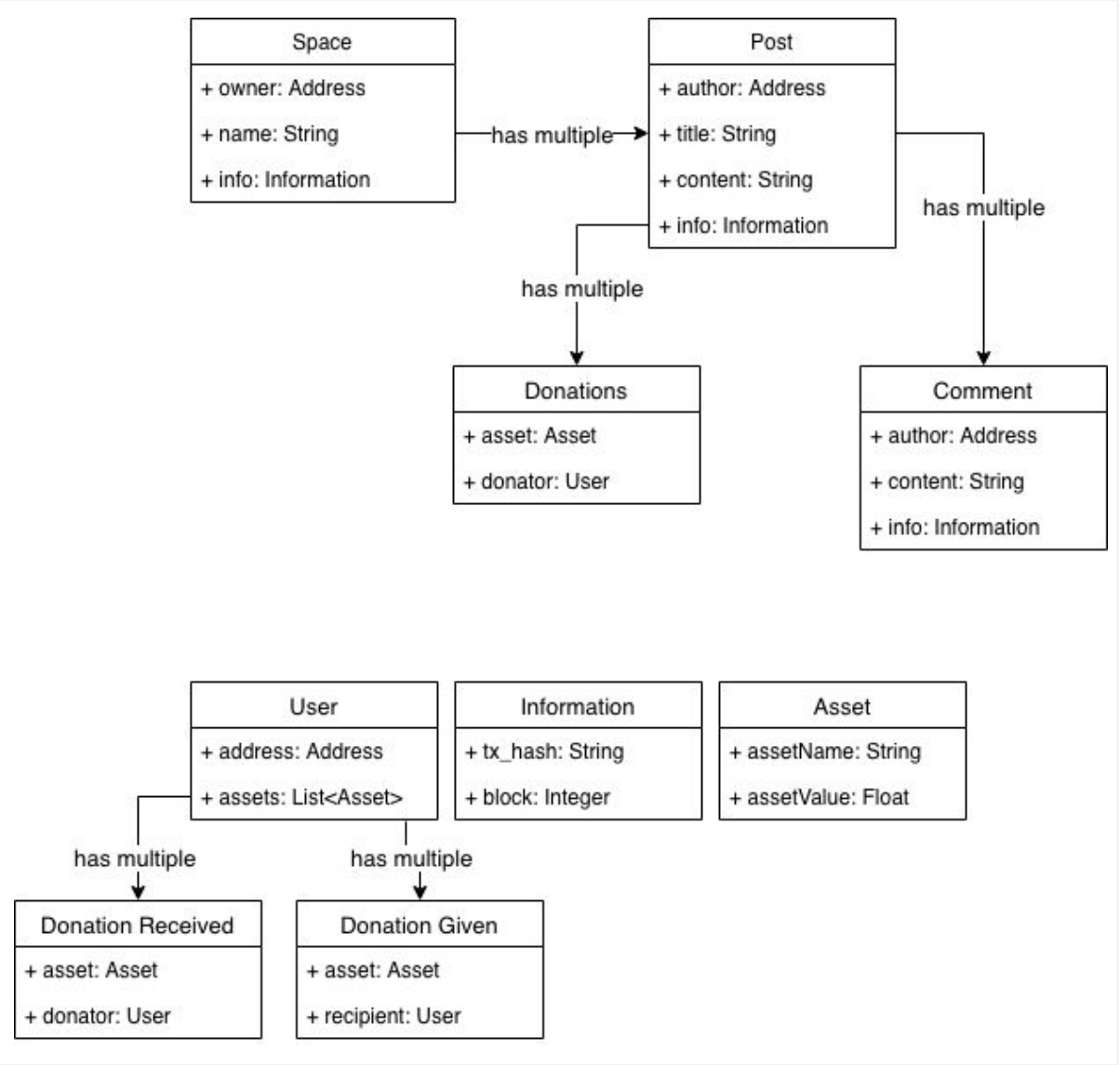


Figure 6

7. Non-Functional Requirements

The system should ensure different non-functional requirements. As a social-media platform that allows digital assets donations between users, we should consider different requirements to ensure the project viability and future growth.

7.1. System requirements

The system requirements are based on the two, of the many, value propositions that Blockchain provides. Security and Transparency.

The platform has to ensure that the user has ownership of the assets and he/she has full control, in that manner we have the Security requirement.

On the other hand, Blockchain is about Transparency and Trust. In order to achieve this requirement, we will use the common components that a Blockchain offer, the Explorer and the API.

- Security

The system must not hold any private key (Blockchain account) to ensure the digital assets are held only by the users and not delegated to the platform.

Satisfaction criteria:

The WebApp and the Server must not use the class `Account` of the `nem2-sdk`, since that class is able to hold `private keys`.

- Transparency

The platform has to provide ways to help the user be sure the data he/she is checking is securely stored on Blockchain.

Satisfaction criteria:

Each content created by a user should have a link to the NEM2 Explorer to let the user verify the transaction related to the content itself.

7.2. Legal requirements

The platform is sensible to two main laws:

7.2.1. General Data Protection Regulation - European Union Law

The GDPR law has a huge impact on Blockchain applications since it has a direct conflict with how the technology works (immutability) and the *Right to erasure*.

The platform has a huge constrain on storing personal data of the different users. Since part of the Minimum Viable Product is make it fully decentralized to verify the business hypothesis, we have the constraint of not asking nor storing the personal user data for now.

7.2.2. “Impuesto sobre Sucesiones y Donaciones” - Spanish Law

The platform might act as a donation channel between *Content Creator* and *Viewers*. A Viewer can transfer digital assets that might have a monetary value on the market, that can be considered a Donation.

Since a platform as it is described here acts globally and cannot only take into account the Spanish law, an expert attorney is needed to know how the platform has to address this scenario and determine the responsibilities of the different parties.

Notice that the assets used during the project development have no value since they are on a testing network.

8. Architectural Design

The architectural design will reflect the different components that compose the system. I aim to give an overview to help follow the different components responsibilities and how they collaborate.

8.1. System overview

The system overview shares a the main components that a user perceives.

The main component build on this project is the LogBook.Social Application. The Blockchain Wallet and Blockchain Network are components external to the Application but required for the Application to work.

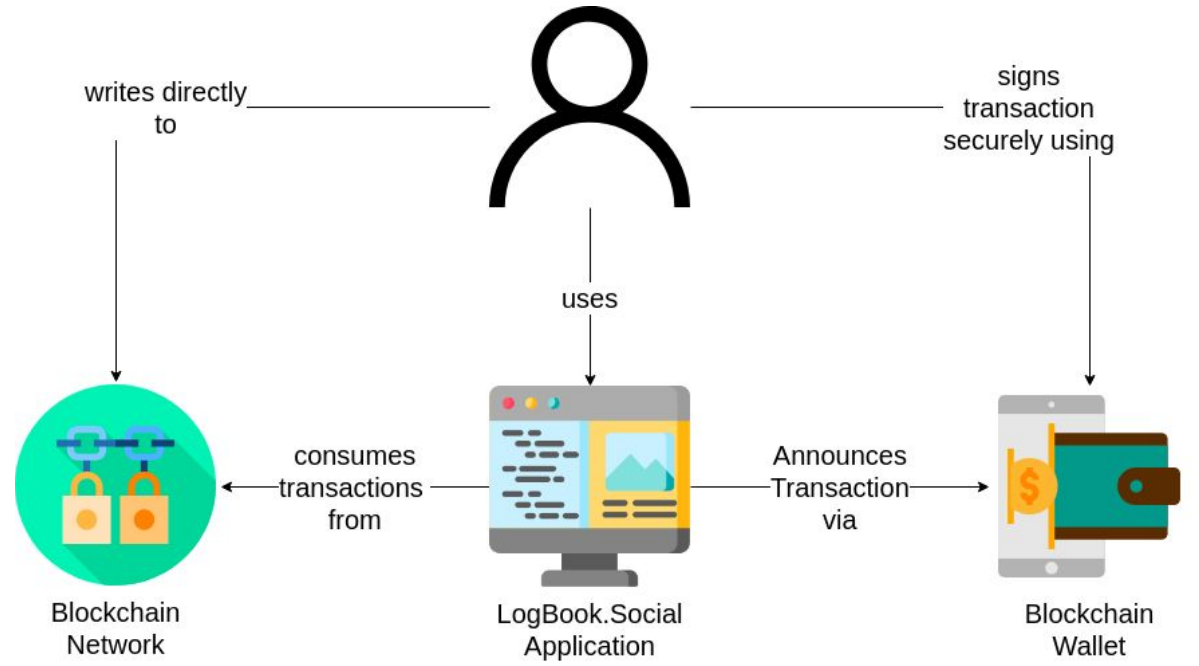


Figure 7

- LogBook.Social Application

Allows publishers, viewers and space administrators view the information and manage it via a nice web application.

- Blockchain Network

It ensures the data integrity and the authority of the created by the publishers, viewers and space administrators.

The user can verify the LogBook.Social content directly with the Blockchain Network via the Blockchain Explorer.

- Blockchain Wallet

The Wallets acts as a middle software between Blockchain Network and LogBook.Social Application. It ensures the user reminds informed about its account status and he/she is on hold of all digital assets.

8.2. Components

The LogBook.Social Application is split into four differentiated components, which only two of those are implemented.

Those four components compose LogBook.Social platform developed on this project. The Figure 8 shares also the Blockchain Wallet and the Blockchain Network to reflect which of the four components are really integrated with the external components.

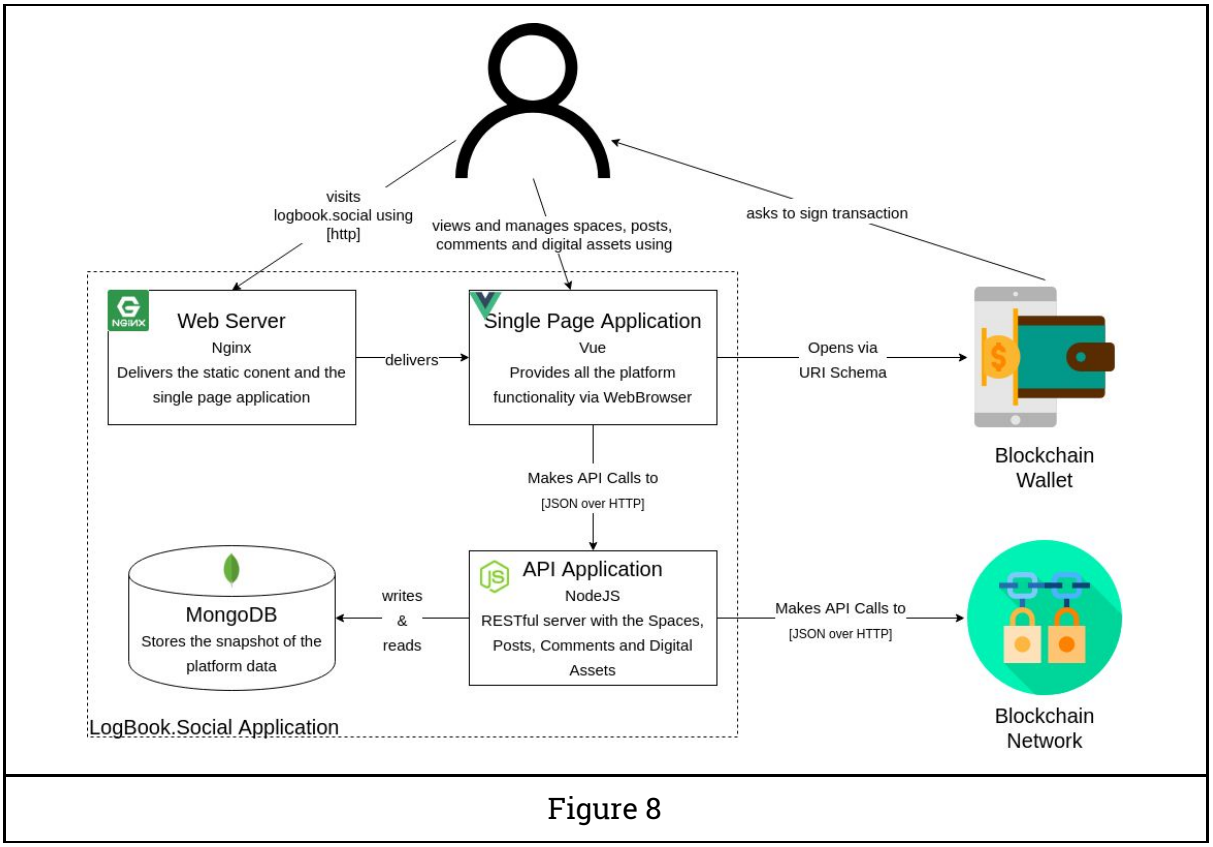


Figure 8

The LogBook.Social application is composed by 4 components (inside dotted square):

- Nginx as Web Server

The Web Server delivers the static content and the LogBook.Social Single Page Application. It responsibility, as a web server, is to handle the external connections and redirect them into the virtual private network.

It, also, has the responsibility to encrypt the HTTP traffic using HTTPS.

- Vue as Single Page Application

As Single Page Application, it contains all the features to manage the LogBook.Social application via a Web Browser.

It has to manage the integration with the Blockchain Wallet too.

- NodeJS as API Application

The RESTful API Server, has the responsibility to handle the REST requests made by the Single Page Application though the Web Server.

The Server also needs to fetch the data from Blockchain Network and ensure the data between the Blockchain and Mongo is consistent.

- MongoDB as Cache Database

MongoDB, as NoSQL Database, stores Snapshots of the Data stored on Blockchain. It is behaving as a Cache for different reasons, as performance, searching options and data enrichment.

8.2.1. Component interactions

How the components interact with each other was implemented and designing during the Sprint 2. More information about the different interaction approaches and the final decision can be found at point 9.7. Storing Information in Blockchain.

9. Implementation

On the development section, I focus on three main points:

1. Planification
2. Final outcome and its deviations
3. Challenges and decisions made during the sprints

With these three points, I consider covered the important outcomes that I can face during the development and explain the reasons behind the deviations and how I acted to correct, if necessary, that deviation.

9.1. Deviations after analysis

During the analysis, I started challenging the main risk points that I could face during the development to be able to react as soon as I detect them and they can be a real risk for the project viability.

The Blockchain Wallet integration had a high probability to become a risk for the project. That's why I take the action to minimize the impact.

9.1.1. Blockchain Wallet and URI Schema integrations

As identified during the analysis, the wallet integration is a risk that could negatively affect the project delivery in case it does not provide the required features that the project expects.

Since the URI Schema was not integrated any Wallet, an effort has been made to create an Open Source Wallet with the URI Schema integration as a value proposition the NEM Ecosystem.

In order to minimize the impact in the project two actions have been made.

- **Be involved on the URI Schema definition and library**

The URI Schema as a central point of integration between Web Application and Wallet is defined as a protocol on the official repository.

The protocol, even though it has been defined some months ago, it has not been accepted as an official protocol yet. I have chosen to follow the draft protocol and help

on the library that implements that draft protocol giving feedback and being a one-time collaborator.

https://github.com/nemtech/NIP/pull/7

https://github.com/dgarcia360/nem2-uri-scheme

- **Engage the Open Source community on the creation of a Browser Wallet**

On the Wallet side I had to make the action of starting a Browser Wallet and involve the NEM Developer Community on the process to create a wallet with the features this projects needs.

I had studied different options:

1. Fork a Mobile Wallet and just support mobile application integration
2. Collaborate on experimental wallets already on the market
3. Integrate the wallet directly on the Web Application
4. Start a Browser Wallet from scratch

I finally chose the option of creating a Browser Wallet (option 4) from scratch because:

1. A Browser Extension would minimize the user entry because due to an easy installation mode that the users would be more familiar
2. The URI Scheme support the Browser Extension API provides is powerful and do no require a complex configuration
3. Multi platform support
4. Could receive more support from the Open Source Community because of common technologies

Up to today, the Browser Wallet has the minimum features that a Blockchain User might need with the URI Schema support. In case during the development, there is some missing feature or bug, it is possible some effort should be dedicated again to the wallet even though it was not in the scope on the initial planning.

https://github.com/aleixmorgadas/nem2-wallet-browserextension

9.1.2. Implications to the planning

The Sprint 1 was planned to be focused on *Content Creation*. The *Content Creation* is the first feature that all others depend on. But, *Content Creation* directly depends on the Wallet, I took the decision of focusing the Sprint 1 into creating the Wallet.

Sprint planning re-prioritized:

#	Original Planning	Implied a change?	Actual Planning
0	Setting up the project	No	Setting up the project
1	Content Creation	Postponed for next sprint	**NEM2 Wallet**
2	Comments, Profiles and Digital Assets	Postponed for next sprint	Content Creation
3	Space Administration	Out of Scope	Comments, Profiles and Digital Assets

9.2. Deviations during implementation

9.2.1. Test Network become unstable

During the week of 2th June, the platform becomes unstable because the testing network was not including transactions on the Blockchain. After some research and debugging, I found that multiple developers reported the same behaviour on the public Slack on the same time period.

The issue with not including transactions into the Blockchain was that, the node used had a chain fork[2]. The test net nodes had a Blockchain that diverged at some point and the consensus algorithm was unable to resolve the fork.

Because the problem remained for at least a week, I decided to create my own test net network with just one node.

[2] https://en.wikipedia.org/wiki/Fork_(blockchain)

Before, the platform was managed by myself and the nem2 blockchain node was managed by a third party as shown in figure 9.

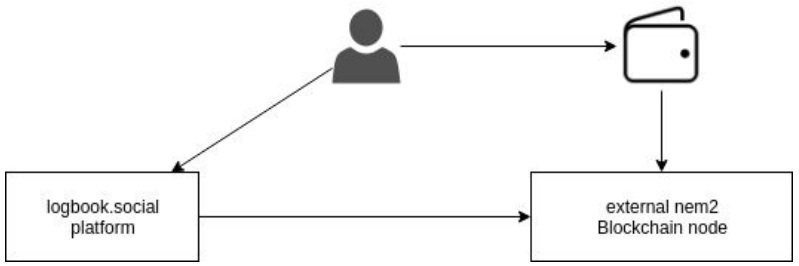


Figure 9

After, I created a managed testing network under logbook.social platform. See Figure 10.

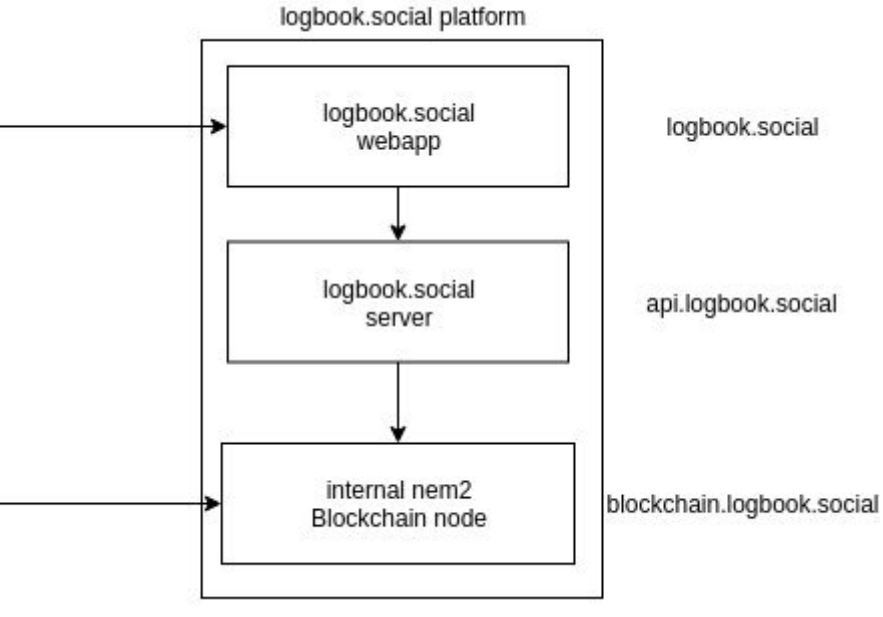


Figure 10

9.2.1.1. Implications

The network issue did not have a huge impact into the project features as the wallet did, but it had another implications.

- Material costs increased

The Material costs have to take into account the new Digital Ocean instance needed to support the platform.

- Only one node dedicated to the Blockchain network

Even though the project is a Minimum Viable Product that did not require a fully working network, only having one node in the Blockchain network does not help to emulate a real production environment. Before going into production, a testing network emulating a production network would be needed to be sure the system behaves correctly on multiple node setup.

9.2.2. New Blockchain version without migration procedure

During the week of 3rd of June, NEM2 got an update at Blockchain level and SDKs[3]. This updated included various enhancements but also it is not 100% backwards compatible. It implied that the project was affected by this update.

Since the NEM2 Blockchain version used during this project is in beta phase, it was expected to have breaking changes between versions during the development as shared on the section 4.4.2.

In order to mitigate the impact on the project deadline and stability, I decided to fix the version into the Blockchain version named Cow and SDKs version 0.11.X, which are those compatible with Cow Blockchain version.

9.3. Sprint 0 - Setting up the project

The Sprint 0 objective is to setup the Continuous Deployment System, the web application and server repositories.

9.3.1. Sprint Planning

After the Sprint 0 finished, as a Developer, I should be able to push into master branch and:

---

3 https://forum.nem.io/t/nem-foundation-technology-department-update-june-2019/22842

1. Jenkins run the test suit
    a. Report in case of failure
2. Jenkins Publishes a Docker image
3. Jenkins starts the production system with the latest changes

Doing so, I ensure that:

- The only way to publish into production environment is though Continuous Deployment System
- In order to deploy into the production environment, all tests has succeed

9.3.3.1. Task list

- Create Server project
- Create Web Application project
- Setup Private Docker Registry
- Setup DNS
- Setup Continuous Deployment System

9.3.2. Continuous Deployment System overview

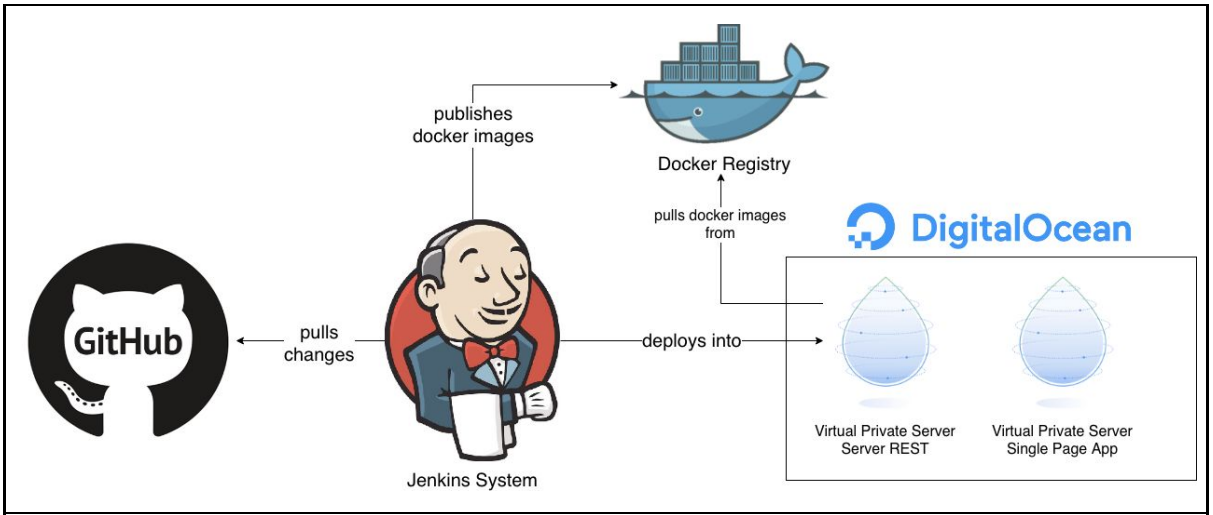


Figure 11

Components description:

- GitHub Repositories

The source control management system is hosted on GitHub. It grants access to Jenkins via a Secret Deploy Key.

- Jenkins System

Jenkins System keeps looking for changes on Git Repositories. On a change, it starts a pipeline defined on the Git Repository. Each repository is responsible to define its own pipeline flow.

- Docker Registry

The Docker Registry stores and serves Docker images created by Jenkins Pipeline.

- Droplets. Virtual Private Servers from DigitalOcean

Using Virtual Private Servers from DigitalOcean Cloud provider, also called Droplets, to deploy the different Application components running inside Docker containers.

9.3.3. Final sprint summary

The sprint's goals have been accomplished.

The effort needed to setup a Continuous Deployment system has been greater than expected since more tools were needed to make the system work.

The extra tools are:

- Terraform, Infrastructure as Code.

In order to make manage Digital Ocean resources in an automated way, I expected to have a Jenkins plugin to make so. That option was not as complete as I expected but Jenkins given the integration with Terraform as a Plugin.

- DigitalOcean Space. Distributed Object Storage.

Terraform needed a way to store the state of the infrastructure. Store that state on the Jenkins file disk was not a solution since the file system was clean between pipeline executions. In order to persist the state between pipelines executions, I setup the

backend for Terraform to persist this state in the Distributed Object Storage provided by DigitalOcean, called Spaces.

The rest of the tools used was expected and planned in advance.

9.3.3.1. Jenkins

The Jenkins system is responsible for:

- Store the sensitive Credentials
- Run the Jenkins pipelines

The Jenkins pipelines are defined on each repository but they share a common structure:

1. Checkout Git Repository
2. Build and Test
    a. Install NPM dependency
    b. Run Unit and Integration tests
3. Build and Publish Docker images
    a. Build Docker image from Dockerfile
    b. Publish Docker image into Nexus Repository
4. Deploy into production

In order to deploy into production, I used Terraform[4]. Terraform is a tool to manage infrastructure as Code. Each repository defines the infrastructure it needs and Jenkins interacts with DigitalOcean API to manage the infrastructure.

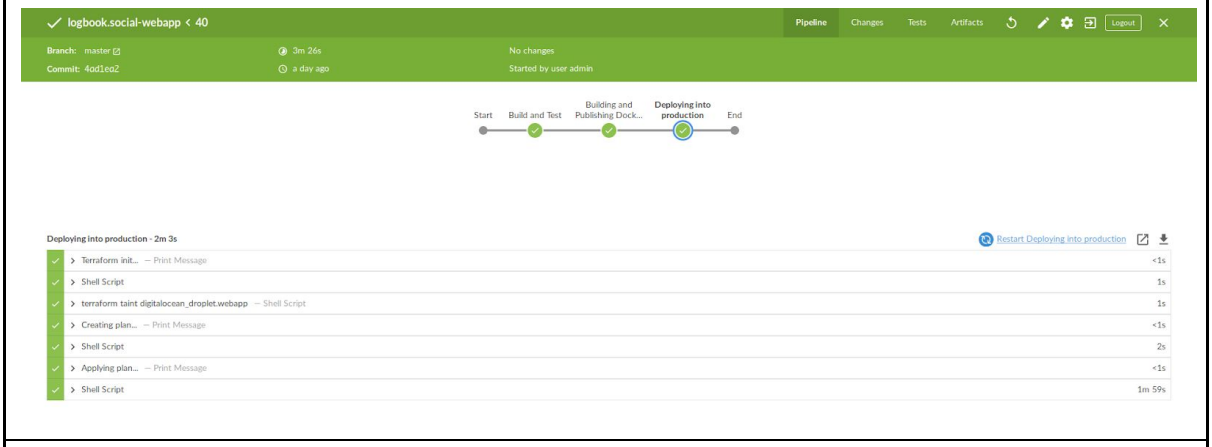


Figure 12
Logbook.social-webapp Jenkins pipeline

[4] https://www.terraform.io/

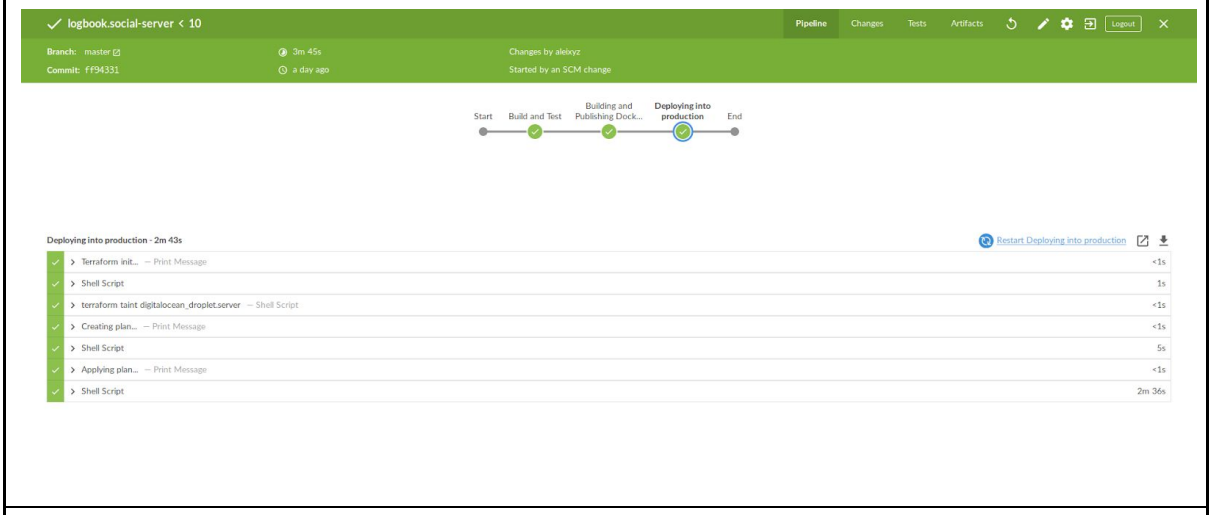


Figure 13
Logbook.social-server Jenkins pipeline

9.3.3.2. Docker Registry using Nexus Repository

Nexus Component exposes an authenticated API for the Docker to connect with. It ensures the Docker client to be authenticated in order to push or pull Docker images by the Jenkins system or the DigitalOcean Droplets.

Nexus keeps track of all Docker images made by the system, helping on rolling back some Docker image contains a bug and we need a fast rollback to make the system functional again in case of emergency.

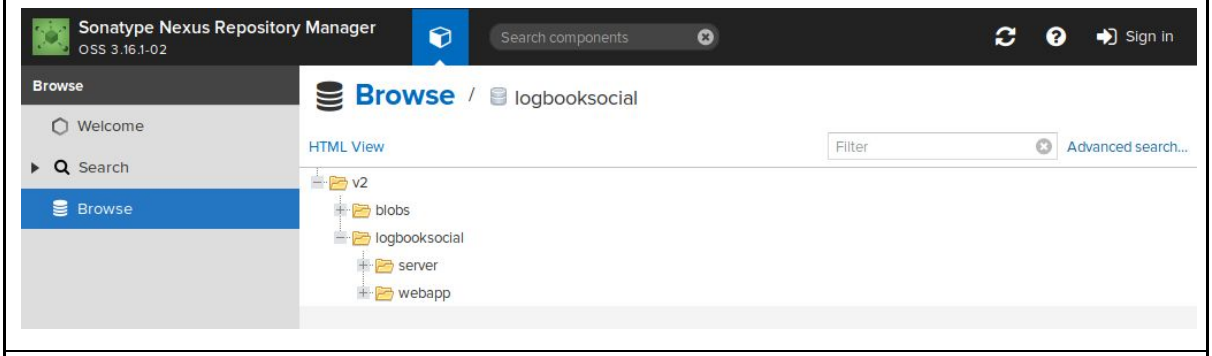


Figure 14
Nexus Repository - Docker Registry

9.3.3.3. Continuous Deployment cloud data

A visible implication of a Continuous Deployment system is the amount of instances created/deleted on the Cloud Provider. We see that the amount of instances related to webapp and server are considerable but aren't related to deliver value to the user. They are related to the amount of trial and error to setup the system correctly.

The products are used by:

- Standard Droplet - 1GB: webapp and server.
- Standard Droplet - 2GB: Jenkins and Nexus.
- Spaces: Terraform backend to store the infrastructure state.

Product	**Collective Usage**	**Price**
Standard Droplet - 1 GB / 1 vCPU / 25 GB SSD @ $5.00/mo ($0.00744/hr)	75 instances - 138.43 hrs	$1.03
Standard Droplet - 2 GB / 1 vCPU / 50 GB SSD @ $10.00/mo ($0.01488/hr)	34 instances - 138.43 hrs	$2.06
Spaces ($5/mo 250GB storage & 1TB bandwidth)	27 hours	$0.20

Even though there are more than a hundred instances the amount of resources active at the same time are:

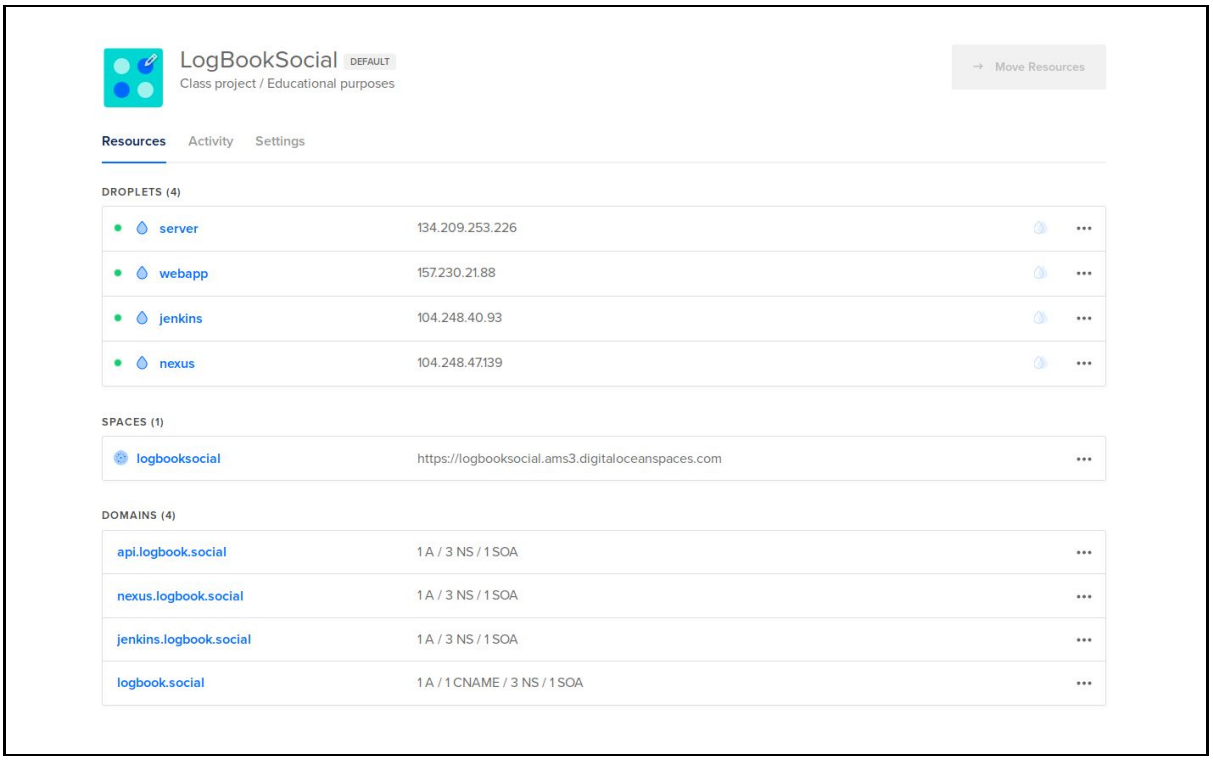


Figure 15

9.3.3.4. Infrastructure Overview

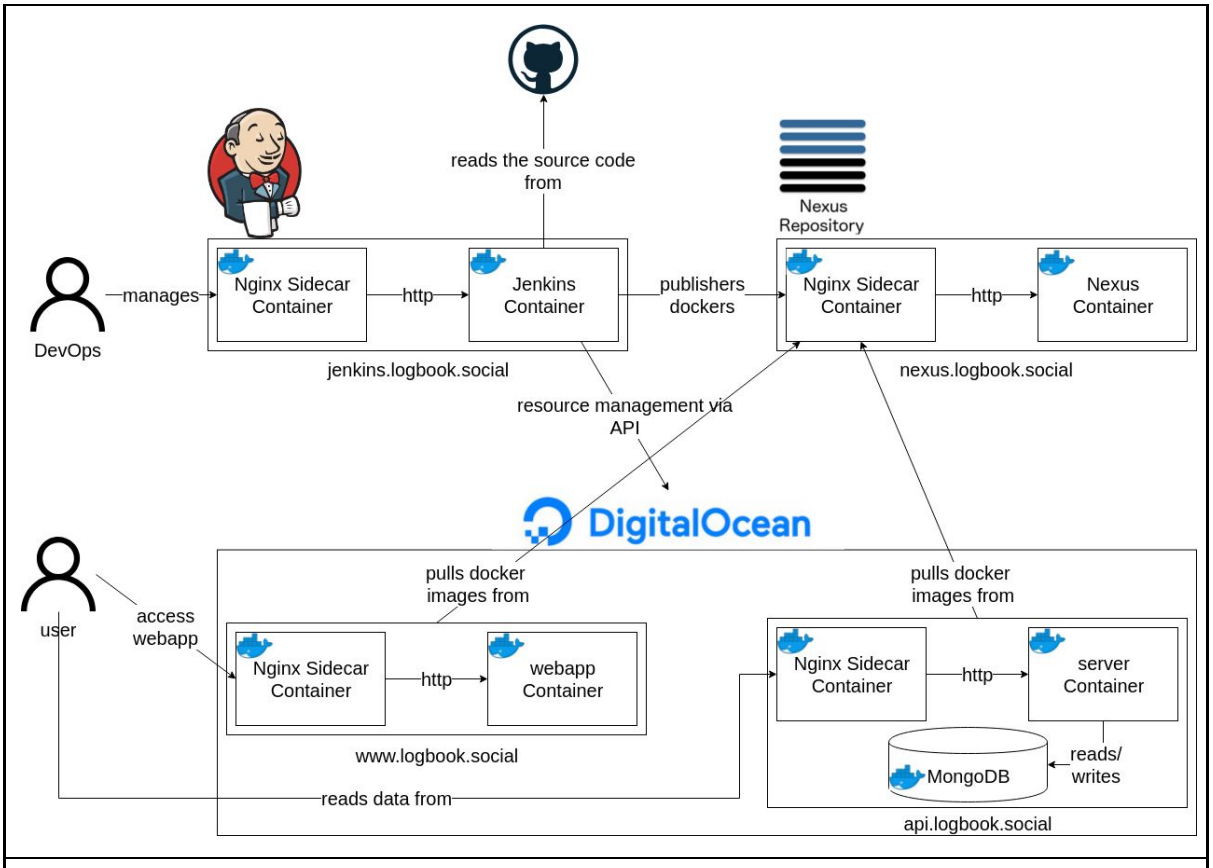


Figure 16

Each component is deployed as multiple Docker containers:

- Jenkins Component is composed by:
  - Nginx Container as Sidecar Container exposing the HTTP and HTTPS protocols
  - Jenkins Container
- Nexus Component is composed by:
  - Nginx Container as Sidecar Container exposing the HTTP and HTTPS protocols
  - Nexus Container
- WebApp (Single Page Application) Component is composed by:
  - Nginx Container as Sidecar Container exposing the HTTP and HTTPS protocols

    - Nginx Container as Static File Web Server
- REST Server Container is composed by:
    - Nginx Container as Sidecar Container exposing the HTTP and HTTPS protocols
    - Server Container as REST Server
    - MongoDB Container as Main Database

The Sidecar Containers added the SSL security for the Components. As Nginx Containers, the Sidecar Containers exposes the secure HTTPS protocol to be accessed via the Internet, forwarding the traffic into the local network, over unsecure protocol HTTP, with the main component. Note that the only way to access the main component is via the Sidecar Container, doing so we ensure the user access the component via HTTPS and a valid SSL certificate.

The SSL Certificate is generated with Let's Encrypt[5] Certificate Authority. Let's Encrypt is a free, automated, and open Certificate Authority. The tool used to automate the SSL certificate is Certbot[6], it is included on the pipeline and the server to renew the Certificate when it is about to expire.

9.3.3.5. Drawbacks

The Continuous Deployment system decision, replacing the instance and change the DNS to point into the new instance instead of using a *Load Balancer*[7], implied that the webapp might not be accessible/functional for the user for some minutes because of DNS resolution.

In case the DNS resolution becomes a real problem, we might consider using the DigitalOcean Load Balancer.

9.3.4. Jenkins File of Server

```
pipeline {
      agent any

      triggers {
      pollSCM('H/5 * * * *')
      }
```

[5] https://letsencrypt.org
[6] https://certbot.eff.org/
[7] https://www.digitalocean.com/products/load-balancer/

```
      environment {
        TF_VAR_FINGERPRINT                  = credentials('FINGERPRINT')
        TERRAFORMVARS                       =
credentials('terraform.tfvars')
            }
      stages {
      stage('Build and Test') {
            agent {
                    docker {
                  image 'node:lts'
                    }
            }
            steps {
                  echo 'Building....'
                  sh 'npm install'
                  sh 'npm run build'
                  echo 'Testing....'
                  sh 'npm run test'
            }
      }
      stage('Building and Publishing Docker Image') {
            steps {
                  script {
                  echo 'Building Docker Image....'
                  docker.withRegistry("https://nexus.logbook.social",
'nexus-credentials') {
                        def customImage =
docker.build("logbooksocial/server:${env.BUILD_ID}")
                        echo 'Publishing Docker Image....'
                        customImage.push()
                  }
                  }
            }
      }
      stage('Deploying into production') {
            steps {
                  echo 'Terraform init...'
                  dir('cloud') {
                  sh "terraform init -backend-config=${env.BACKEND}"
                  sh "terraform taint digitalocean_droplet.server"
                  echo 'Creating plan...'
                  sh """
                        terraform plan \
                        -input=false \
```

```
                        -var="ssh_fingerprint=${env.TF_VAR_FINGERPRINT}"
\
                        -var="pub_key=/root/.ssh/id_rsa.pub" \
                        -var="pvt_key=/root/.ssh/id_rsa" \
                        -var="version=${env.BUILD_ID}" \
                        -var-file=${env.TERRAFORMVARS}
                    """
                    echo 'Applying plan...'
                    sh """
                        terraform apply \
                        -input=false \
                        -auto-approve \
                        -var="ssh_fingerprint=${env.TF_VAR_FINGERPRINT}"
\
                        -var="pub_key=/root/.ssh/id_rsa.pub" \
                        -var="pvt_key=/root/.ssh/id_rsa" \
                        -var="version=${env.BUILD_ID}" \
                        -var-file=${env.TERRAFORMVARS}
                    """
                    }
                }
            }
        }
}
```

9.4. Sprint 1 - a minimum viable wallet

9.4.1. Sprint Planning

nem2-wallet-browserextension first features were prioritized to fulfill this project requirements. The features and tasks I made are:

- Start the project base using Vue Technology

The project skeleton had to help to create independent components. In that way, multiple people can work on independent pieces of features without depending on others. This approach helped the collaboration of other developers.

- Handle URI Schema calls from Browser

The integration between a web application and the browser extension. The feature required be able to start the Wallet on a specific page and information received from the website.

- Webhooks for URI Schema[8]

The first implementation had not the capacity to perform webhooks as URI Schema requested.

- Basic Transfer Transaction feature

Be able to sign and send Transfer Transactions into the Blockchain Network

- Basic Account Information display
- Basic Account management

Be able to create new accounts and switch between them.

9.4.2. Components diagram

In order to create the Wallet, I used VueJS as a Web Application framework. Vue is designed to create Components to encapsulate behaviour and re-use.

Those components are organized into a "tree" of nested components. I would differentiate between Feature Components and reusable components, because otherwise we could have a graph of components instead of a tree.

A Feature Components contains a more complete behaviour and it is more difficult to be reusable cross components. An example is a Page with the behaviour of sending a Transfer Transaction.

On the other hand, a reusable component has a finite behaviour and style. It serves to encapsulate logic that it is duplicated between components. An example can be a form field.

At the end of the iteration, the feature components looked like:

[8] https://github.com/nemfoundation/nem2-wallet-browserextension/pull/102

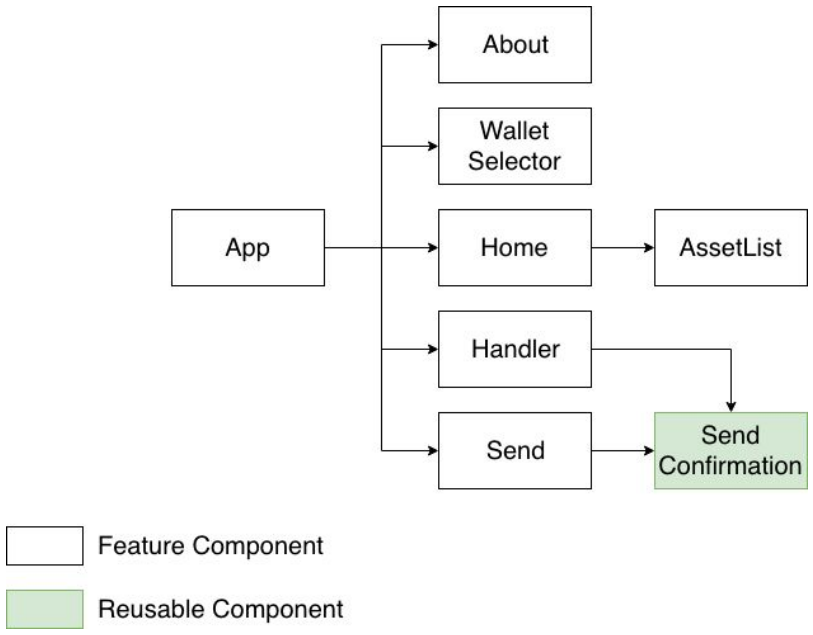


Figure 17

- App Component

It controls the application state and the user flow. It contains the lateral menu and the top bar to help the user navigate between feature components.

- About Component

It shows the information about the Wallet, the open source license and how to collaborate in the development.

- Wallet Selector Component

It manages the Blockchain account and the nodes. It helps the user to fast change between different Accounts.

- Home Component

It shares the Account information, which is the Account PublicKey, Address and a list of the latest transactions.

- Asset List Component

It shows to the user the amount of assets he/she owns.

- Handler Component

This component is the entry level for the wallet-browserextension when the Firefox Browser handles a NEM-URI. It understands NEM-URI Schema and displays the important information to announce the transaction into the Blockchain.

- Send Component

It has the logic of sending digital assets and messages.

- Send Confirmation Component

It is a reusable component. The Send Confirmation Component has the logic of signing and announcing the transactions. It has to handle all the Blockchain transaction types, not only the Transfer Transactions. This component is not only used by Handler and Send Component but also for all other Components that have to send transactions into the network.

I would like to notice that the diagram just includes the components I have worked on. As of today, the Wallet had multiple contributions and the amount of different components is more than shared on the diagram.

9.4.3. Wallet UI

The UI was based on Material Design[9] with Vuetifyjs[10] as UI Framework for Vue. At the end of the Sprint, the Wallet looked like:

[9] https://material.io/design/
[10] https://vuetifyjs.com/

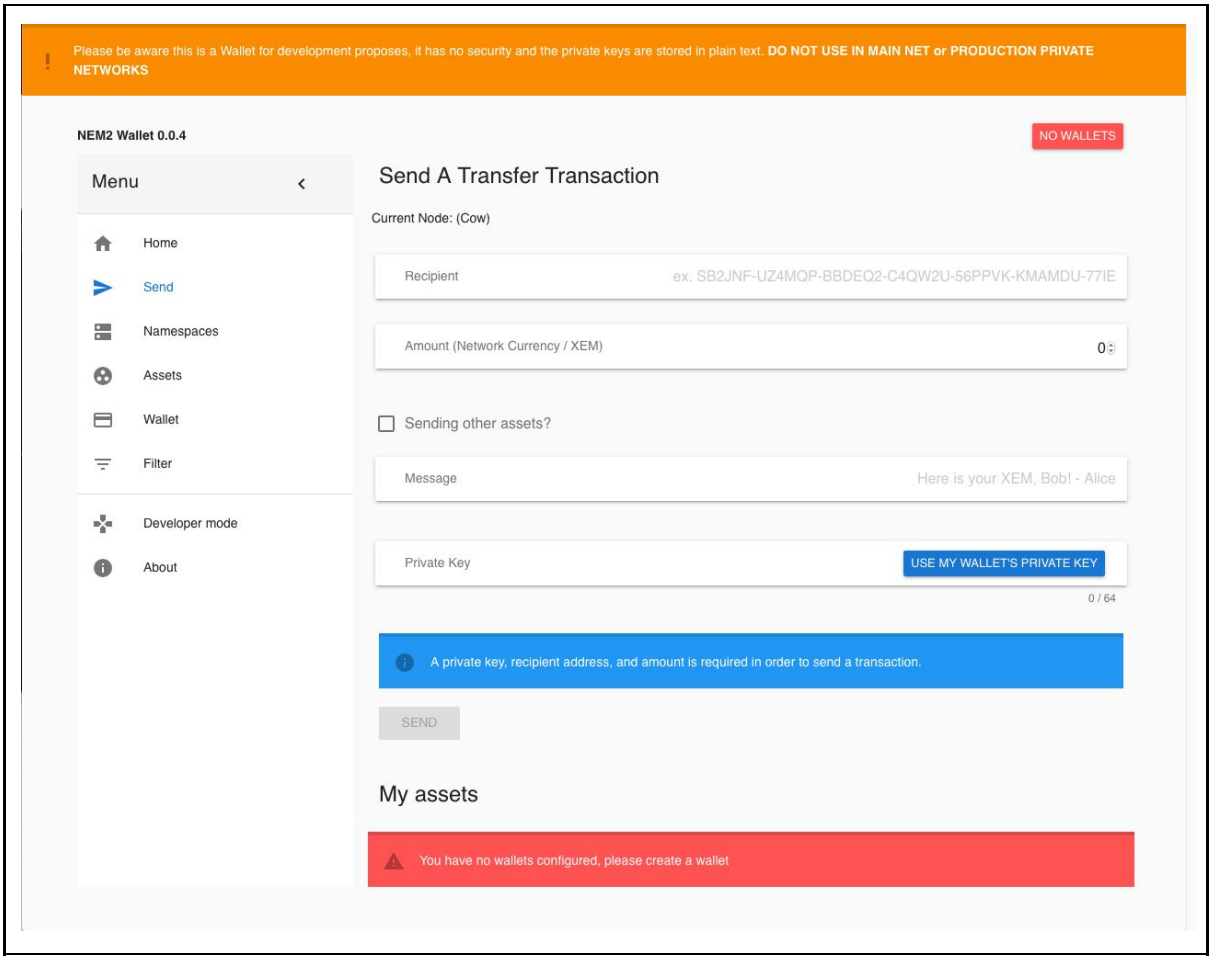


Figure 18. Wallet UI

Note that wallet looks completely different but the idea is to share the state on that point instead of the current to be more close to the actual work I did related to the Wallet.

9.4.3. Sprint retrospective

The sprint 1 had no previous analysis nor design until it was started. Because of that, the quality was not as good as expected compared to other planned components.

At the end of the Sprint the Wallet had all the required features to continue with the main purpose of the project. On the other hand, the wallet captured the developers attention and kept growing independently of this project.

I consider it a successful result of the sprint.

9.5. Sprint 2 - Content Creation

9.5.1. Sprint Planning

The value proposition for Sprint 1 is the Content Creation focusing on the *Publisher* stakeholder.

The Features selected for Sprint 2 are:

- Publish a Post
- Post Detail
- List Posts
- Announcing a Transaction

The reason why there are four features for the first sprint that delivers value to the end user is because it has integration with the Wallet and the Blockchain to accomplish those features. It is expected to need to support other components to be able to close the user journey to publish a Post on the platform.

9.5.2. Approaches to create content

This section aims to explain the different approaches on creating content from how to connect the different components to accomplish the feature. In order to know how the data is stored in the Blockchain, see section 9.8. Command Sourcing Implementation.

Option 1) Subscribe to a specific Address

NEM2 API supports WebSockets. WebSockets are used to push notifications to Subscribers about changes on:

- New Block added into the Blockchain
- Confirmed Transaction for a specific Address
- Unconfirmed Transaction for specific Address
- Unconfirmed Transaction removed for specific Address
- Status Transaction for specific Address

With this subscriptions we are able to track all the behaviour that users perform on the platform.

Figure 19 shows how the interactions between the components are being made with this approach. The user flow is:

1. Starting at WebApp, the *Content Creator* fills the Post Form shown in figure 19. When he/she finishes checking the content, the WebApp asks to Sign and Announce the Transaction via a Wallet to continue. If so, then:
2. NEM2 Wallet displays the important information about the Transaction, mainly the network cost and what will be sent to the network, like message and assets. Then the *Content Creator* can accept the transaction and it will be announced to the NEM2 Blockchain.
3. NEM2 Blockchain receives the Transaction and validates that the user can really platform the Transaction. Two scenarios:
    a. It is not valid. NEM2 notifies that the Transaction caused an error.
    b. It is valid. NEM2 notifies that it is in an unconfirmed state and eventually will be included in a Block. When that happens, it notifies that the transaction is confirmed into the Block.
4. Server receives a notification from NEM2 Blockchain. Two scenarios:
    a. A transaction is not valid. It does nothing.
    b. A transaction is valid and included in a Block. It performs the LogBook.Social validations and include it in the MongoDB.

More information about the nomenclature related to Blockchain can be found at Annex A.

Issues and Drawbacks

1. Creation flow does not involve the Server.

It is not possible to have a Correlation ID to help the system track the Post creation. The WebApp and the Server are disconnected from each other in the *Content Creation flow*.

2. Unable to notify the WebApp when the Post is created.

The system is unable to know which is the user that has to notify that created the content because of this disconnection between the WebApp and the Server. The user experience about creation is poor since we cannot tell what really happen. The user will find that the Post appears on the WebApp after sometime or it never appears.

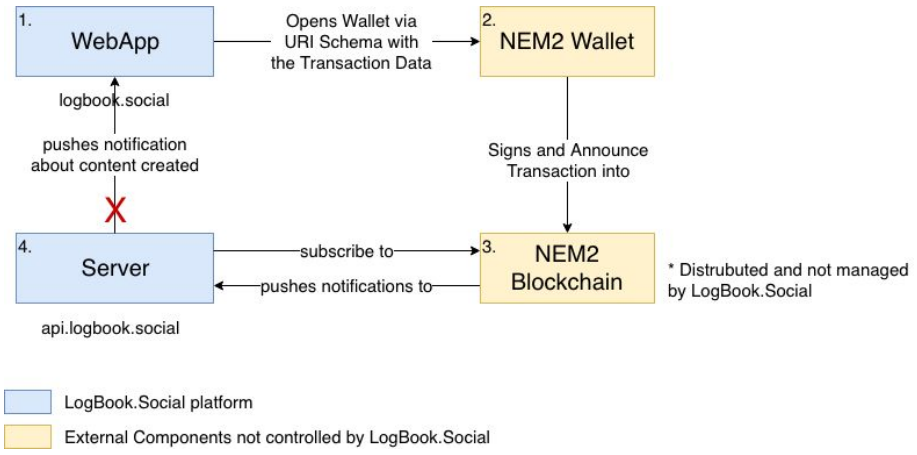


Figure 19

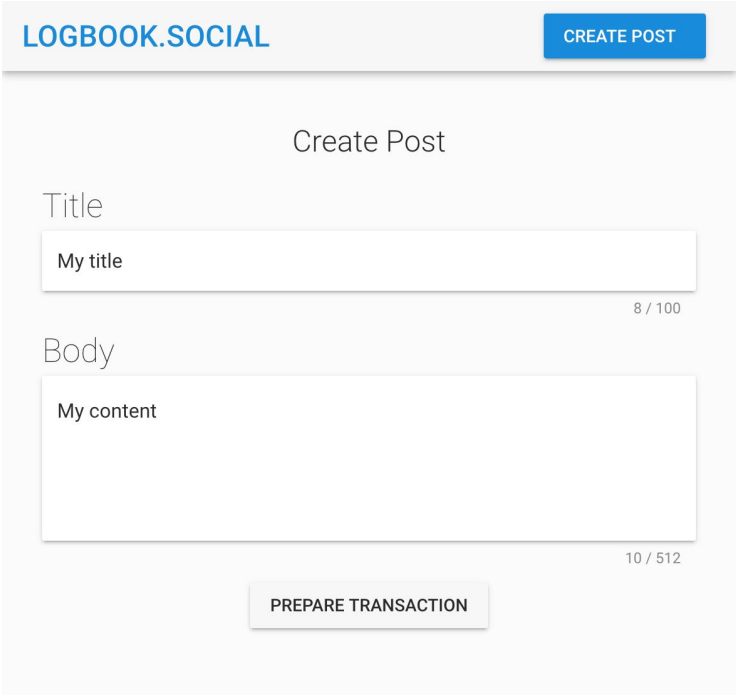


Figure 20

Option 2) Webhooks

Option 2 is similar to Option 1 but adding a Correlation ID to solve the issues and drawbacks. In order to so, NEM2 wallet offers a Webhook support to notify a specific service about the Transaction Hash related to the Correlation ID.

The user flow is:

1. Starting at WebApp, the *Content Creator* fills the Post Form shown in the figure 21. When he/she clicks *Prepare Transaction* button, the WebApp asks the server for a Correlation ID for the *Post Creation* action.
2. When the correlation ID is received and he/she finishes checking the content, the WebApp asks to Sign and Announce the Transaction via a Wallet to continue. If so, then:
3. NEM2 Wallet displays the important information about the Transaction, mainly the network cost and what will be sent to the network, like message and assets. Then the *Content Creator* can accept the transaction and it will be announced to the NEM2 Blockchain. When the user accepts announcing the transaction, then:
    a. Transaction is sent to the NEM2 Blockchain network
    b. The Webhook with the Correlation ID and the Transaction Hash is send to the Server.
4. NEM2 Blockchain receives the Transaction and validates that the user can really platform the Transaction. Two scenarios:
    a. It is not valid. NEM2 notifies that the Transaction caused an error.
    b. It is valid. NEM2 notifies that it is in an unconfirmed state and eventually will be included in a Block. When that happens, it notifies that the transaction is confirmed into the Block.
5. Server receives a notification from NEM2 Blockchain. Two scenarios:
    a. A transaction is not valid. The Server notifies the User via WebApp about the failure.
    b. A transaction is valid and included in a Block. It performs the LogBook.Social validations and includes the Post in the MongoDB. Then the Server notifies the User via WebApp that the Post is created and shows a link to view the Post.

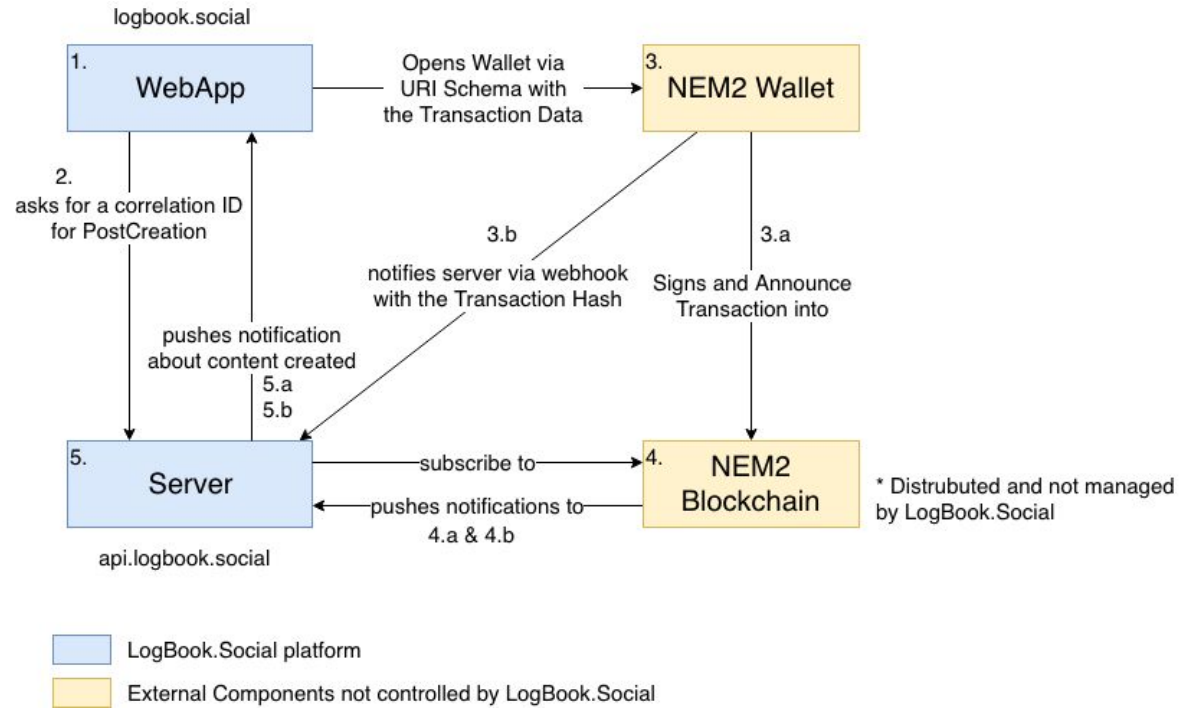


Figure 21

9.5.3. Supporting Webhooks

It was not possible to close the user journey of publishing a Post on the platform using Webhooks because the nem2-uri-schema and nem2-wallet-browserextension did not support it yet.

For nem2-uri-schema:

- Issue: https://github.com/dgarcia360/nem2-uri-scheme/issues/2

Shared the need about a WebHook. We discussed about how a Webhook might look like and which is the required data to be send to the server in order to track the Transaction.

- Pull Request: https://github.com/dgarcia360/nem2-uri-scheme/pull/3

The Pull Request had the required code to full fill the expected behaviour discussed on the Issue with the proper unit testing. It was accepted and included on the next library version.

For nem2-wallet-browserextension:

- Pull Request: https://github.com/nemfoundation/nem2-wallet-browserextension/pull/102

The Pull Request implemented a way to add Webhooks into the Sign and Announce action when the Action was triggered by the nem2-uri-schema library.

9.5.4. Delegating the development of nem2-wallet-browserextension

As shown in the section 12.1.1. Blockchain Wallet and URI Schema integrations, there was the need on creating a Wallet that could allow the creation of a Blockchain Web Application as this project is aiming to build.

The NEM development community found it interesting to contribute to and more developers joined forces to keep adding features. It implied that I had to dedicate hours on managing pull requests and community expectations. Since it was not the original objective to build a wallet to become mainstream and used for developers in the first place, I decided to delegate the development to other community members that could take the lead of the development and push the wallet forward.

At one point, NEM Foundation, one of the main entities developing NEM Blockchain and NEM Tools, got interest on the project and they took the development under their own GitHub Account.

https://github.com/nemfoundation/nem2-wallet-browserextension

From the point I transferred the repository, I adopted a position of contributing to the wallet sporadically on the features needed for this project only.

9.5.5. Sprint retrospective

The sprint overall was satisfactory.

There was a good part of research of how to make the platform viable using Blockchain for all User interactions.

The task that took longer was Publish a Post since it implied:

- Integrate the WebApp with the Wallet Browser Extension
- Create a Correlation Id to improve the User Experience

- Design how to store the information in NEM2 Blockchain. Which it let into the usage of Command Sourcing Pattern.
- Design how to restore the system state after each restart

The time invested on this feature helped on the design and implementation of the next features since it reuses how the components are integrated.

9.6. Spring 3 - Comments, Profile and Digital Assets

9.6.1. Sprint Planning

The last sprint had to accomplish the main value proposition of this project, allow the users donate digital assets to the content creators and boost the user interactions via Comments.

The features selected for the Spring 3 are:

- Logically Remove Posts
- User Profile
- Digital Assets
- Search
- Comments

The Feature Comments was not part of the Specification at the beginning of the project. The reasons why it was included instead of adding some other features that were already planned are:

- Thanks to the previous sprints experience, I was able to measure better the effort needed for each task. The remaining tasks needed more time that I had available for the Sprint.
- Comments feature was similar to previous feature Create Posts. It would be possible to apply the same approach and similar business logic for this feature.

So, in order to make the platform more complete and have more interaction types, I decided to add the Comments Feature since the balance between value-effort seem to be approachable.

9.6.1.1. Comments Acceptance Criteria

In order to evaluate the effort needed, I made the Acceptance Criteria to

```
@comments
Feature: Comments
    In order to add my opinion or just share my thoughts about a
Post
    As a viewer
    I want to add a comment to a Post

    Scenario: Some text
        Given a comment with at least one character
        When the Viewer announce the transaction
           And in the Blockchain
        Then the comment appears in the Post

    Scenario: No text
        Given an empty comment
        When the Viewer tries announce the transaction
        Then an error message appears indicating that the comment
must not be blank
```

9.6.2. Blockchain Network issues

During the week of the 2nd of June, I found that the platform was not working anymore. I did the proper platform debugging and found that was the Blockchain that was unstable. Since I am using a Beta version of the Blockchain, it was expected that it had some issues.

I found on the NEM2 Slack some other developers sharing that they were facing the same issue as myself. Because it might take a while to reset the state of the Blockchain, I decided that the most effective solution was to setup my own NEM2 Blockchain Testing Network.

Using the previous knowledge on Sprint 0 of setting up the infrastructure using Terraform as Infrastructure as Code, I created a network using as a starting point the Catapult Service Bootstrap[11].

Catapult Service Bootstrap is a set of docker-compose files that setups a NEM2 Blockchain Network for testing with just one node.

Combining the Catapult Service Bootstrap and Terraform, I was able to create an environment on my Cloud Provider, Digital Ocean.

[11] https://github.com/tech-bureau/catapult-service-bootstrap/

Further explanation can be seen at point 9.2.1. Test Network become unstable.

9.6.3. Sprint retrospective

The third Spring, even though it had the incident with the Blockchain Testing Network. I was able to deliver all the planned feature thanks to the previous knowledge acquired.

I was able to predict accurately the time needed for each feature and react faster to the different incidents that I had.

9.7. Storing information in Blockchain Research

Command Sourcing has been the backbone of the project. At the beginning of the project I had a blurry idea of how to use Blockchain combined with Event Sourcing Pattern, that's why there was a research during the Sprint 2 about the different options and their tradeoffs.

As part of the Analysis Phase a research about how to store properly Domain Models on Blockchain was made.

The Analysis approached two ways to approach the modeling:

- Event Sourcing
- Command Sourcing

Event Sourcing is described by Martin Fowler as:

> "Event Sourcing ensures that all changes to application state are stored as a sequence of events. Not just can we query these events, we can also use the event log to reconstruct past states, and as a foundation to automatically adjust the state to cope with retroactive changes."
>
> - Martin Fowler[12]

So, Event Sourcing helps in recording the system mutations and be able to reproduce the state of the system using those Events recorded on the Event Log, also known as Event Store.

[12] https://martinfowler.com/eaaDev/EventSourcing.html

On the other side, Command Sourcing is about recording the user intentions before the system mutate the state.

An *Event*, in contrast to *Command*, records a Fact that happened in the system. One of the main differences between a Command and an Event is that a *Command* might fail but an *Event* cannot.

9.7.1. Command Store and Event Store

A Command Store and Event Store can use the same technology, in our case Blockchain, but the main difference where they are involved in the system behaviour.

A Command Store saves the commands before the Application Logic is applied meanwhile the Event Store records the facts or state changes after applying the Application Logic as can be seen in Figure 22.

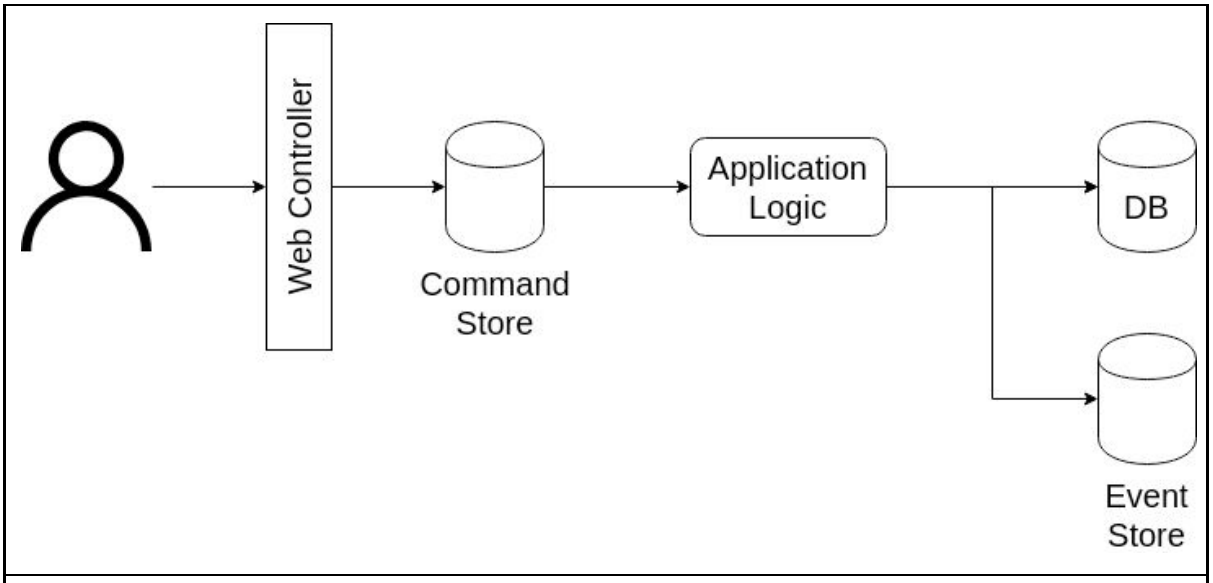


Figure 22

Because we use Blockchain, we have different component flows depending on which approach we use.

9.7.2. Command Flow

When using the Command Sourcing Approach, we need the user to Store the Commands before we apply our Application Logic.

1. A user interacts with the Web Application

2. The Web Application opens the Wallet with a Transaction data to be signed
3. User Signs the Transaction with its Wallet
4. User Announce the Transaction into the Blockchain
5. The System updates the state based on the new Command stored on the Blockchain

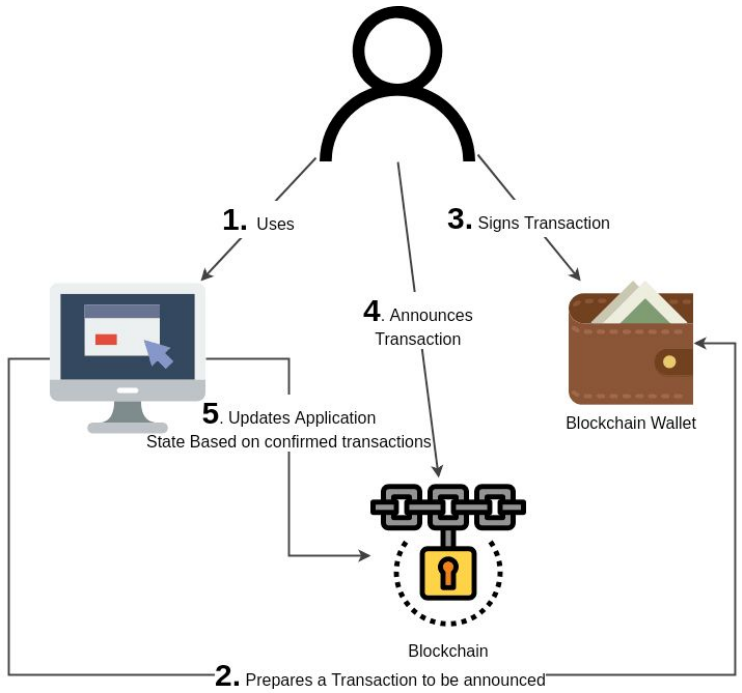


Figure 23

9.7.3. Event Flow

When using the Event Sourcing Approach, we need the Application to Store the Events after we apply our Application Logic.

1. A user interacts with the Web Application
2. The Web Application opens the Wallet with a Transaction data to be signed
3. User Signs the Transaction with its Wallet
4. **User Announce the Transaction into the Application**
5. The System updates the state based on the new Command
6. Captures the state mutation and stores it in the Blockchain

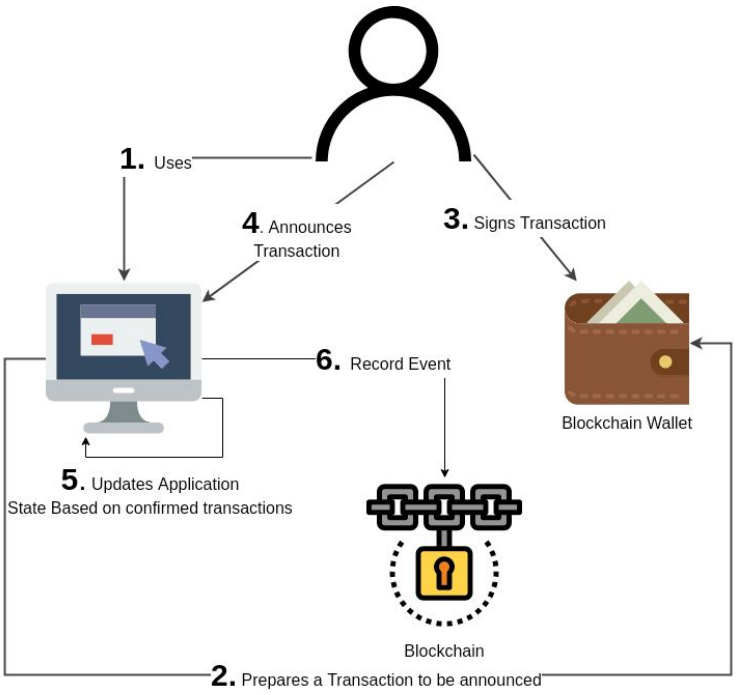


Figure 24

9.7.4. Command Sourcing vs Event Sourcing Approaches

The main difference between Command Sourcing and Event Sourcing is who writes into the Blockchain. In case of Command approach, who writes into the Blockchain is the User, meanwhile in case of Event approach, who writes into the Blockchain is the Application.

The reason for storing *Commands* in Blockchain is because we want Blockchain to be the source of truth regardless of the application. Storing the *User Intentions* we are able to:

1. Recreate the state from scratch.
2. Force the Blockchain to be the source of truth instead of the Application*
3. Empower the User allowing him/her to prove that application did something it should not do.
4. User sends signed transactions directly to the Blockchain network

9.7.5. Drawbacks

- The logic still resides in the Application.

As a Company, we want to own the code that gives value to the user in order to get revenue out of the investment, in case we would like to share how the system behaves, we could release a library that given an Account to apply the business logic.

It might happen that an Application might ignore a Command in purpose, when that happens, we can share the proof of a Transaction that contains a Command was not processed by the system.

- The dependency of Wallet integration.
- There is a lot of mapping from a Transaction to a Command.
- Infrastructure edge cases manual handling
- Pull-based (reading from the account history) instead of push-based (web sockets for example)

9.8. Command Sourcing Implementation

A command as a user intention that happens in the system, it looks like *RegisterUser*, *CreatePost*, *AddCommentIntoPost*, *DonateAssetsToPost*, ...

We do not store the *"Entity"*, instead, recording the user intentions as Commands we can recreate the *Entity* from those.

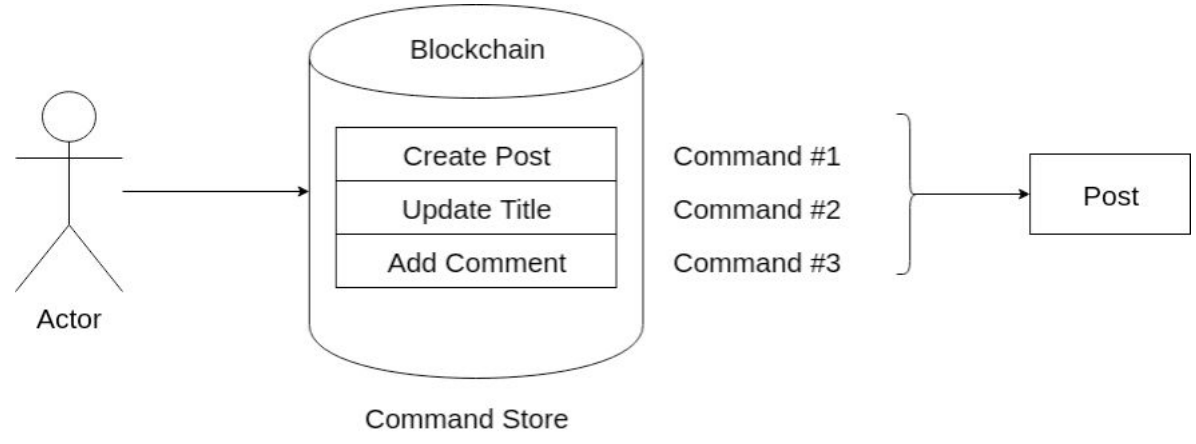


Figure 25

Post, as a Domain Model, is the result of applying in order the Commands.

9.8.1. How a Command look like

A Command is a Transaction with a JSON message to a specific address.

Field	**Type**	**Used as**
Signer	Blockchain Address	Author
Recipient	Blockchain Address	Unique Entity Id
Block Included	Integer	Command Order
Transaction Hash	Hash	Command Id
Message payload	Raw text	Command Data

Example of a Transaction that behaves as a Command:

Field	**Value**
Signer	SDC6YL-BMPDR2-S3PFEI-F4TG5D-OTHJEC-MZZF66-UEEA
Recipient	SAQXPA-3M3F3J-WZU4Y5-OFNI4X-CKZZU7-I6J4EU-TABI
Block included	1212
Transaction Hash	B203F2B6AAD27599D9FFCECDB07F75A79E514784545CBFC609F...
Message payload	{ "_type": "CreatePost", "title": "Blockchain is cool", "body": "xyz" }

Then, using those fields we can transform them

9.8.2. Receiving the Commands from Blockchain

The process of receiving the commands are simple:

1. Fetch the transactions given an Address/Account
2. Transform/Map those transactions into valid Commands

On the step 2. we might need to discard invalid commands, it's possible due to multiple-accounts can send transactions to the Account and those might not follow the *Command format*.

Psecudo-code:

```
interface CommandFetcher {
      givenAnAddress(address: Address): List<Command>;
}

class CommandFetcherHttp implements CommandFetcher {
      constructor(private readonly accountHttp: AccountHttp) {}

      givenAnAddress(address: Address): List<Command> {
      const transactions = this.accountHttp.transactionsOf(address);
      return transactionToCommandMapper(transactions);
      }
}

// Blockchain dependent, just used as signature clarification
interface AccountHttp {
      transactionsOf(address: Address): List<Transaction>;
}

function transactionToCommandMapper(transactions: List<Transaction>):
List<Command> {
      // Implementation
}
```

9.8.3. Applying the commands to an entity

When we have a list of *Commands*, we want to apply them into an *Entity*. A common way to deal with that process is using a *Service*. The service has the responsibility to create/update/delete entities depending on the command.

Psecudo-code:

```
const postService = new PostService();
const commands =
commandFetcher.givenAnAddress(Address.of("SXXXPA-...-I6J4EU-TAAA"));

commands.forEach(command => {
```

```
    if (command.type == "CreatePost") {
        postService.createPost(command.id, command.title, command.body,
command.user);
    }
    else if (command.type == "AddComment") {
        postService.addComment(command.id, command.postId,
command.comment, command.user);
    }
    // Other commands
});

interface PostService {
    createPost(id, title, body, creator);
    addComment(id, postId, comment, creator);
}
```

9.9. Exposed RESTful API

The API is principally made to consume data because of how the Blockchain Applications is meant to behave.

The API follows the RESTful[13] architectural style. The reason is because it is meant to be consumed by a WebApplication, providing the interoperability between both components.

9.9.1. Models

The data returned by the API has the following properties:

```
type Post = {
  title: string,
  body: string,
  author: string,

  donations: Donation[],
  amountDonations: number,

  comments: Comment[],

  transactionInfo: TransactionInfo,
}
```

[13] https://en.wikipedia.org/wiki/Representational_state_transfer

```
type User = {
  address: string,
  posts: Post[],

  amountDonated: number,
  amountDonationsReceived: number,
  donations: Donation[],

  comments: Comment[],
}

type Comment = {
  author: string,
  message: string,

  transactionInfo: TransactionInfo
}

type Donation = {
  postId: string,
  amount: number,
  donator: string,
  receiver: string,

  transactionInfo: TransactionInfo
}

type TransactionInfo = {
  block: number,
  hash: string,
}
```

9.9.2. Post Resources

9.9.2.1. GET /post

HTTP status code	**Response**
200 OK	[Post]

9.9.2.2. GET /post/:postId

- Parameter postId: post transaction hash

HTTP status code	Response
200 OK	Post
404 Not Found	

9.9.3. User Resources

9.9.3.1. GET /user/:userId

- Parameter userId: address

HTTP status code	Response
200 OK	User
404 Not Found	

9.9.4. Search Resources

9.9.4.1. GET /search

HTTP status code	Response

200 OK	{
	users: [User],
	posts: [Post]
	}

9.9.5. Webhook Endpoint

9.9.5.1 POST /webhook/:correlationId

- Parameter correlationId: UUID.

HTTP status code	Body	Response

204 No Content	{
	"action": String,
	"Data": {
	"hash": Transaction Hash,	

	"signer" Signer Public Key } }	
404 Not Found		

9.9.6. WebSockets

Emitter	**Event Name**	**Purpose**
WebApp	createPostRequest	Track User intention to Create a Post
WebApp	deletePostRequest	Track User intention to Delete a Post
WebApp	donatePostRequest	Track User intention to Donate to a Post
WebApp	commentPostRequest	Track User intention to Comment to a Post
Server	createPostCorrelationId	Give a Correlation Id related to User Intention to Create a Post
Server	deletePostCorrelationId	Give a Correlation Id User intention to Delete a Post
Server	donatePostCorrelationId	Give a Correlation Id User intention to Donate to a Post
Server	commentPostCorrelationId	Give a Correlation Id User intention to Comment to a Post
Server	websocketArrived	Notify the User that the system has received the Transaction by the Wallet and it starts to monitor the Command
Server	postCreated	Notify the User that the Post has been created
Server	postDeleted	Notify the User that the Post has been deleted
Server	donationMade	Notify the User that the Donation has been made
Server	commentMade	Notify the User that the Comment has been made

9.10. MongoDB Collections and Indexes

9.10.1. Generic Data Types

Donation Schema:

Field	**Type**
postId	String
amount	Number
donator	String
receiver	String
transactionInfo	TransactionInfo

Comment Schema:

Field	**Type**
author	String
message	String
transactionInfo	TransactionInfo

TransactionInfo Schema:

Field	**Type**
hash	String
block	Number

9.10.1. Posts Collection

Schema:

Field	**Type**	**Index Type**

title	String	Text index
body	String	Text index
author	String	-
donation	List<Donation>	-
amountDonations	Number	-
comments	List<Comment>	-
transactionInfo	TransactionInfo	transactionInfo.hash is unique index

9.10.2. User Collection

Field	**Type**	**Index Type**
address	String	Unique index
posts	Reference to Post	-
amountDonated	Number	
amountDonationsReceived	Number	
donations	List<Donation>	
comments	List<Comments>	

9.10.3. Intentions Collections

The next Collections share the same Schema:

- Comment Post Intentions
- Create Post Intentions
- Delete Post Intentions
- Donate Post Intentions

Schema:

Field	**Type**	**Description**

correlationId	String	CorrelationId send to the user
clientId	String	Socket Id to reply to a specific user

10. Platform Analysis, from a technical perspective

I will analyze the project from the technical perspective. As a project in the speciality of Software Engineering, I will analyze each part of the project separately.

10.1. Technical Debt

An important aspect to analyse when finishing the project is the amount of technical debt[14] introduced in the project.

The technical debt is not bad by itself as long as we are aware of it, we control it and reduce it when it is about to become a problem.

In fact, technical debt is a tool we can use when necessary to increase features in less time, learn about a domain or a tool and similar in exchange of code quality that will be paid in the future.

In this project, I introduced more technical debt on the WebApp component more than others, but those others still have small technical debt that are worth sharing.

10.1.1. Continuous Deployment System

The technical debt on the continuous deployment is low. The next improvement on this system is to reduce the downtime to 0. Currently, there is a downtime, between the old service version being replaced with the new one, around 2 and 5 minutes.

More information explained at Annex B: Difficulties about Continuous Deployment on DigitalOcean.

10.1.2. Server Application

The Server Application has low technical debt.

The technical debt is localized in:

- PostService has some code duplication
- No Testing for WebSockets because of the complexity of emulating a full component behaviour.

[14] https://en.wikipedia.org/wiki/Technical_debt

10.1.3. Web Application

The Web Application is the Component with higher Technical Debt compared with the others, but it is localised and known.

The technical debt is:

- Few tests: The WebApp does not have enough testing compared to the Server Application. The reason is that it depends a lot about the user interaction and it does not contain specific logic that can be tested in isolation.
- Few reusable components: There are some Components that have similar behaviour and look and feel. Those Components can remove duplicated code and encapsulate it into reusable components, making the WebApp more stable.

That technical debt is relevant on this component since, in order to make the project grow, it might become a problem to maintain the code base and might cause the component to break because of the lack of enough good test suite.

On the other side, thanks to introduce this technical debt, the project had all required features planned at the end of each sprint.

10.2. Non-functional Requirements Completeness

10.2.1. Security

The Platform is designed in a way that it delegates all the Private Key management into the Wallet. Because of that design decision, there is no need to manage private keys.

10.2.2. Transparency

On the WebApp, each content created by the users have links to the Blockchain Explorer and Blockchain API.

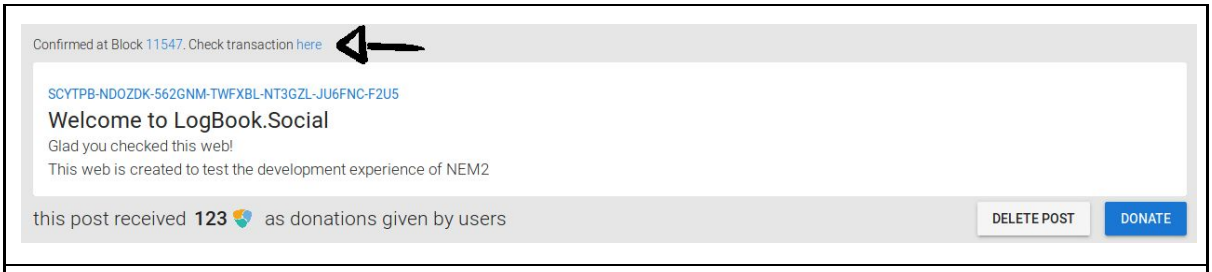


Figure 26

Unfortunately, NEM2 Explorer does not support transactions links, forcing us to use the NEM2 API to show a proof about the created content.

This approach has multiple drawbacks:

- Poor readable content: API returns JSON with data encoded on non-human readable content. It can not really help the user to determine the validity of the content.
- An API does not give a good trust since the URL might not be a known domain or an IP. Causing a lack of trust.

On the other side, NEM2 Explorer does not support human readable content neither. Making this non-functional requirement hard to accomplish because of external dependency.

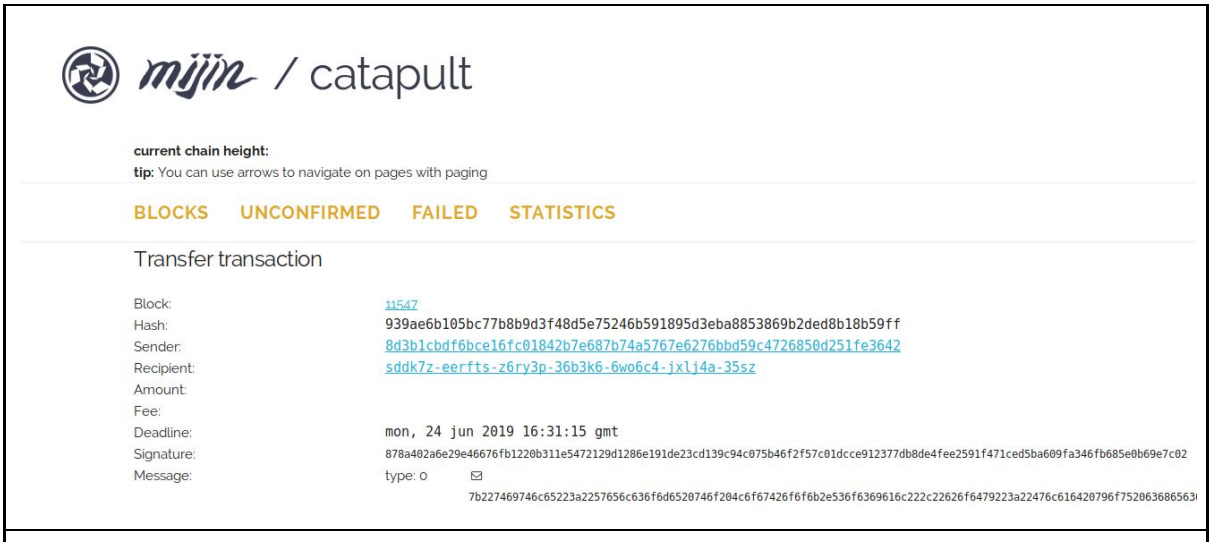


Figure 27

Figure 28

10.3. Features completeness

The Specification had a set of features that I considered necessary for the MVP before starting the development.

After the implementation and the different problems during the development, I made the decision to implement the most valuable features and let others out.

10.3.1. Not implemented features

The next features have not been implemented:

- 6.1.3. Editing
- 6.2. Spaces
    - 6.2.1. Registering
    - 6.2.2. Administration
- 6.3. Username alias

The reason, already shared and argued on the previous points, were the need to dedicate the Spring 1 to create the Wallet in order to make this project viable. I decided that it was reasonable to let those features out of scope since they had a dependency to the other features to be made.

The impact on the project is that the user type Space administrator cannot use the platform.

10.3.2. Implemented features completeness

The rest of the features were implemented and accessible via the WebSite https://logbook.social.

I consider a feature complete when the user is able to complete the all live cycle of each feature. Using the WebApp, creating content via the Wallet and displaying that content later on the WebApp again.

The best way to check the feature completeness is using the WebApp.

11. Economical dimension of the project

In this section, we will analyse the economic costs of the project, dividing them in human, material and indirect costs. The analysis will be based on the previous scope and the gantt diagram.

11.1. Budget

11.1.1. Human costs

Per Activity

Activity	Profile	Hours	€ per hour	Estimated cost
Project Planning		**73h**		**4.380€**
Context and Scope	Head of project	16h	60€/h	960€
Project Planning	Head of project	10h	60€/h	600€
Budget and Sustainability	Head of project	16h	60€/h	960€
Final document and Presentation slides	Head of project	16h	60€/h	960€
Prepare oral presentation	Head of project	10h	60€/h	600€
Oral Presentation	Head of project	5h	60€/h	300€
Analysis and Design		**80h**		**4.580€**
Requirements collecting	Analyst	20h	55€/h	1.100€
Tools analysis	Analyst	6h	55€/h	330€
User Stories	Analyst	6h	55€/h	330€
Wireframes	Analyst	6h	55€/h	330€
Acceptance Criteria	Analyst	6h	55€/h	330€
Architectural Design and Specification	Architect	30h	60€/h	1.800€
Conceptual class diagram	Architect	6h	60€/h	360€
Development Phase		**239h**		**11.235€**
Setting up the environment		**23h**		**1.215€**
Setup Git Repositories	Developer	1h	45€/h	45€
Setup Continuous Integration	DevOps	6h	55€/h	330€
Local Environment	Developer	4h	45€/h	180€
Development Environment	DevOps	12h	55€/h	660€

Content Creation		**72h**		**3.340€**
User story split into smaller tasks	Analyst	2h	55€/h	110€
Analysis, design and risk mitigation	Architect	4h	60€/h	240€
Implementation + Unit Testing	Developer	62h	45€/h	2.790€
Acceptance Criteria review	Developer	2h	45€/h	90€
Deploy to production	DevOps	2h	55€/h	110€
Comments, Profile and Digital Assets		**72h**		**3.340€**
User story split into smaller tasks	Analyst	2h	55€/h	110€
Analysis, design and risk mitigation	Architect	4h	60€/h	240€
Implementation + Unit Testing	Developer	62h	45€/h	2.790€
Acceptance Criteria review	Developer	2h	45€/h	90€
Deploy to production	DevOps	2h	55€/h	110€
Space Administration		**72h**		**3.340€**
User story split into smaller tasks	Analyst	2h	55€/h	110€
Analysis, design and risk mitigation	Architect	4h	60€/h	240€
Implementation + Unit Testing	Developer	62h	45€/h	2.790€
Acceptance Criteria review	Developer	2h	45€/h	90€
Deploy to production	DevOps	2h	55€/h	110€
Test Phase	QA	**20h**	55€/h	**1.100€**
Final Phase		**40h**		**2.400€**
Memoir redaction	Head of project	25h	60€/h	1.500€
Prepare oral presentation	Head of project	10h	60€/h	600€
Oral presentation	Head of project	5h	60€/h	300€

Per role

Role	**Estimated hours**	**Cost per hour**	**Estimated cost**

Head of project	113h	60€/h	6.780€
Analyst	50h	55€/h	2.750€
Architect	48h	60€/h	2.880€
DevOps	24h	55€/h	1.320€
QA	20h	55€/h	1.100€
Developer	197h	45€/h	8.865€
Total	**452h**		**23.695€**

11.1.2. Material costs

The planned material cost is just the laptop used for development. Because of the Continuous Integration system and the development/staging environment configuration is defined on *Analysis and Design phase*, we cannot know which is the precise amount dedicated to the project, that is why we will allocate a budget for virtual machines in the Cloud for the duration of the Bachelor's thesis.

Product	**Unit Cost**	**Total Cost**	**Units**	**Lifecycle**	**Amortization**
Asus Zenbook	800€	800 €	1	5 years	800€
IntelliJ IDEA Ultimate	80€ year subscription	80 €	1	1 year	0
DigitalOcean droplet instance 2GB RAM standard instance	$10/mo $0.015/hr	52,8 €	2 instances for 3 months	Until project is finished	0
DigitalOcean droplet instance 1GB RAM standard instance	$5/mo $0.007/hr	39,6 €	3 instances for 3 months	Until project is finished	0
Total		**972,4 €**			**800 €**

* Dollar to Euro conversion rate at $1 = 0.88€.

Aside of the paid resources, the project uses software that does not need a paid license to be used and is excluded from the material cost table.

11.1.3. Indirect costs

We consider the next indirect costs:

Concept	**Price**	**Cost**	**Final price**
Electricity	0,11667 €/kWh	0,11667 €/kWh * 450h * 0,022 kWh	1,16 €
Internet connection	35€ per month	35€ per 4 months	140,00 €
Total			**141,16 €**

11.1.4. Unforeseen contingencies & deviations

We dedicate a 15% of the sum of the direct and indirect costs as a budget for contingencies. This budget aims to minimize the deviation of the estimation due to lack of information on the estimation itself.

(23.695€ + 1.018,9€ + 141.155033€) * 0.15 **= 5103.77€** for the contingencies budget

As the project has a high uncertainty because of using an unstable technology, we will add a 25% to the tasks that have more risk.

Activity	**Current budget**	**Calculation**	**Price**
Content Creation. Implementation + Unit Testing	2.790€	2.790 * 0,25	**697,50 €**
Comments, Profile and Digital Assets. Implementation + Unit Testing	2.790€	2.790 * 0,25	**697,50 €**

Total			**1.395,00 €**

11.1.5. Control management

In this project, we consider the human error as the most probable deviation. So, we do not take into account the material cost since we almost have not got uncertainty in that part.

On a weekly basis, we will check the task deviation and the human resources invested. In order to check that deviation, we will use the following indicators:

- Variance in human costs = (estimated cost – actual cost) * actual consumption in hours
- Variance in the realization of one task = (estimated cost – actual cost) * actual consumption in hours
- Variance in the realization of a sprint = (estimated cost – actual cost) * actual consumption in hours

11.1.6. Total budget

Concept	Budget
Human	23.695,00 €
Material	972,4 €
Indirect	141,16 €
Contingencies	5.103,77 €
Uncertainty	1.395,00 €
Total	**31.307,33 €**

12. Sustainability Matrix

12.1. Environmental Dimension

From the PPP point of view, the electricity cost is the laptop in which I develop the platform, the continuous integration instance and the development environment.

Devices and Machines	**Consumption**	**Hours used**	**Total consumed**
Development Laptop	0,045 KW	6,5 hours a day during weekdays from 28th February until 5th July, excluding holidays. 98 days. 637 hours	28,665 kWh
DigitalOcean Droplet 1GB - 1vCPU	0,2 KW*	3 484, 16 h	696, 832 kWh
DigitalOcean Droplet 2GB - 1vCPU	0,2 KW*	2 826, 43 h	565, 286 kWh
Total			**1 290, 783 kWh**

* I was not able to receive any information about the consumption from Digital Ocean or fingerprint. I looked at the Droplet specs and searched for an equivalent server on the web to find a guideline of the consumption. Because it was a dedicated server, I expect the consumption by Digital Ocean to be less than 0,2 kw but I will use that number as a reference.

The Digital Ocean droplets consume 547,546 kWh a month.

Note that not all hours of the Development Laptop consuming energy were dedicated on the project.

A way to reduce the energy consumption is, instead of hosting a Continuous Integration system like Jenkins, using a Continuous Integration system on demand like Travis-CI, so the system will only run when there is a required trigger instead of actively wait for changes on the source code.

On the other side, regarding Useful life, since I am using Blockchain technology, I expect this solution to **“waste” more resources compared to centralized solutions**.

There are multiple *Consensus Algorithms* for Blockchain technologies, but the most popular is *Proof of Work*[15], an algorithm that consumes a lot of energy as a requirement to solve a complex algorithm. This project, instead, uses *Proof of Stake*[16]. It is a cheaper consensus algorithm compared to *Proof of Work,* but still consumes more energy compared to a centralized system.

In order to minimize the environmental impact, I decided to not use a public network because it is composed by up to 400-500 nodes. Instead, I decided to use a private network with few nodes, 1 to 3. This should reduce drastically the consumption made by the platform. I also considered using an existing private network managed by a third party that it is used by multiple systems, so when the platform is not interacting with Blockchain, other systems can take advantage of Blockchain and its resources instead of waste computer power.

When moving the project from a Test Network with 1 node to the Main Network with more than 800 nodes, it will have an environmental impact. We cannot calculate that the impact will be 800 times more because those resources are used by multiple people and projects, and we just consume the resources when we interact with the Blockchain, so, a proper study about the new consumption has to be made.

12.2. Social Dimension

From the PPP point of view, it has been a tough experience. The project was ambitious on using a quite new and not stable technology, in comparison with more traditional technologies that have been in the market for a long while, as it is Blockchain Technology.

As a Software Engineering project, I aimed to use good practices and put into action the knowledge acquired during the carrier. The amount of time and dedication each part of the project required more than I expected in the first place. The knowledge to accomplish each step in a quality that I could feel comfortable with was not enough and I needed to do the proper research during all the project.

The project has been a continuous learning using the good practices that helped me and the project become viable on small and incremental steps.

---

[15] https://en.wikipedia.org/wiki/Proof-of-work_system

[16] https://en.wikipedia.org/wiki/Proof-of-stake

Most of the documentation related to Blockchain was not as abundant as other materials but I was able to reuse knowledge from other areas and apply it. It has been a good experience to realise that good technical and practices fundamentals helped me navigate in a project with a lot of uncertainty and problems that I had during the project.

I have to admit that deciding to collaborate on Open Source as I did with the NEM2 Wallet and find out that it was adopted by the Community has been awesome. I was totally unexpected and I felt that people really liked to collaborate on Open Source. I feel that Open Source is a software model that I do like and feel that I will continue collaborating on projects.

On the social impact of the project, I have concerns. During the development, I become more and more informed about the problems that a distributed and non-centralized systems can become.

Currently, use Blockchain is hard and there is a big user experience barrier for most of the users. Projects like this one, that starts simplifying the process of interacting with the Blockchain can help more people to use it for the wrong purpose. Having a Social Platform anonymous and non-centralized could help people to share adult content, illegal content or, a more recent problem, the fake news[17].

If companies as big as Facebook have problems to control those issues, I imagine that a platform as this one have even a bigger challenge to solve them.

Regardless of the possible bad side of a non-centralized systems, we can see beneficial points. The project helps on the content creator, in case of use the platform for the good, they can share posts and get economic benefit directly from the visitors.

12.3. Economic Dimension

From the PPP point of view, the cost estimation for the project has been higher than I would have expected on the first place because of the uncertainty the project had.

Even though the problems I had during the development, because I was using an Agile Methodology. The cost did not differ too much (after adding the material cost increase

[17] https://en.wikipedia.org/wiki/Fake_news

due to hosting my own Test Network) because we used the planned deviations and removed features because they would require more hour dedication. We choose to not add more hours to the project and instead resize the scope of the project based on the information we had at that time, the skills and abilities, and the situation at hand.

On the Useful life side, we have that the project is reducing the time between a content creator publishing and its monetization. Part of the economic improvement is being able to have a secure digital assets transferred between parties without a central authority controlling it. Thus, because of the decentralized monetary system Blockchain provides, it reduces the amount of work done by multiple entities in order to enforce that no money has been stolen, like banks and Visa.

The economic impacts are:

- Reduced time between the sending and receiving of digital assets from a viewer to a content creator (less than 1 minute)
- Hypothesis of reduced cost of central authorities trying to secure how the money is transferred between parties.

A high potential risk for the project viability is how the law applies to the project, there is no information about how the laws shared at point 7.2 Legal requirements apply to Blockchain projects.

Other risks related to the Technology is time to market for a stable version of the Blockchain. We used a technology that is still in beta phase because of its features, but we need a stable version and a Main Network to deploy the project to get the maximum potential.

13. Conclusions

In the course of the project, we have had to explore new technologies like it is Blockchain and combine it with a broad types of different technologies and practices to reach the Bachelor's thesis milestone.

We had used:

- NEM2 Blockchain as a Distributed system to secure the digital assets and take advantage of the integrated message system to use it as a Command Store
- Created a WebApp using VueJS 2 as Web Framework with TypeScript programming language
- Created a RESTful API using Nodejs, expressjs and MongoDB with TypeScript programming language
- Created a WebExtension for Firefox Browser using VueJS 2 as Web Framework with TypeScript programming language
- Used RxJS library to handle asynchronous behaviour and make the system resilient using patterns like Retry
- Socket.io to create real-time communication between the WebApp and the server
- Jest as JavaScript/TypeScript testing tool for the WebApp and Server.
- TestContainers for integration tests between Domain Models and MongoDB on the server test suit
- Used Digital Ocean as Cloud Computing platform
- Docker containers to create deployable units of our components, sidecar containers to add features like HTTPs to our custom components and as TestContainers
- Created a Continuous Deployment pipeline using Jenkins and Terraform
- Created a Custom Docker Registry with Sonatype Nexus Repository Manager

The project required a broad understanding on different technologies and learn why, when and how to combine them together.

Blockchain, as the project central value differentiator, is a technology that itself required dedication to understand each part. Cryptography, digital asset structure, network knowledge for all possible things that could go wrong on the wire, immutable data structures and more.

The resources available to learn about Blockchain on the network are limited. It implied a lot of trial and error learning.

I learned that a Distributed Systems, and those that use Blockchain, have a broad failure points that cannot be fully planned in advance how to react to them and instead, it is a continuous learning of how the system behaves on the different scenarios and apply mitigations strategies to those failures.

On the practices and Methodologies, it has been a painful but fulfilling experience since I realized that, using good practices and methodologies like Test-Driven Development, continuous deployment systems and agile methodologies helped me to create a Minimum Viable Product that works and can keep growing in the future.

The combination of the practices and methodologies implied that I had a short and fast feedback loop during the whole project development, I knew the state of the project and how it worked in every moment, it helped me to have trust on the project and have less stressful moments in comparison if I had not used them. It has been a learning experience to realize that good methodologies not just help the project from a technical point of view but it also helps on the mental health along the project.

On the side of a product, how to reason about the Business needs and using a new technology on the market, as Blockchain is, has been a total experience. Argue which decisions to make based on the business needs have been a challenge.

I had tried to not only take this project from a technical perspective but also based on the product needs. I realized that do technical decisions based on business needs are far from obvious and they come with trade-offs.

The decision of making a Wallet on Sprint 1 was based on the product needs, instead of doing a dirty solution or changing the requirements to match the technology, is important and it has been a learning experience. I forced myself to create a software that matches the user and business needs rather than create a solution that it is easier to be implemented based on the technology limitations.

13.1. Project Conclusions

The decision of using NEM2 Blockchain had its trade-offs to match the Minimum Viable Product goal.

The benefits of using NEM2 Blockchain have been its simplicity and Blockchain native features out of the box. It had all the features needed to create the MVP and a good support from the SDKs.

It also had meaningful drawbacks.

- NEM2 Blockchain is under development and it has breaking changes between versions
- No good support for Wallets and URI Schema integration
- The explorer had some issues and was not possible to link transactions to the explorer to verify their validity, making difficult to make the platform enough transparent.

Even though those drawbacks had an impact on the project, I consider that they are expected on a platform still under development and they will be resolved on a stable release of NEM2 Blockchain.

Part of the project was learning the limits of a cutting edge technology as it is Blockchain. After implementing the basic features, I could consider to not use Blockchain for all the features. The dedication needed for each feature have been a way higher that compared with more common technologies like Spring Framework, Ruby or Rails, etc.

Next time I had to use Blockchain Technology, I will consider only start integrating a small portion of the system with Blockchain, verify the business hypothesis and then slowly migrate to Blockchain the features that make sense to do so, instead of forcing from the beginning everything being distributed.

13.2. Technical Competences

- **CES1.1:** Desenvolupar, mantenir i avaluar sistemes i serveis software complexos i/o crítics. [In depth]

Analysing, designing and implementing a Blockchain project required a breadth of technical knowledge.

The project is composed of multiple components, some external and some selfmade. The project had the complexity of not being able to adapt all the components to the project needs, instead, I had to make consider tradeoffs to actually achieve the platform goals without re-implementing the third party components.

- **CES1.2:** Donar solució a problemes d'integració en funció de les estratègies, dels estàndards i de les tecnologies disponibles. [In depth]

In order to make this project viable, I had to create a platform that had API integrations and URI-Schema integration via Browser Web Extension.

The nature of this project is how I integrated the different components to collaborate together to achieve the project goals. During that integration, I considered different approaches and studies the tradeoffs of each. See point 9.5.2. Approaches to create content and 9.7. Storing Information in Blockchain.

- **CES1.3:** Identificar, avaluar i gestionar els riscos potencials associats a la construcció de software que es poguessin presentar. [A little bit]

During the project planning, I evaluated the different risks associated with the project development, see point 4.4. Alternatives and action plan.

Since the project was using a quite new technology, it was expected to find issues that I was not able to predict when implementing the solution, that is the reason why I chose to use an agile methodology that helped me to make the process visible as soon as possible and minimise the risks of finding late issues with the platform. That helped to minimize the impact of each component issue that I found during the implementation.

- **CES1.4:** Desenvolupar, mantenir i avaluar serveis i aplicacions distribuïdes amb suport de xarxa. [Enough]

I was able to implement the platform with Blockchain as backbone technology.

- **CES1.5:** Especificar, dissenyar, implementar i avaluar bases de dades. [A little bit]

Even though the source of truth is Blockchain, I used MongoDB to store the information with a schema that makes easier to query data for the different use cases. That schema is optimized to fetch information and search queries to find content.

- **CES1.7:** Controlar la qualitat i dissenyar proves en la producció de software. [A little bit]

As shared during the Sprint 0, see point 9.3. Spring 0 - Setting up the project, I created a Continuous Deployment system for the project. A Continuous Deployment is not only publish each commit to production but also all the good practices around this approach to create content.

A Continuous Deployment system requires a strong test suit to prevent push into production broken software. Because of that, I added Unit Testing and Integration Testing during the integration on both components. Those tests are part of the development cycle since I was using the Test-Driven Development methodology.

- **CES2.1:** Definir i gestionar els requisits d'un sistema software. [A little bit]

A huge part of Software Engineering project is to define good project requirements. I did a previous research on the market to know more about how others platforms behave, so I could argue why those requirements instead of others.

During the specification phase, I dedicated time to write down the Acceptance Criteria a some Mockups to be used as a guideline during the development as part of the Definition of Done.

Some of those requirements where reviewed during the implementation to find which features are more likely to be implemented and which are not. It was easy to review thanks the detailed information and known expectations of each feature. Without that information, it could have been difficult to decide which task is more complex than others.

See point 6. Specification and 7. Non-Functional Requirements

13.3. Future work

For future work, we can consider different feature paths to based on the business needs.

At first, we would need to finish the features related to *Space Administration* since they have an important weight on the business model proposed at the beginning, without it, we cannot discard or approve the business hypothesis.

Secondly, we could start working on different scenarios that can benefit the project:

- Introducing the Fee System:

The Minimum Viable Product had no Fee System due two reasons. First, we wanted to keep the user entry as low as possible since it is already more complex compared to traditional systems. The second, the NEM2 version Cow[18] had not the fee system implemented yet. On the next version, Dragon[19], it had the first fee system implemented. Making this feature approachable.

A Fee System on Blockchain could imply that the user would need a specific digital asset to pay to the Blockchain in order to create or promote content. In case of a Public Blockchain Network, it is common that, in order to announce any transaction, the user has to pay a small fee.

Investing time and resources into this feature is essential to find a viable way to make the platform grow.

- Multiple LogBook.Social implementations:

A benefit of using Blockchain is that multiple systems can share the same contract and be able to talk with each other. Currently, the Minimum Viable Product does not handle the scenario of content created outside the LogBook.Social web application but it can be possible that different systems could interact with it.

- Custom implementation of crucial Blockchain components:

[18] Cow is the version code name for the third milestone. NEM2 Catapult v3 Cow.
[19] Dragon is the version code name for the fourth milestone. NEM2 Catapult v4 Dragon.

As identified during the development of the project, a Blockchain project has a meaningful dependency on external components and their reliability and user experience are crucial to make the project successful.

Even doe some Blockchain components are mature and stable, the project requires all to share a stability and minimum features in order to rely on them.

It might be needed to create custom implementations of those components, mainly the Blockchain Explorer and the Blockchain Wallet, to fulfill the project requirements.

- Monitoring:

In order to identify the issues early, it is needed to add a Monitoring system, like Prometheus or Datadog, in order to know in real time how is the system behaving and send alarms when there is an issue.

- Command Sourcing improvement:

Since the MVP had low traffic, the system did not have meaningful performance issues.

The Recover State Process loads all the Commands from Blockchain into Memory and then process them in sequence. This works until today because of the low traffic, a performance improvement to mitigate the loading time as the content grows. One approach is to not remove all the content of the DataBase on each deploy, only one needed, and read from the last point the server consumed. Another approach is to use Apache Kafka[20] to decrease the network latency of reading from an HTTP API and just re-process the duplicated commands stored into Apache Kafka.

- Improve the User Interface Design

The user interface is based on the default Components of Vuetify[21] UI framework. The Web application needs further work to improve how it looks and

- Unpleasant content

[20] https://kafka.apache.org/
[21] https://vuetifyjs.com/

A problem of a fully distributed platform is when it is used for the wrong purposes. The project has no protection for adult content or illegal content. It is a serious problem that has to be resolved.

Bibliography

[1] Wikipedia contributors. Facebook–Cambridge Analytica data scandal, [Online; accessed 24-February-2019] URL https://en.wikipedia.org/wiki/Facebook%E2%80%93Cambridge_Analytica_data_scandal.

[2] Statista editors. Combined desktop and mobile visits to Reddit.com from February 2018 to January 2019 (in millions), 2019. URL https://www.statista.com/statistics/443332/reddit-monthly-visitors/.

[3] Wikipedia contributors. Blockchain, [Online; accessed 24-February-2019]. URL https://en.wikipedia.org/wiki/Blockchain .

[4] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2009. URL https://bitcoin.org/bitcoin.pdf - Bitcoin whitepaper.

[5] BloodyRookie gimre Jaguar0625 Makoto. NEM Technical Reference, 2018. URL https://www.nem.io/wp-content/themes/nem/files/NEM_techRef.pdf - NEM Technical Reference.

[6] Wikipedia contributors. Smart Contract, [Online; accessed 24-February-2019]. URL https://en.wikipedia.org/wiki/Smart_contract .

[7] Wikipedia contributors. The DAO (organization), [Online; accessed 24-February-2019]. URL https://en.wikipedia.org/wiki/The_DAO_(organization).

[8] Cointelegraph editors. Parity Multisig Wallet Hacked, or How Come?, 2017. URL https://cointelegraph.com/news/parity-multisig-wallet-hacked-or-how-come .

[9] Wikipedia contributors. Ethereum, [Online; accessed 24-February-2019]. URL https://en.wikipedia.org/wiki/Ethereum .

[10] Andreas Antonopoulos, Gavin Wood. Mastering Ethereum, [Online; accessed 20-March-2019]. URL http://shop.oreilly.com/product/0636920056072.do.

[11] Wikipedia contributors. Hyperledger, [Online; accessed 24-February-2019]. URL https://en.wikipedia.org/wiki/Hyperledger#Hyperledger_Fabric .

[12] Anthony O'Dowd, Venkatraman Ramakrishna, Petr Novotny, Nitin Gaur, Luc Desrosiers, Salman Baset. Hands-On Blockchain with Hyperledger, [Online; accessed 20-March-2019]. URL https://www.oreilly.com/library/view/hands-on-blockchain-with/9781788994521/.

[13] Brendan Burns. Designing Distributed Systems. O'Reilly

[14] Brendan Burns. Designing Distributed Systems. O'Reilly, 2018.

[15] Vaughn Vernon. Implementing Domain-Driven Design. Addison-Wesley, 2013.

[16] Gregor Hohpe and Bobby Woolf. Enterprise Integration Patterns. Addison-Wesley, 2003.

[17] Jez Humble and David Farley. Continuous Delivery. Addsion-Wesley, 2010.

[18] Roy Osherove. The Art of Unit Testing Second Edition. Manning, 2014.

Annex A: Glossary

General Blockchain Concepts

- Blockchain Node[22]: A Blockchain Node is a machine that is running the Blockchain software and is part of the Peer-to-Peer network.
- Block[23]: Block is the main data structure to store transactions and propagate the information inside the Blockchain Network.
- Address: An Address is a String that identifies an Account. This address is generated from the Public Key with an algorithm.
- Public Key: Alphanumeric string derived from the Private Key. It is used to verify the signature of the transactions made with the Private Key.
- Private Key: Alphanumeric string that allows sign transactions to transact assets between users. It must be secret and not known by anyone.
- Digital Asset:

A digital asset is how NEM2 Blockchain refers to tokens[24] or cryptocurrencies. On NEM2 Blockchain not all digital assets have a monetary value.

NEM has its main cryptocurrency, called XEM. That currency has a value because it can be traded in Exchanges for Bitcoins or Dollars. On the other hand, NEM offers the feature of creating any cryptocurrency, but those new cryptocurrencies that are inside the NEM Blockchain have no value yet since they are not available in any Exchange for example.

- Wallet: A wallet can be two things. 1) An encrypted file that stores the private key and has to be unlocked via password. 2) A software used to see the Account information, send transactions and more.
- Transaction Hash: A transaction hash is the transaction identifier. It is unique inside the blockchain and it is derived from the Transaction data
- Test Network: A Test Network is a network with different information that has no value. It is used for development and usually there are few nodes.

[22] https://en.bitcoin.it/wiki/Full_node
[23] https://en.wikipedia.org/wiki/Blockchain#Blocks
[24] https://blockchainhub.net/tokens/

- Main Network: A Main Network is the only network that can store cryptocurrencies. When we say that the users are able to transfer monetary value, it must happen on the Main Network.

Common Actions performed by Blockchain Users

- Sign a Transaction:

We refer to sign transaction to the process of using the Private Key, sign cryptographically a Transaction.

- Announce a Transaction:

We refer to announce a Transaction to the process of sending the Transaction signed to the Blockchain to be processed.

Annex B: Difficulties about Continuous Deployment on DigitalOcean

The decision of using Digital Ocean was because its simplicity and cheaper prices compared to its competitors. One of the implications that it had was that it had not support to handle a full Continuous Deployment system and secrets management; needed for MongoDB credentials for example, not related to the Private Key management.

Not using Load-Balancers

To simplify the Jenkins Pipeline from run the tests, build the Docker images and deploy new instances into production, I did not use Load-Balancers because then I had to manage which droplets to replace and I was running out of time for Sprint 0.

It caused that the system might be not accessible by the user because of a DNS resolution. Because the machine is replaced on each deploy, it has a new IP that has to be assigned to the DNS A Record, impling an interval of time where the system is not accessible until the DNS has been propagated over the Internet.

Blue-Green Deployment

Blue-Green deployment enables smooth deployments where, if something fails on deployment time, it rollback and recovers the previous state.

DigitalOcean does not provide any feature to do so, implying that Jenkins pipeline to be smart enough to make that process.

An alternative compatible with DigitalOcean is use the Kubernetes service that comes with this feature. It was not done in the beginning because it implied a huge learning curve and it was not the project purpose.

Secrets Management Service

Digital Ocean does not give a Secret Management Service compared to its competitors, like Amazon Web Secrets Manager.

In order to resolve this, the secrets were managed by Jenkins system. Making it the most vulnerable point on the whole system.

A partial solution could have been use Vault by HashiCorp. Vault is a software to manage secrets and protect sensitive data.

This solution was not implemented because the scope required was not giving enough value for the Bachelor's thesis since it was not the main purpose of the project make it extremely secure.

Virtual Private Networks

DigitalOcean enables private communication between droplets but it has not a easy way to create a VPN to secure the sensible services like Jenkins and Nexus Repository. During the project development, those have been accessible but with strong credentials to minimize the risk of third party access.

An alternative could have been create self managed VPN and configure the sensible services to make them accessible via the VPN.

It has not been implemented because the project did not require this level of security and it did not hold any sensible data.

Annex C: Icon credits

- Profile icon: Designed by Freepik from www.flaticon.com
- Application icon: Designed by prettycons from www.flaticon.com
- Blockchain icon: Designed by ddara from www.flaticon.com
- Wallet icon: Designed by smashicons from www.flaticon.com