# Sustainable Open Access Publishing using Blockchain, Interledger and Web Monetization

# Publicación de acceso abierto sostenible mediante Blockchain, Interledger y monetización web

By

Elena Pérez Tirador

Departmento de Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid

Trabajo Fin de Grado del Doble Grado en Ingeniería Informática y Matemáticas, Facultad de Informática, Universidad Complutense de Madrid.

2020/2021

Directors:
Antonio A. Sánchez Ruiz-Granados
Ámbar Tenorio Fronés

# Abstract

One of the main problems the **academic community** faces is the high prices of publishing and accessing academic papers. Authors and reviewers usually work for free or even pay for their work to be published, while the revenue generated goes to the big publishers.

This project is centered around studying the technologies that would be best suited for the creation of a system or platform for **donations** to journals and researchers. Various centralized and **decentralized** technologies have been studied. Regarding centralized technologies, **PayPal** has been the main focus, and regarding **decentralized** technologies, **Blockchain**, **Interledger** and **Web Monetization** have been studied. Their advantages and limitations have been analyzed, as well as their different integration possibilities.

These technologies have been used to implement a system that allows **donations** by different centralized and **decentralized** means. The **donations** can be sent between any two centralized or **decentralized** currencies. The system's main feature is a **donation** button that can be integrated in any journal's web page. JavaScript and React were used for the front-end implementation. This connects with the underlying subsystems that manage the different types of **payments**. To test the possibilities of the **decentralized** technologies, a new ERC20 token was implemented and integrated in the system.

It was concluded that centralized technologies are easier to use for implementation, better documented and more intuitive for the user. However, PayPal, for example, imposes certain barriers and limitations that block the developers' work. **Decentralized** technologies, on the other hand, are newer, so they have less documentation and are a bit more cumbersome to use. But they have proven to be way more flexible and adaptable and easier to integrate with. So, despite their limitations, they have potential to become an important part of online **donations** and **payments** in the future.

**Keywords:** Academic community, Blockchain, decentralization, donations, Interledger, PayPal, online payments, Web Monetization

# Resumen

Uno de los principales problemas que afronta la **comunidad académica** son los altos precios para publicar artículos y acceder a ellos. Habitualmente los autores y revisores trabajan gratis, o incluso tienen que pagar para poder publicar su trabajo, mientras que son los grandes editores quienes reciben los beneficios generados.

Este proyecto se centra en el estudio de tecnologías que puedan resultar adecuadas para la creación de un sistema o plataforma para **donaciones** a revistas académicas e investigadores. Se han estudiado diversas tecnologías centralizadas y **descentralizadas**. Entre las centralizadas, **PayPal** ha sido la principal, y entre las **descentralizadas**, se han estudiado **Blockchain**, **Interledger** y **Web Monetization**. Se han analizado sus ventajas y limitaciones, así como sus diferentes posibilidades de integración.

Estas tecnologías han sido utilizadas para implementar un sistema que permite realizar **donaciones** por vías centralizadas y **descentralizadas**. Las **donaciones** pueden realizarse entre cualesquiera dos divisas diferentes, ya sean centralizadas o **descentralizadas**. El elemento principal del sistema es un botón de **donación** que puede integrarse en la página web de cualquier revista. Para la implementación del front-end se utilizaron JavaScript y React. Esto se conecta con el subsistema subyacente que se encarga de los diferentes tipos de **pagos**. Para probar las posibilidades de estas tecnologías **descentralizadas**, se ha implementado un nuevo token ERC20 y se ha integrado en el sistema.

Se comprobó que las tecnologías centralizadas son más fáciles de usar en implementación, están mejor documentadas y son más intuitivas para el usuario. Sin embargo, PayPal, por ejemplo, impone ciertas barreras y limitaciones que pueden bloquear el trabajo de los desarrolladores. Las tecnologías **descentralizadas**, por contra, son más nuevas, por lo que disponen de menos documentación y son algo más engorrosas de utilizar. No obstante, han demostrado ser más flexibles, adaptables e integrables. De modo que, a pesar de sus limitaciones, tiene el potencial de convertirse en una parte importante de los sistemas de **pagos** y **donaciones** en el futuro.

**Keywords:** Blockchain, comunidad académica, descentralización, donaciones, Interledger, PayPal, pagos online, Web Monetization

# *Acknowledgements*

After a long while, this project is coming to an end. This year has undoubtedly been one of the weirdest up to this day for everyone. Therefore, I would like to thank everyone that made this project possible, despite the weird conditions we all are in. So thanks to:

- *Antonio Sánchez* and *Ámbar Tenorio*, for directing and supervising this project.
- *Grant for the Web* and the *Interledger Foundation* for financing it.
- The *Quartz OA team*, for making the Quartz project a reality.
- *My family*, for all the invaluable support.
- *My friends*, for helping me keep my sanity in this difficult year.

Thank you very much.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **API** | **A**pplication **P**rogramming **I**nterface |
| **CTA** | **C**all **T**o **A**ction |
| **ERC** | **E**thereum **R**equest for **C**omments |
| **GDPR** | **G**eneral **D**ata **P**rotection **R**egulation |
| **HTML** | **H**yper**T**ext **M**arkup **L**anguage |
| **IDE** | **I**ntegrated **D**evelopment **E**nvironment |
| **ILP** | **I**nter**L**edger **Protocol** |
| **IPFS** | **I**nter**P**lanetary **F**ile **System** |
| **JSON** | **J**ava**S**cript **O**bject **N**otation |
| **MVP** | **M**inimum **V**iable **P**roduct |
| **ORCID** | **O**pen **R**esearcher and **C**ontributor **ID** |
| **QTZ** | **Q**uar**tz** Token |
| **REST** | **Re**presentational **S**tate **T**ransfer |
| **URL** | **U**niform **R**esource **L**ocator |
| **WM** | **W**eb **M**onetization |

*To my family*

# Chapter 1

# Introduction

Scientific production is a very demanding and dedicated job, as well as an immensely necessary one. But this kind of work, in most cases, is not paid or compensated in any way [38]. Most authors, editors and reviewers work for free (or almost for free), and at the same time have to pay huge amounts of money to access the scientific contents published in most journals [3, 51].

The main reason for this is that academic and scientific publishing is primarily controlled by a reduced group of big publishers (such as Elsevier, Springer Nature or Wiley) [33]. Although the majority of the work involved in this complex process is carried out by the academics, taking the role of authors, reviewers, editors, etc. most of the benefits generated by this market go to these big publishers [32].

These benefits often come from the high prices researchers need to pay to access the content [5]. There is an option for the authors to publish their articles in Open Access, so whoever wants to access the content can do it for free, for it is open. However, in exchange for this, researchers have to pay very expensive prices [60].

It is not uncommon that, for these reasons, academics feel frustrated, since they can feel like their work is not properly recognised and compensated. And also for some, publishing in Open Access is not an option, because they can not afford the high prices [23].

The problem with the academic publishing rises from the centralization of its more powerful actors (the big publishers). For this reason, it is interesting to study decentralized solutions to try and tackle these kind of problems. Nonetheless, it is known that these technologies are still not very accessible for most people because of their sometimes cumbersome usability [1], so it is also necessary to combine these decentralized technologies with some centralized solutions to make adaptation easier.

Another matter that is also worth exploring is the rewards for the academic community [40]. These rewards could materialize in different forms, whether they be monetary or not [65]. For these reason it is also interesting to explore the possibilities that cryptocurrencies or tokens bring to this context.

In this context, several initiatives to help improve the situation of the academic community begin to appear [18]. One such initiatives is Decentralized Science[1] [57, 55], a project aiming to provide an open community of reviewers, open peer reviews and a reputation network, providing rewards for the peer reviewing work. As a natural continuation of Decentralized Science, and thanks to the funding received from Grant

[1] https://decentralized.science/

For The Web[2], the Quartz OA[3] project is born, aiming to create a community of academics and journals that enables donations in a distributed way. By bringing these additional sources of capital, they strive to make Open Access fairer, more sustainable and independent [34].

This TFG is part of the Quartz OA project, in collaboration with the ISIA department of the Facultad de Informática at UCM [4].

## 1.1 Objectives

The goal of this project is to create a system (the *Quartz* system) to facilitate donations to journals and authors, in order to reward their work. Specifically, the project is centered in exploring what centralized and decentralized technologies can be used for this purpose, and find out their strengths and weaknesses, while using them to create the system.

In the case of decentralized technologies, the Interledger Protocol [56] will be studied, exploring its applicability and limitations in order to introduce it into the system. Interledger proves to be interesting, as it is a new technology that allows intercommunication among ledgers (both centralized and decentralized). This means exchanges between any two coins or payment systems are possible. In particular, this project will explore how new payment systems can be integrated into this system to enable different kinds of payments.

The concept of microdonations will also be explored, as a way to allow users visiting websites to pay based on the time they spend in them.

In terms of development, the goal is to create a donation button that can be integrated in the web page of a journal, allowing the visitors to perform donations in different ways:

- Via a centralized payment system, such as PayPal.
- Via a decentralized payment system, using any currency. In particular, a specific coin created for donations to journals.
- Via microdonations, using a browser extension.

## 1.2 Planning

The development of this project took place between September 2020 and May 2021. This document was written in the following months.

The first four months of the project were centered in studying the problem, analyzing the possible solution and developing the first centralized solution. This is, a button allowing centralized donations.

The next four months of the project were centered in exploring the decentralized possibilities, essentially Interledger and Web Monetization (both will be explained in detail in the next chapters of this document), and developing the decentralized solution. This is, adding the button the possibility to perform donations via Interledger and integrating with Web Monetization so that the user can send micropayments while visiting the journal's website.

[2]https://www.grantfortheweb.org/
[3]https://quartz.to/
[4]https://www.ucm.es/disia

The rest of the time was dedicated to polish the application and write this document.

In terms of methodology, scrum [8] was applied, implementing two-week sprints. Every other Thursday, a meeting was held with the directors of the project to review the work of the previous two weeks and decide the steps to be taken in the next sprint.

For the final phase of polishing and writing the document, the sprints were changed to one-week sprints.

## 1.3 Structure of this document

This document explains the development process of the app, along with initial considerations and final results.

Chapter 2 introduces the current context in which the project is developed. It explains the importance of donations for open software projects, mentioning Wikipedia as a well known example. It also introduces PayPal as a technology for online payments and gives various examples of donation platforms, both using centralized and decentralized currencies.

Chapter 3 explores the technologies that enable the creation of a distributed donation platform. It introduces the Interledger Protocol and Web Monetization as two new technologies that can be used for this purpose. It also gives some examples of applications using these technologies.

Chapter 4 gives an overview of the implemented system from the user's perspective, explaining the actors that will interact with it and the main functionalities it provides. The workflow is explained in detail, illustrated by screenshots of the system.

Chapter 5 explains the implementation of the centralized part of the system. It first gives some implementation details about PayPal and its API and then shows and explains the architecture of the system, as well as its flow of interactions (through sequence diagrams).

Chapter 6 explains the implementation of the decentralized part of the system. It first gives implementation details about Interledger and the architecture of an Interledger network. Secondly, it does the same for Web Monetization. It then explains the integration of new coins in this architecture and finally describes the architecture, implementations details and flow of interactions of the implemented system.

Chapter 7 analyzes the conclusions that can be extracted from the project, regarding its initial objectives, and then proposes the next steps that can be taken to further develop and improve the system and the project.

## 1.4 Source code

The project has been developed using JavaScript, HTML, CSS and Solidity. The source code for the project, along with some deployment instructions can be found in the following GitHub repository:

https://github.com/ElenaPT/TFG_Inf_2021

The project is licensed under a GNU General Public License v3.0.

# Chapter 2

# Web donations and platforms

This chapter studies how web donations are working nowadays. First, an introduction is given illustrating the importance of donations in open source projects, and why they are so necessary. Then, Wikipedia is introduced as an example of a well known project that is financed using donations. After that, PayPal is briefly described as a tool for enabling donations and payments through the web, since it is the most commonly used technology for this purpose. Lastly, some donation platforms and similar projects are listed and analyzed in terms of their usability.

## 2.1 Introduction

Nowadays, open source projects are very common and serve as a basis to both open and non open source software development [54]. However, most of the time the work in open source is not well recognized, and financing this sort of projects can be challenging [11]. Although multiple funding models exist in the context of open source creation, donations through platforms such as PayPal or Patreon are one of the most common practices [45].

Open software development is usually a voluntary work, and there are different opinions on whether or not to accept monetary rewards [58]. Participating in the development of open source software can start from both an intrinsic motivation (the development itself is motivating, for it is interesting or challenging) or an extrinsic motivation (academic rewards fall in this category) [29].

The importance of donations relies on their voluntary nature, for an obligation or coercion to finance the developers would defy the philosophy behind open software itself [52].

From the perspective of the donor, the will to donate can come from different reasons. It could just be an altruistic decision, but some other factors can be in play, such as private gains derived from the software being developed in the end, or some kind of recognition or prestige [31].

We think Open Access publication is not that different from Open Source development in these aspects, so this same reasoning can be applied [64].

## 2.2 Analysis and categories

The following sections of this chapter will study existing applications and technologies that are similar to the one being developed (under the name of *Quartz* or *QuartzOA*).

The technologies and applications have been researched and then subjectively analyzed and scored, based on a group of categories. The goal of this analysis is comparing the applications and finding out their strengths and weaknesses, in order to learn from them. This way, the best qualities can be taken as an example and the mistakes can be avoided.

### 2.2.1 Categories

The first step is defining the categories around which the analysis will be performed. These categories define what will be understood as an overall *good* system.

The categories that will be analyzed are the following:

- **Accessibility:** This category measures to what degree the donation tool can be accessed following a fluid process, with no dilatory intermediate stages.
- **Interest:** This category measures the degree in which the user identifies with the values and purposes of the tool.
- **Usability:** This category measures how structured and comprehensive the information is displayed in the tool, and whether it is distributed through the channels most frequented by users.
- **Dialog:** This category measures how agile the exchange of information between the tool and the user is.
- **Effectiveness:** This category measures to what degree the process design helps to achieve the objectives in a reasonable amount of time.

### 2.2.2 Analysis and score

For every application analyzed that has a specific system for performing donations, these categories will be studied. For each, a description will be given along with a numeric score ranging from 0 to 5, where 0 means that the category is not fulfilled at all by the application and 5 means it is perfectly fulfilled.

It is important to note that both the analysis and the scoring system are subjective. In any case, they are still very useful measures for gaining a general knowledge of the current donation tools.

## 2.3 Wikipedia

*Wikipedia*[1] is a well known free collaborative online encyclopedia. It is maintained as an open collaboration project by a community of independent volunteer editors using a wiki-based editing system. The project is managed by a nonprofit organization (Wikimedia Foundation) that is financed via donations. The organization raised more than \$110M in 2019 and more that \$120M in 2020[2].

Since the project is financed by donations, it has its own built-in system to facilitate them. It consists of a simple interface that allows the user to select an amount and choose a way to perform the payment (credit card, PayPal). Usually, the user has to actively go to the donation page to perform the donations, but occasionally Wikipedia

[1] https://en.wikipedia.org/

[2] Wikimedia Foundation Financial Statements, 2019-2020: https://upload.wikimedia.org/wikipedia/foundation/f/f7/Wikimedia_Foundation_FY2019-2020_Audit_Report.pdf

performs donation campaigns, displaying the donation module in every page of the site.

Regarding the defined categories, the following can be concluded:

- **Accessibility:** (5/5) The donation page is slightly hidden in the lateral menu. However, the donation module makes operations easy. The multitude of amounts and ways that are available also facilitate the operation. In general, clarity of information.
- **Interest:** (3/5) The language used is clear and inspiring, but the way the information is showed is not very attractive.
- **Usability:** (5/5) The web provides a detailed description of what the user is doing.
- **Dialogue:** (4/5) The exchange of information between the page and the user is agile.
- **Effectiveness:** (2/5) Effectiveness is somewhat diminished by the fact that the option is slightly hidden in the menu. Although it is a first level option, the page is full of information, what makes it difficult for the user to read everything and find the option. A user willing to make a donation would more likely do a google search to get to the page than find it in the main page's menu. It should also be taken into account that it is one of the most famous websites on the Internet, with very different targets.

It is also worth mentioning that Wikipedia is a free software.

## 2.4 PayPal

The first approach that was taken in the development of the project was a centralized one. For this centralized approach, PayPal was used in order to create a donation button that could be integrated within the journals's webpages.

PayPal is one of the most important payment platforms worldwide. It allows users to transfer money to others and make online payments. It supports over 20 different currencies, including Euros, Pounds, US Dolars and all the main currencies worldwide.

PayPal offers a REST API, along with several tools for developers. Given the simplicity to access these tools and the wide scope PayPal provides as a technology, it was regarded to be a good starting point for the implementation of this project.

Unlike the rest of the applications studied in this chapter, PayPal will not be analyzed in the same way, following the categories and giving a score. The reason for this is that what PayPal offers is a solution that different applications and tools can integrate. Therefore, most of the categories would get different scores depending on how said integration is carried out. Moreover, some of the tools that will be studied use PayPal as a payment system, and each of them gets different scores depending on how the application as a whole is designed.

For this reason, PayPal is presented as a tool, and studied as such.

### 2.4.1 API

PayPal offers a REST API to manage payments among accounts. By creating a PayPal account, the user is granted credentials which can be used to produce a token that allows the user to make the REST API calls. Unauthorized calls return a failure. For this authorization of the API calls, OAuth 2.0 protocol is used [24].

OAuth is an open standard for simple authorization flows in websites, applications or, as in this case, APIs.

### 2.4.2 Marketplaces and platforms

Apart from the basic functionalities, PayPal also offers a full-stack solution for processing and managing a marketplace commerce platform.

A platform or marketplace is a structure for managing a group of *sellers*, which are organizations or entities that are selling products or providing services. The platform acts as an intermediary between this sellers and the *buyers*, which are users interested in purchasing the items they are selling or the services they are providing. It gives the buyers the tools for sending payments to the sellers.

Quartz's case is an abstraction of this idea. The *sellers* are the journals. They are not directly selling a product or service, but they are providing access to papers and other valuable resources. In a similar sense, the *buyers* are the donors. They are not directly purchasing a product or service, but they are making donations. There is not a direct exchange of money for goods, but the structure works the same way, since the goal is to give the donors the tools to send money to the journals, in a context where multiple journals exist.

In PayPal's solution, the sellers have to be registered in the platform to be a part of it and be able to receive payments. The process of registering a seller in the platform is called *onboarding*. Once the seller is onboarded, the platform can create specific payment or donation buttons for the seller. The platform can then manage the payments, meaning the way the money is sent and when. For example, they can allow periodic payments, or charge a small fee for the service.

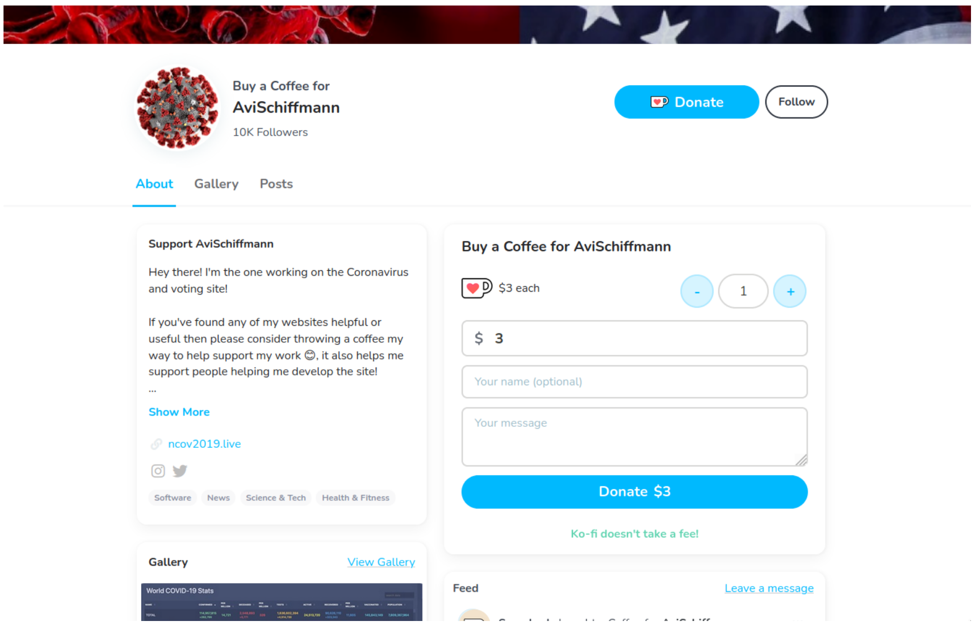
Prior to this, the platform needs to be properly defined. There are two main aspects that have to be defined. First, PayPal offers different types of accounts for its users, and the platform has to decide which of them should be allowed to be sellers. Second, the platform can decide what payment methods should be implemented (PayPal accounts, credit card...).

Once the platform is defined, the sellers can onboard, and once they are onboarded, the payments can begin to be performed.

## 2.5 Centralized donation platforms

This section centers on the analysis of systems designed to make donations to one or various projects using centralized payment systems. Namely: *Ko-fi*, *Flattr* and *Plaudit*.

*Goteo*, *Patreon* or *PayPal Me* are similar systems, that were also considered during the analysis. However, for the sake of simplicity, these will not be studied in detail in this document.



Support AviSchiffmann

Hey there! I'm the one working on the Coronavirus and voting site!

If you've found any of my websites helpful or useful then please consider throwing a coffee my way to help support my work 😊, it also helps me support people helping me develop the site!
...

Buy a Coffee for AviSchiffmann

$3 each

$ 3

Your name (optional)

Your message

FIGURE 2.1: Profile page of one of Ko-fi's featured creators

### 2.5.1 Ko-fi

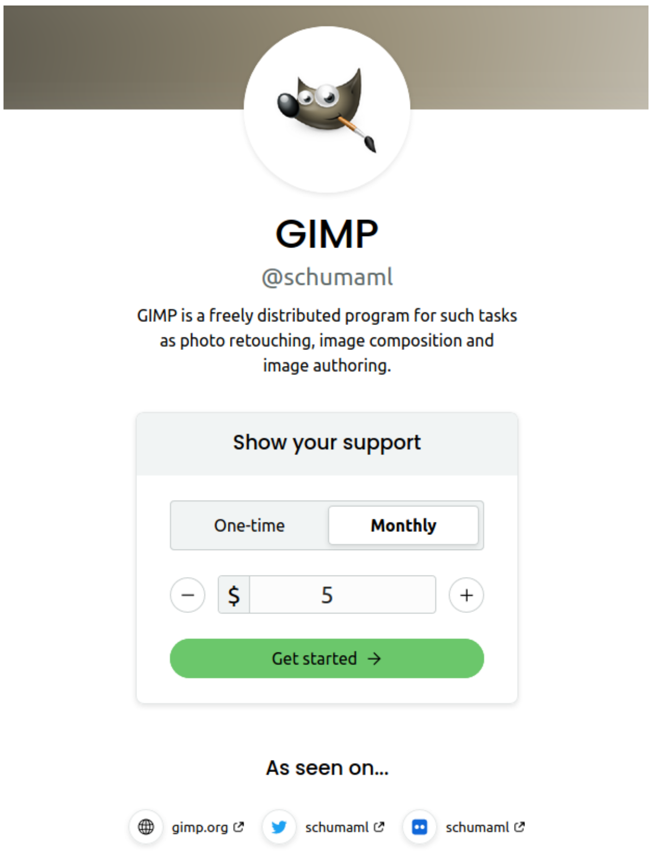
*Ko-fi* [30] is a platform that allows the user to make micro-donations and support their favorite artists. It is built around the metaphor of inviting the artist to a cup of coffee, which, in the context of the page, translates into a donation of $3 (at least).

The website harbors a collection of pages, one for each creator inside the community. In the page, the creator can display their information and other content (images, etc.). Within the page there is also the option for any user to donate *a coffee* to the creator. A creator page can be seen in Figure 2.1.

A Ko-fi button can also be integrated in any website to facilitate the donations. So any creator that uses the page can integrate the Ko-fi button within their own personal web page.

Regarding the categories defined in Section 2.2.1, the following can be concluded:

- **Accessibility:** (5/5) Inside the website, the donation button can be easily found in the top menu. It is also easy to integrate the button into another web page.
- **Interest:** (5/5) The website is very attractive and uses inspiring language and illustrations. There is a "Explore" page with highlights, featured artists, categories and a browser, which makes it easy and interesting to discover people. The narrative of the site is social network-like.
- **Usability:** (4/5) The coffee cup animation makes the button nice and friendly. The possibility to change the color of the button makes the experience like a kind of friendly "gamification". The process for creating a page, as a creator, is also straightforward.
- **Dialogue:** (4/5) The exchange of information between the page and the user is agile.
- **Effectiveness:** (5/5) The site is very popular among internet content creators (such as artists, writers, podcasters...).


GIMP

@schumaml

GIMP is a freely distributed program for such tasks as photo retouching, image composition and image authoring.

FIGURE 2.2: Profile page of Gimp in Flattr

### 2.5.2 Flattr

*Flattr* [16] is also a platform that allows the user to make micro-donations and support their favorite artists. It is similar to Ko-fi, but it focuses on free (in the sense of open) content creators.

The donations can be made in different ways. As other similar sites, it allows the user to make a specific donation to a specific artist, that will be directly sent to them. It also provides the option to pay a monthly subscription, which will distribute the money among the user's favorite creators.

The website harbors a collection of pages, one for each creator inside the community. In the page, the creator provides a small description, links to their personal pages and the donation button itself (with the option of one-time and monthly donations). A creator page can be seen in Figure 2.2.

Regarding the categories defined in Section 2.2.1, the following can be concluded:

- **Accessibility:** (4/5) There is a section to contribute and a list of some featured creators, but it doesn't seem like there is a place where all creators can be found. It is nonetheless easy to contribute. The page is somewhat similar to Ko-fi, but a little more difficult to navigate. The Log-in button is very clearly accessible in the homepage. On pages where you can contribute is also quite accessible. The donation button inside a creator's page is also easily accessible.

- **Interest:** (3/5) The website has clear instructions and is overall correct, but it is not too attractive. Based on the type of content the page hosts, it is assumed that the user has interest in free software.
- **Usability:** (4/5) The situation of the Call to Action for becoming a contributor is correct and clear. The same applies to the donation button.
- **Dialogue:** (3/5) It is possible to register both as a contributor or as a creator. The forms are standard.
- **Effectiveness:** (4/5) The site is quite popular among free software interested people.

It is also worth mentioning that Flattr is a free software.

### 2.5.3 Plaudit

*Plaudit* [46] is a browser extension that allows the user to endorse academics and their research. It is designed around the idea that, since these endorsements are made by respected members of the academic community and are also publisher-independent, they provide credibility for valuable research.

Regarding the categories defined in Section 2.2.1, the following can be concluded:

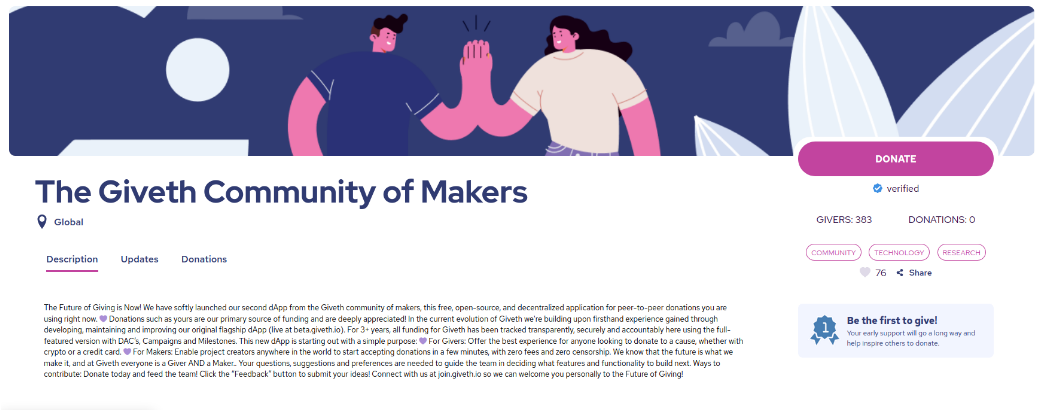
- **Accessibility:** (3/5) Searching the creators is not easy. The page does not have an explicit search functionality. On the other hand, contributing on the paper's website is easy with the Plaudit button. However, the website does not provide clear examples of how this can be done.

  The button to get the extension in the homepage is very clear, and so is the donation buttons in the pages where it is integrated.
- **Interest:** (4/5) The website has clear instructions and is overall correct, but it is not very attractive. The site states that the technology is easy to integrate, but it requires coding, so it is targeted to an audience that understands code. They use funny copies in the tutorial (such as Albert Einstein endorsing your work) in order to show how the product works.

  There is a lot of demand for these type of initiatives in the academic world, so it is likely to generate a lot of interest, although it is not easy to find what the actual number of people using the system is.
- **Usability:** (2/5) The situation of the Call to Action is correct. The process of contributing can only be performed if the user has an ORCID account. The process for integrating the button within a paper web page is not straightforward.
- **Dialogue:** (3/5) The exchange of information between the page and the user is correct but not too clear. The forms used are standard.
- **Effectiveness:** (4/5) The site is getting some recognition among certain sectors of the academic community because there is a great need for this type of initiatives, but it can still be improved upon. It seems that is not yet a very mature product.

Plaudit is a lesser known system than the rest, and it is not really focused on donations but on endorsements. However, it was considered as an interesting case study, since it is a tool targeted for the academic community, just as Quartz.

It has been included in the present section because it is, indeed, centralized and, although it is not a donation platform, it has much in common with them.



The Giveth Community of Makers

Global

The Future of Giving is Now! We have softly launched our second dApp from the Giveth community of makers, this free, open-source, and decentralized application for peer-to-peer donations you are using right now. 💜 Donations such as yours are our primary source of funding and are deeply appreciated! In the current evolution of Giveth we're building upon firsthand experience gained through developing, maintaining and improving our original flagship dApp (live at beta.giveth.io). For 3+ years, all funding for Giveth has been tracked transparently, securely and accountably here using the full-featured version with DAC's, Campaigns and Milestones. This new dApp is starting out with a simple purpose: 💜 For Givers: Offer the best experience for anyone looking to donate to a cause, whether with crypto or a credit card. 💜 For Makers: Enable project creators anywhere in the world to start accepting donations in a few minutes, with zero fees and zero censorship. We know that the future is what we make it, and at Giveth everyone is a Giver AND a Maker.. Your questions, suggestions and preferences are needed to guide the team in deciding what features and functionality to build next. Ways to contribute: Donate today and feed the team! Click the "Feedback" button to submit your ideas! Connect with us at join.giveth.io so we can welcome you personally to the Future of Giving!

Be the first to give!
Your early support will go a long way and help inspire others to donate.

FIGURE 2.3: A verified project's profile page in Giveth

## 2.6 Decentralized donation platforms

This section centers on the analysis of systems designed to allow the user to make donations using cryptocurrencies and blockchain. Namely: *Giveth*, *Helperbit* and *Givetrack*.

### 2.6.1 Giveth

*Giveth* [21] is a platform that allows the user to make donations to social good projects without commissions or other secondary charges, by using the blockchain technology. They describe themselves as a "Decentralized Altruistic Community".

The website harbors a collection of pages, one for each project inside the community. Inside each page, there is information about the project and their goals, a section for updates and a summary of the donations, showing the total amount donated to the project and a list of the donors ordered from most recent to least recent. A project's page can be seen in Figure 2.3.

As for now, the donations can only be done using cryptocurrencies (DAI, YAY, UNI or ETH). A payment by credit card is being implemented, but is not working yet.

Regarding the categories defined in Section 2.2.1, the following can be concluded:

- **Accessibility:** (5/5) Inside the website, the donation button can be easily found in the home page. There is a "Projects" page that is easy to find in the top menu. From there the donation button for each project is also easy to find and clear to use. The login button and the one for creating a project are also very visible in the home page.
- **Interest:** (4/5) The website is attractive and uses inspiring illustrations and language. The "Projects" page makes it interesting to discover new projects. The fact that all the projects featured are for social good helps the user identify with the values.

  It expects the user to be interested in cryptocurrencies and their use (the donations can only be made using cryptocurrencies).
- **Usability:** (2/5) The donation button is big and clear. However, the fact that the donations can only be performed using cryptocurrencies makes the process a little cumbersome, especially for people not very familiarized with the field.

  The process for logging-in and creating a project seems to not be working well.

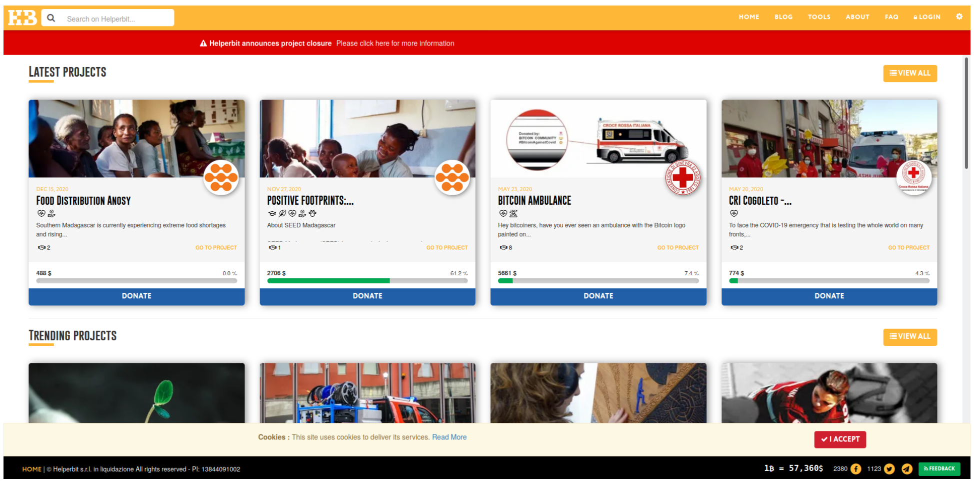

HOME BLOG TOOLS ABOUT FAQ LOGIN
Helperbit announces project closure Please click here for more information
LATEST PROJECTS
VIEW ALL
FOOD DISTRIBUTION ANOSY
POSITIVE FOOTPRINTS:...
BITCOIN AMBULANCE
CRI COGOLETO -...
GO TO PROJECT
DONATE
TRENDING PROJECTS
Cookies : This site uses cookies to deliver its services. Read More
I ACCEPT
FEEDBACK

FIGURE 2.4: Helperbit main page

- **Dialogue:** (4/5) The exchange of information between the page and the user is agile.
- **Effectiveness:** (3/5) The platform is not very popular yet, although it has a decent number of users. It has a limited niche, because it requires prior knowledge and usage of the technology. However, inside its niche it is the most popular technology for donations.

### 2.6.2 Helperbit

*Helperbit* [53] is a platforms that allows the user to make donations to charities using cryptocurrencies, so the charities can receive the money in a safe way.

The website harbors a collection of pages, one for each charity inside the community. In the page, the charity can display information about what they do and some related media, such as photos or videos, that can illustrate their labor. The page also shows the income goal of the organization and the quantity that has already been donated. There is a subsection where a list of the donors is displayed. The main page of the website can be seen in Figure 2.4.

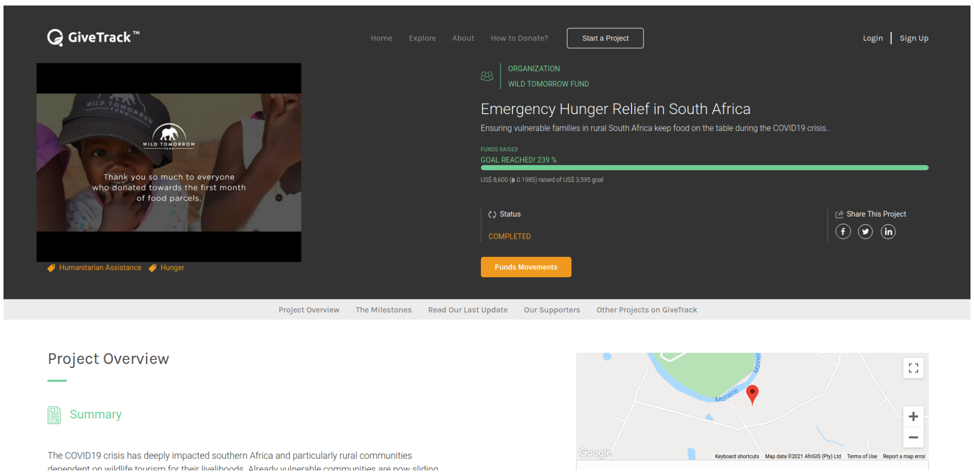
The donations can be done using different cryptocurrencies, via Metamask[3] [37] or from the Helperbit account of the user.

Regarding the categories defined in Section 2.2.1, the following can be concluded:

- **Accessibility:** (5/5) The main page of the website shows the different charities that are registered in the community and can receive donations. The button for donating to a charity is very clear and easy to find. The login button is also easy to find in the top menu.
- **Interest:** (4/5) The interface is not the most attractive one, but it is sufficiently clear. The main page showing the organizations by categories (latest projects, trending projects...) makes it easy to discover the collectives.

  The fact that the page is for donating to charities makes it easy for the user to identify with the values, which are clear and relatable.

[3]Metamask is a browser extension for managing crypto wallets and operating with cryptocurrencies.



ORGANIZATION
WILD TOMORROW FUND

Emergency Hunger Relief in South Africa

Ensuring vulnerable families in rural South Africa keep food on the table during the COVID19 crisis..

FUNDS RAISED
GOAL REACHED! 239 %

US$ 8,600 (฿ 0.1985) raised of US$ 3,595 goal

Status: COMPLETED

Project Overview

Summary

The COVID19 crisis has deeply impacted southern Africa and particularly rural communities dependent on wildlife tourism for their livelihoods. Already vulnerable communities are now sliding

FIGURE 2.5: Profile page of a project in GiveTrack. The project has already reached its milestone.

- **Usability:** (3/5) The donation button is very clear and it stands out. The donations can be done using cryptocurrencies or through the account created in the website.
- **Dialogue:** (3/5) The exchange of information between the page and the user is agile, although a bit scarce in some situations.
- **Effectiveness:** (2/5) There is a decent number of charities inside the community, but it does not seem like the initiative is very well known. Most of the charities are quite far from their income goals. Also, it seems that the webpage will close because the company running it could not sustain itself economically. So apparently things did not work out well.

Apparently the project will disappear by the end of may 2021, and the page explains how the charities can do to withdraw their funds.

### 2.6.3 GiveTrack

*GiveTrack* [22] is a platform that allows the user to make donations to non-profit organizations and NGOs using the blockchain technology to provide real-time tracking of the funds. The project is developed by the BitGive Foundation [4].

The website harbors a collection of pages, one for each nonprofit organization inside the community. In the page, the organization can display an overview of their project and goals. The page also shows the fund milestones of the organization and the quantity that has already been donated. An organization's page can be seen in figure 2.5.

The donations can be done using different cryptocurrencies or using a credit card.

Regarding the categories defined in Section 2.2.1, the following can be concluded:

- **Accessibility:** (5/5) Inside the website, in the home page, the button "Explore projects" is the main call to action and can also be found in the top menu. The donation button inside each project is also very easy to find and stands out. The login and sign up buttons are also visible and accessible in the top right.
- **Interest:** (5/5) The page is very clean and attractive and uses inspiring language. The "Explore" page makes it interesting to discover the different initiatives.

The fact that the projects are nonprofit and NGOs makes it easy for the user to identify with the values.

- **Usability:** (5/5) The donation button is very clear and the bar showing how much money is left to reach the milestone encourages the user to donate.

  The payment can be done using Bitcoin, credit card or via Uphold [59], which supports a lot of different currencies. The variety of options makes if quite easy to donate. The user is required to have an account in GiveTrack in order to donate. Creating an account is easy.

  The process for creating a project is straightforward. It requires government approval, because it is meant for recognized organizations.
- **Dialogue:** (5/5) The exchange of information between the page and the user is agile.
- **Effectiveness:** (4/5) There is a decent amount of organizations and a good percentage of them are reaching their milestones.

## 2.7 Conclusions

Each of the platforms that were studied in this Chapter were analyzed following the categories described in Section 2.2.1. During this analysis, for each of the categories in each of the applications, a subjective numeric score was given. The score ranges from 0 to 5, where 0 means that (according to a subjective criteria) the category is not fulfilled at all by the application and a 5 means it is perfectly fulfilled.

For certain categories, several applications may cover them, to an extent. However, the score for a certain category for a certain application will be lower if there are others that fulfill said category more successfully.

These numeric scores serve as a quantitative measure of how adequate each platform is, in terms of its interaction with the user. It is useful for the development of the Quartz platform, since it serves as a guide to which practices give the best results, in order to make the user experience more pleasant.

Table 2.1 shows the scores for each category in each one of the analyzed platforms.

| | WIKIPEDIA | KO-FI | FLATTR | PLAUDIT | GIVETH | HELPERBIT | GIVETRACK |
|---|---|---|---|---|---|---|---|
| ACCESSIBILITY | 5 | 5 | 4 | 3 | 5 | 5 | 5 |
| INTEREST | 3 | 5 | 3 | 4 | 4 | 4 | 5 |
| USABILITY | 5 | 4 | 4 | 2 | 2 | 3 | 5 |
| DIALOGUE | 4 | 4 | 3 | 3 | 4 | 4 | 5 |
| EFFECTIVENESS | 2 | 5 | 4 | 4 | 3 | 2 | 4 |
| TECHNOLOGY | Free software | | Free software | | Free software | | |

TABLE 2.1: Comparison of the different platforms studied

Wikipedia is also included in the table, although it is not a donation platform. In its case, the analysis is made regarding only Wikipedia's functionality for donations.

At first glance, this analysis may give the impression that there already exist a lot of alternatives for the Quartz system. However, this is not the case. None of the studied applications cover the intended scope for one or several reasons.

Platforms such as Ko-fi, Flattr, Giveth or GiveTrack are well built and usable and already have a decent base of users. However, they are all intended for different targets. Ko-fi and Flattr are targeted for artists and creators, Giveth is targeted for social good projects and GiveTrack is targeted for NGOs and non profit organizations. Furthermore, Ko-fi and Flattr only offer centralized solutions for the payments, while Giveth only offers decentralized ones.

Plaudit's target, on the other hand, is the academic community. Nonetheless, it is not focused on donations but on recognition and endorsements. Moreover, it is quite cumbersome and not easy to use.

Helperbit does not only have a different target, but is also closed for the time being.

Quartz proposes a solution that has not been covered yet by any of these platforms. It offers a donation platform for users to donate to academic journals and articles. For the payments, it allows both centralized and decentralized payments.

From the analysis it is clear that offering only decentralized options for the payments makes the applications less usable and effective. Most users are not familiar with this new technologies. For this reason, offering also centralized options seems like the optimal choice.

The analysis has also showed that it is essential for the donation button to be easy to find and it should stand out. It is also important that the interactions with the user are clear and the exchange of information is agile. An attractive design is also important. All these ideas will be taken into consideration when designing the system.

# Chapter 3

# Technologies for a distributed donation platform

The previous chapter, Chapter 2, was centered on the donation platforms and applications that populate the market, an how they are structured, whereas this chapter explores some decentralized technologies that can be used to create a new kind of decentralized donation platform.

## 3.1 Blockchain

Most decentralized technologies and systems rely on the blockchain technology as a basis. Therefore, giving a brief introduction to blockchain is key for giving context to the rest of technologies this chapter will explain.

### 3.1.1 Blockchain

*Blockchain* is a decentralized technology based on a data structure with the same name. The data structure, as the name itself indicates, is a chain of blocks [42].

The system was first introduced with the creation of Bitcoin [39]. It served as a basis for the creation of an electronic payment based on cryptographic proof, so that two parties or entities would be able to send money to each other without having to trust a central entity. The system is proven to be secure as long as the majority of the nodes involved in the system are independent and honest, so they control more CPU power than any hypothetical group of attackers or dishonest participants.

#### Basic concepts

A *transaction* is defined as an exchange of money between two participants. This is, a sender, a receiver and a quantity.

A *block* is a set of transactions. It is connected to the previous block and to the next one, thus forming a chain. A new block is added to the chain once enough transactions have happened.

The block also contains a timestamp and a link to the previous block (or parent), in the form of a hash of said block. It also contains a random number, called *nonce*. This number is important for the calculations of the block's hash when the next block is added to the chain.

The blockchain represents a complete register or ledger of the full history of transactions in the system, since for every new group of transactions, a block is added to the end of the chain. By cryptographic means, blocks can be validated as truthful part of the chain.

The immutability of the blocks is guaranteed by the hashes. If any data within the block (any information about any of the transactions) is modified, the hash of this new modified block will be completely different to the previous one.

## 3.2 Ethereum and ERC20 tokens

### 3.2.1 Ethereum

*Ethereum* [7] is a decentralized blockchain, and one of the main networks as for today. As bitcoin or other systems, it includes a coin, and mechanisms for performing transactions and payments. But the main interest of the Ethereum blockchain is the fact that it can run code, in the form of so called *smart contracts* .

A *smart contract* is a piece of code that can be executed in the Ethereum blockchain, autonomously. The language in which smart contracts are written in is *Solidity* [7].

The advantage of smart contracts is that, since they run over a blockchain, the trust in the contract relies in itself, rather that in a third party that warranties the compliance of the contract [43]. For this reason, it is completely impartial and everybody can trust it will be executed. Whatever rules are written in the contract will be executed and complied without a doubt.

### 3.2.2 ERC20 tokens

An *ERC20* [15] *token* is a smart contract running over the Ethereum blockchain containing code for the definition of a token or coin. This includes operations such as creating an initial amount of tokens, sending money between accounts or checking the balance of an account.

ERC20 is the standard that determines the functions and events this sort of contracts should implement.

The goal behind the creation of the ERC20 tokens is the standardization and compatibility among tokens in the Ethereum network, in order to facilitate interoperability and create a richer ecosystem.

*OpenZeppelin* [44] is a standard for building secure blockchain applications. They provide a library for secure development and deployment of smart contracts.

Among the contracts contained in this library, there are contracts for the deployment of ERC20 and ERC721 tokens.

## 3.3 Interledger Protocol

The Interledger Protocol (or ILP, for short) can be used in order to implement a system that provides both the donor and the receiver the freedom to choose their preferred currency or cryptocurrency, without having to worry about how the transactions will be performed.

### 3.3.1 Definitions

Interledger [56] is a protocol for sending and receiving payments between systems that maintain a register of financial transactions. Being a protocol means it is a set of rules that are publicly accessible.

Before understanding the idea behind Interledger, it is necessary to introduce two main concepts:

- **Ledger:** An archive or book storing economic transactions, columns for credit and debit, initial and final balance for an account. In other words, an account book.
- **Decentralized Ledger Technology:** Digitized and decentralized ledger. This is, decentralized database managed by several participants, without a central authority for verification.

So, in this sense, Interledger can be understood as a technology for interconnecting different decentralized ledgers. But this is a simplification.

Interledger can be defined as a system that allows communication among different payment systems, in order to perform a transaction of a certain value. The systems communicated by this protocol can (but do not need to) be decentralized ledgers. Ideally, any two payment systems (or ledgers), centralized or decentralized, can be connected via Interledger.

In a more technical sense, Interledger is an open protocol suite for the shipment of payments among different ledgers [27]. In analogy to internet routers, the **connectors** route money packages through independent payment networks. Since the architecture and the protocol are open, interoperability is allowed for any system.

The actors that participate in the system are [26]:

- **Sender:** The actor that sends the money.
- **Receiver:** The actor that receives the money.
- **Connector:** In case the sender and the receiver are not using the same paying system, they need intermediaries to connect them. So this is the role of the connectors: they forward money through the network from the sender all the way to the receiver. They may charge fees for the work. It is expected that the different connectors in the network compete among themselves in terms of velocity, reliability, coverage and price.

Any of these participants of the Interledger network is called a **node**.

The architecture of the Interledger network and how these nodes work together is further explained in chapter 6.

## 3.4 Web Monetization

Web Monetization [61] is an API that allows websites to request payments in small quantities via the browser and the Web Monetization provider.

A *Web Monetization provider* or *Web Monetization sender* is a digital entity that can issue payments in behalf of a user and, in particular, send micropayments to Web Monetization receivers [63] through ILP. In a similar way, a *Web Monetization receiver* is a digital entity that can accept payments performed via Interledger in behalf of a user and, in particular, receive micropayments through Web Monetization.

The Web Monetization receiver provides the user with a so called *payment pointer*, which is a URL assigned to a certain Interledger account. This URL is unique and, therefore, serves as an identifier of the account.

The idea behind Web Monetization is for the user running a website to embed its payment pointer in the website itself. As long as the users visiting the site have an application that allows them to send the micropayments (such as Coil - see Section 3.5.3), they will start sending little amounts of money to the payment pointer defined in the source code of the page.

The concept of *micropayments* is simple: small amounts of money sent every certain time. This is, for every unit of time (e.g. every hour or every thirty minutes), a small amount of money (e.g. a cent, two cents) is sent from the user visiting the page to the payment pointer registered in the code.

## 3.5 Some wallets and applications

In order to send or receive payments using ILP or Web Monetization it is necessary to use an ILP wallet. In the case of Web Monetization, some wallet providers can work as WM senders or receivers.

This section describes three applications that host ILP wallets: *Rafiki.money*, *Uphold* and *Coil*.

Both *Rafiki.Money* and *Uphold* can be used as standalone wallets and as Web Monetization receivers. In turn, *Coil* works as a Web Monetization provider.

### 3.5.1 Rafiki.money

*Rafiki.money* [47, 25] is an example Interledger wallet prototype developed by Interledger themselves. It allows the user to have an account with real money or test money using a testnet, and send and receive money.

It is *not* a donation tool. Similarly to Uphold (3.5.2), Rafiki provides the user with a payment pointer that can be used to receive donations (for example, the donations issued by Coil (3.5.3)).

It is not a real finished product, but a prototype for testing and experimenting. The repository itself indicates the project must not be used in production.

### 3.5.2 Uphold

*Uphold* [59] is a web exchange for buying and selling digital coins and cryptocurrencies. It allows the user to have an account with any type of coin and use it to issue and receive payments, as well as for buying and selling coins.

It is *not* a donation tool. But if a user wants to receive micropayments, for example, the ones that Coil issues, they must have a *payment pointer* to receive them. A payment pointer is the address of a wallet within a node of an ILP network [10]. Uphold provides the user with a payment pointer, so they can receive the donations.

### 3.5.3 Coil

*Coil* [9] is a browser extension and tool for micropayment streaming to web pages and creators.

A user can pay a subscription fee and download the extension. With the extension installed and active, every time the user visits a site that is webmonetized (i.e. supports Web Monetization) [61], Coil starts sending small amounts of money depending on the time the user expends in said page. In particular, Coil pays $0.36 for every hour a user expends on a page. The micropayments are sent roughly every second for each visiting member.

The main interest of this initiative is the fact that it is currency-agnostic. This means the user that makes the donations can be using any type of currency - being it any physical currency or a cryptocurrency. The receiver of the donations can also receive them in whichever currency they decide. Both currencies do not have to be the same, nor related whatsoever.

Coil is built over Web Monetization, which is the technology that manages the micropayments. It works as a Web Monetization Provider.

## 3.6 Other technologies

There are other technologies that have been used during the development of the project. The technologies explained in this Section are not the core of the project, but have been necessary for the implementation nonetheless.

### 3.6.1 Docker

*Docker* [12] is an open source software project that provides a system of software containers, in which the developers can deploy several applications to virtualize networks and systems.

In the context of this project, Docker is used to simulate a simple Interledger network. This network is explained in detail in Chapter 6 (see 6.2.2).

Docker containers are used to simulate an Ethereum testnet and three different Intereldger nodes, and then they are interconnected to simulate the whole three-node ILP network.

### 3.6.2 Node.js

*Node.js* [41] is a JavaScript runtime environment designed to create scalable network applications. It is open source and cross-platform. It is oriented to asynchronous events and it is used to write scripts for the server side in a client-server web application.

In the context of the project, Node.js is used to implement the servers to which the client connects in order to ask for the monetary transactions to be made.

### 3.6.3 React (javaScript)

*React* [48] is a JavaScript framework designed to help creating user interfaces. It facilitates the inclusion of HTML code within the JavaScript code, and it provides some other advantages, such as lambda expressions.

In the context of the project, it was used to implement the user interface and the client of the application.

### Material-ui

*Material-ui* [36] is a React library that provides components for developing a user interface following the material design principles.

# Chapter 4

# System overview

In this chapter, an overview of the system's functionality is given. The two actors that represent the main users of the system are introduced, followed by further explanation of how they interact with the system. Use cases and functionality are provided, along with screenshots of the actual system to illustrate the flow of the interactions. The goal of this chapter is giving a general high level overview of the system, so the architecture, design and implementation can be explained in detail in the next chapters (see chapters 5 and 6).

## 4.1 Quartz Ecosystem

For the purpose of this project, a system to perform donations have been designed. In a general sense, the application is basically a button than can be integrated within a journal's web page and allows the users to perform donations to the journal. The system contemplates different ways to perform the donations, through various payment systems. Part of the functionality relies on PayPal as a payment bridge, so it would require some of the users to have a PayPal account. In the same way, part of the functionality relies on Interledger as the payment bridge, so the users would be required to have Interledger accounts.

The system is designed for two main profiles: the journals and the donors. Each of these actors has a particular goal when using the system:

- For the journals, their goal is to be able to receive donations. In a more particular sense, they expect to be able to add the system to their website so visitors can perform donations, and then they want to receive the money in their account.
- For the donors, their goal is to support their favorite journals by sending them donations. In a more particular sense, they expect to visit the website of the journal and find a simple way to send money with their preferred currency of choice.

The interactions and workflow for both of these actors will be studied later in this chapter.

However, the system is more complex than just an application. It is composed by several different parts that together form what will be from now on called the *Quartz Ecosystem*. The application just described is an abstract description of the functionality, but more specifically, the system can be divided as follows.

### 4.1.1 Quartz button

The button represents the high level part of the system, and it mainly represents the functionality that has already been described. In short, it is a button that can be integrated in a web page and allows the visitor to perform donations using different methods.

The appearance of the button is showed in Figure 4.1.

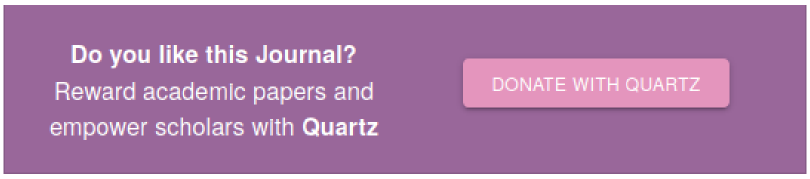

Do you like this Journal?
Reward academic papers and
empower scholars with Quartz
DONATE WITH QUARTZ

FIGURE 4.1: Quartz button

### 4.1.2 Quartz Token

The Quartz Token or coin (abbreviated QTZ) is an ERC20 token designed for an academic context.

It has no value as for today, but it is planned to work as an exchange currency for academic publication. For example, some of its applications would be: rewarding good reviews or papers, paying or getting discounts in the publication of papers or sending donations.

In the context of the project, the token was implemented and deployed to test the compatibility of Interledger with new tokens and coins, and study the advantages this provides. For this reason the payments in the test application are done in QTZ, while the receiver receives the corresponding amount in another currency (XRP, in this case).

### 4.1.3 Quartz ILP node

The Quartz node exists as part of the implementation, more than at a user level. Its existence should be transparent to the user, so it will be explained in more detail in the implementation chapter (Chapter 6).

Without diving into much detail, this node can be understood as the low-level entity that makes it possible to make transactions using the QTZ coin. It is an intermediate node in the network that manages the transactions (the ILP network).

The integration of the token within the ILP network allows any donations to be performed using QTZ. This means, for example, the donor can send any type of currency and the journal can receive QTZ (and vice versa). This also means a user with a Coil subscription payed in dollars can be sending microdonations that will be received by the journal in the form of QTZ.

### 4.1.4 Quartz wallet

It is the account that stores the user's QTZ funds. The wallet can be referenced using its address and it is necessary in order for the user to own QTZ, send payments using QTZ and receive QTZ.

### 4.1.5 Quartz Platform

The *Quartz Platform* is an internal structure that serves as an intermediary between the donors and the journals in the case of the PayPal donations (see Section 2.4.2). It works similarly to how a usual marketplace does: the journals register in the platform, so they no longer have to worry about the management of the donations. A button can then be created specifically for them. Then, when a donor sends money using the button, the platforms forwards it to the corresponding journal. This structure also offers other possibilities, such as periodic donations.

## 4.2 Payment methods

The system integrates with different payment systems, so to use some of the functionalities, users need to create accounts in them. Depending on the actor and the type of payment to be performed, one type of account or another (or even none) are needed.

For each actor, and for each type of payment, the requirements are as follows:

- **Donor:**
  - **Pay using PayPal:** In this case, a PayPal account is needed.
  - **Pay using credit card:** In this case, no accounts are needed. The user is just required to have an active credit or debit card.
  - **Pay with the Interledger button:** In this case, the user needs to have an Interledger account. In the proof of concept application, this account is supposed to exist within the Quartz system.
  - **Web Monetization micropayments:** In this case, the Coil[1] browser extension is used. For that reason, a Coil account is needed.
- **Journal:**
  - **Receive PayPal and credit card payments:** In these cases, a PayPal account is needed.
  - **Interledger and Web Monetization payments:** In these cases, the user needs to have an Interledger account, and its corresponding *payment pointer*. This account could exist within the Quartz system, or in some Web Monetization receiver, such as Uphold[2].

The creation of each type of account will now be briefly explained.

## 4.3 Creating the accounts

### 4.3.1 PayPal account

To create a PayPal account, the user should go to the PayPal website[3]. In the homepage there is a big button for creating an account.

There are two options for the accounts. In the case of the donor, the personal account is enough, for the donors are only supposed to send the donations. However, the journals need a business account in order to be part of the Quartz platform (integrating in the platform is a necessary step to be able to receive the donations). See Sections 2.4.2 and 4.1.5 for more detail on this.

[1] https://coil.com/
[2] https://uphold.com/
[3] https://www.paypal.com/es/home

To create an account, it is necessary to add in a name, an email, a password and a phone number. Then, it is possible to add in a credit card and verify the account. In the case of the business account, some information about the business itself is also required (such as business name and contact information).

After following the steps the sign in requires, the account should be created and ready to be used.

### 4.3.2 Coil account

To create a Coil account, the user should go to the Coil website[4]. In the home page there is a big button saying "Become a member".

To create the account it is necessary to enter an email and a password. Then, the email has to be verified (via a message sent from Coil to the address that was introduced in the form).

Once the account is created, the user can configure it by adding a credit card and a membership, as well as more personal information (such as a name or a profile picture).

### 4.3.3 Uphold account

To create an Uphold account, the user should go to the Uphold website[5]. In the homepage there is a button for creating an account.

There are two options for the accounts: an individual account and a business account. In this case, for both the donor and the journal the personal account is enough, although for the journal it makes more sense to create a business account.

To create an account, it is necessary to enter an email, password and country of residence. Then, it is also necessary no introduce some personal data, such as name, date of birth and a phone number.

After following the steps the sign in requires, the account should be created and ready to be used.

### 4.3.4 Quartz account

This project is still on an early state, so the Quartz accounts are registered manually through the command window. However, this process will be automatized in the future, so the user will be able to create an account via a graphic interface, as it is the case for the rest of the systems.

In the next sections, it is assumed that the actors already have a Quartz account. In the case of the donor, it is also assumed this account has some QTZ to pay with.

## 4.4 The interaction

The system is an intermediary for donations from donors to journals. So there is an interaction between the two actors, that is carried out through the application. A simple scheme of the interaction is shown in Figure 4.2.

[4]https://coil.com/
[5]https://uphold.com/es

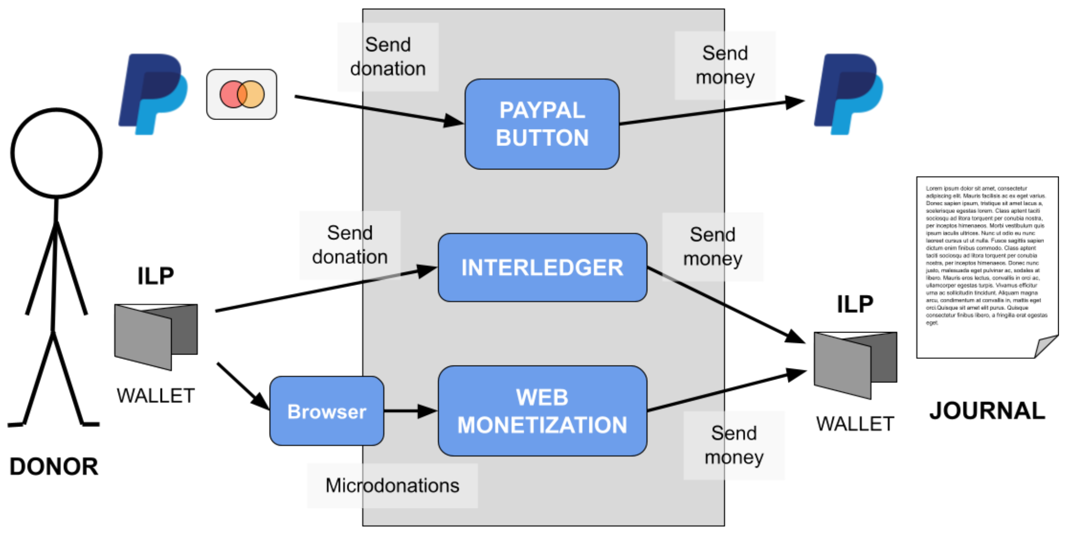


FIGURE 4.2: Interaction between a donor and a journal

As the diagram shows, the system offers essentially three payment options: a PayPal button, an Interledger button and the Web Monetization micropayment protocol.

The donor is the actor that actively interacts with the system to start the process of a donation. The donor can have a PayPal account, a credit card or an ILP wallet.

In the first two cases, they can choose the option of sending a donation using the PayPal button. Then, the system will make the necessary operations so that the money is finally sent to the journal's PayPal account.

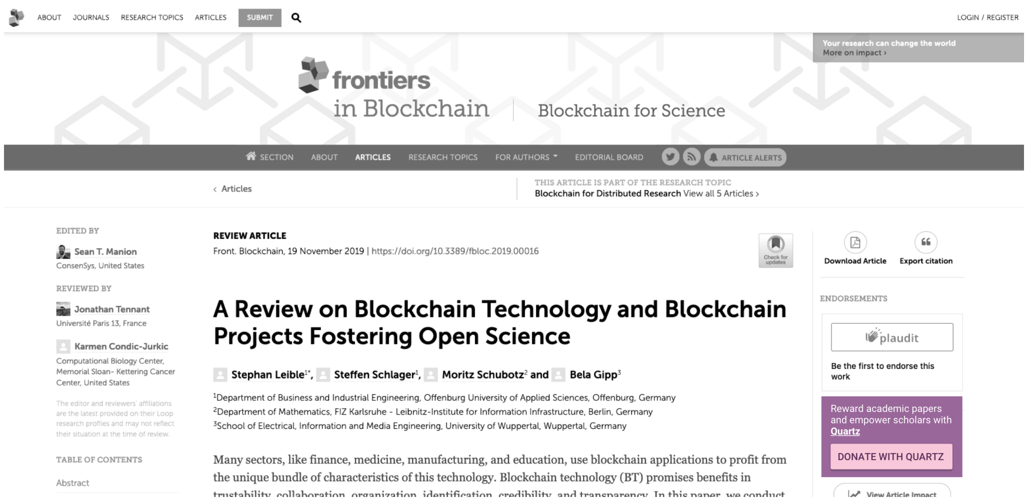
In the last case, by having an ILP wallet, the donor can use the Interledger button, so that the system can eventually send the money to the journal's ILP wallet. In addition to that, by using the Coil browser extension, the donor can also be making micropayments while visiting the web page of the journal. The system then sends the corresponding money to the journal's ILP wallet.

## 4.5 Donations

Once each user has the appropriate accounts created, the button can be installed in the website and the payments can begin to be made. In this stage, the use case for each type of user (or actor) is different, so they will now be explained separately.

### 4.5.1 Donor: Explicit donation

The donor could be any internet user reading academic papers online. When browsing the web page of a journal, or a certain article published by the journal, they encounter the *Quartz button* (Figure 4.3).



THIS ARTICLE IS PART OF THE RESEARCH TOPIC
Blockchain for Distributed Research View all 5 Articles ›

EDITED BY
Sean T. Manion
ConsenSys, United States

REVIEWED BY
Jonathan Tennant
Université Paris 13, France

Karmen Condic-Jurkic
Computational Biology Center, Memorial Sloan- Kettering Cancer Center, United States

The editor and reviewers' affiliations are the latest provided on their Loop research profiles and may not reflect their situation at the time of review.

REVIEW ARTICLE
Front. Blockchain, 19 November 2019 | https://doi.org/10.3389/fbloc.2019.00016

A Review on Blockchain Technology and Blockchain Projects Fostering Open Science

Stephan Leible[1*], Steffen Schlager[1], Moritz Schubotz[2] and Bela Gipp[3]

[1]Department of Business and Industrial Engineering, Offenburg University of Applied Sciences, Offenburg, Germany
[2]Department of Mathematics, FIZ Karlsruhe - Leibnitz-Institute for Information Infrastructure, Berlin, Germany
[3]School of Electrical, Information and Media Engineering, University of Wuppertal, Wuppertal, Germany

Many sectors, like finance, medicine, manufacturing, and education, use blockchain applications to profit from the unique bundle of characteristics of this technology. Blockchain technology (BT) promises benefits in trustability, collaboration, organization, identification, credibility, and transparency. In this paper, we conduct

ENDORSEMENTS
Be the first to endorse this work

Reward academic papers and empower scholars with Quartz

FIGURE 4.3: Quartz button integrated in a journal's web page

By interacting with the button, the application opens a new dialogue, where the user can select the amount of money to donate and decide the way in which they want to perform the payment (Figure 4.4):

- Paypal
- Debit or credit card
- Interledger

The first two options open a PayPal popup window, where the user can perform the payment.

The third option opens a dialog asking the user to write in their credentials for the Interledger account (see Figure 4.5). For the proof of concept purpose of this project, all the operations are performed over predefined accounts in a test network, so for now this is still a placeholder dialog.

If the payment is performed correctly, the application shows a dialogue confirming everything worked properly and giving the user the option to visit the Quartz website[6] and join the community. If there is any error during the process, the application shows a dialogue stating there has been an error (Figure 4.6).

There is also the Web Monetization payments, that are micropayments sent to the journal for each time unit a reader expends in the page. This process is a little different, so it will be explained in the next section (Section 4.5.2).

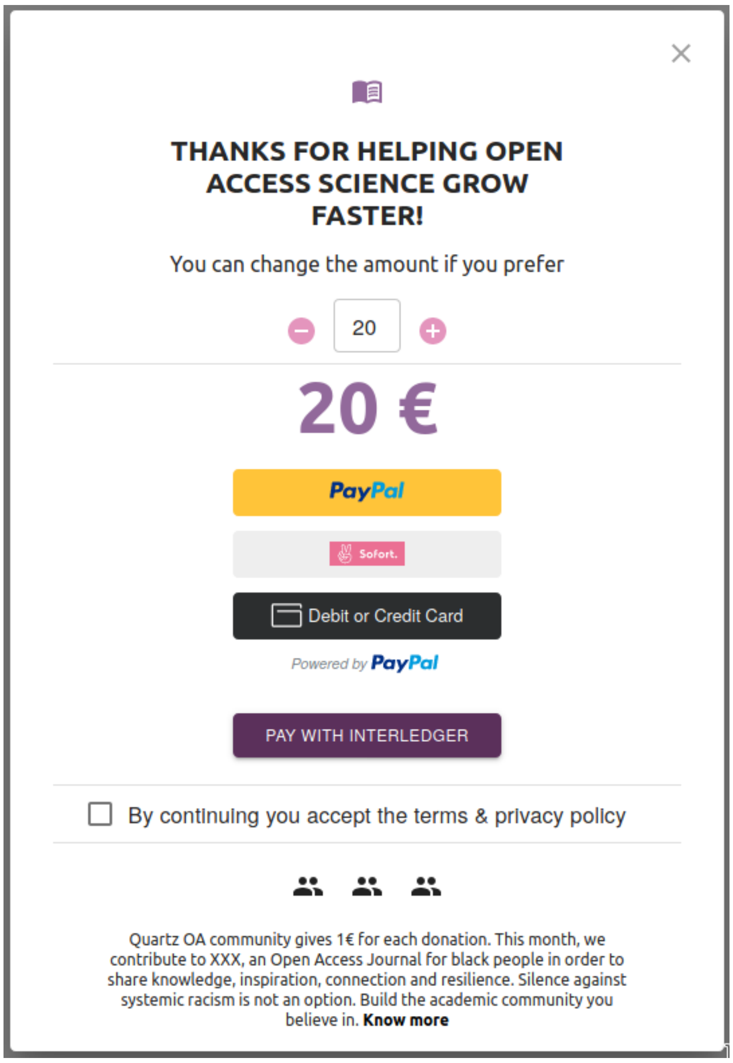
### 4.5.2 Donor: Indirect donation (micropayments)

One of the ways the donor can donate to the journal is via microdonatinos.

In the case of the journal, the process for integrating the system into their web page is the same that was already explained in Section 4.5.3. When the button is integrated in the page, it makes the proper changes so that the website becomes webmonetized and can start receiving microdonations.

In the case of the donor, the process is different to the one explained in the previous section.

[6]https://quartz.to/


THANKS FOR HELPING OPEN ACCESS SCIENCE GROW FASTER!

You can change the amount if you prefer

20

20 €

PayPal

Sofort.

Debit or Credit Card

Powered by PayPal

PAY WITH INTERLEDGER

☐ By continuing you accept the terms & privacy policy

Quartz OA community gives 1€ for each donation. This month, we contribute to XXX, an Open Access Journal for black people in order to share knowledge, inspiration, connection and resilience. Silence against systemic racism is not an option. Build the academic community you believe in. **Know more**

FIGURE 4.4: Payment dialogue

First, it is required that the user has a Coil account (see 4.3.2). Then, they have to download the Coil browser extension. This extension is the one that allows the user to send microdonations to the webs they visit. In the Coil account, they have to select a configure a membership. This is, a monthly fee that is payed to Coil and then redistributed among the webmonetized websites the user visits.

Once the membership is configured and the extension is downloaded, the donor just has to log in to Coil and then, whenever they visit a webmonetized website - the journal's website, in this instance - the extension will indicate that it is sending money and the micropayments will be performed. Coil will pay $0.36 for each hour the donor expends in the journal's website.

Figure 4.7 shows the browser extension before logging in and after logging in, when visiting a website that is webmonetized. Figure 4.8 shows the extension in context, in a website that has Web Monetization enabled.

### 4.5.3 Journals

The use case for the journals is the same for both implicit and explicit donations.

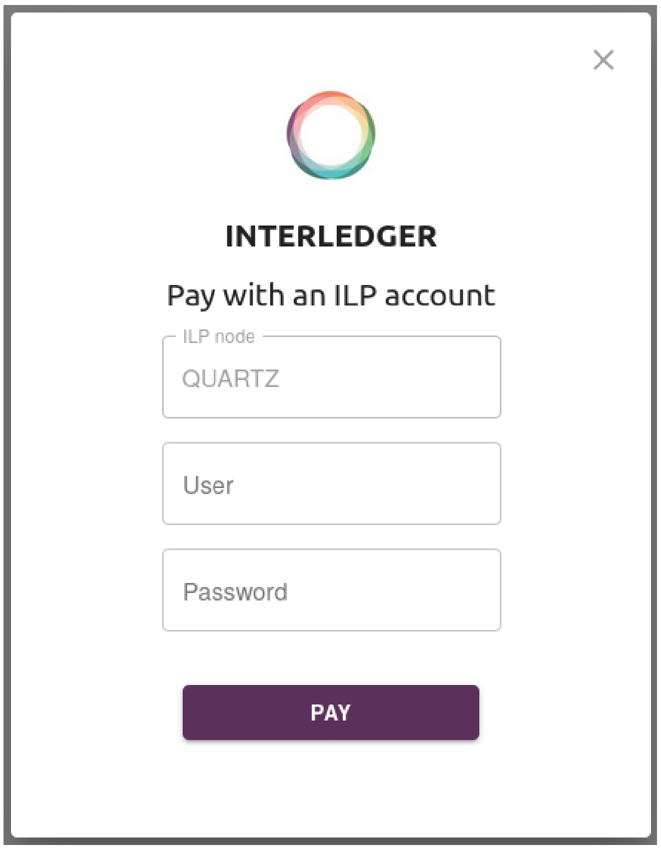


FIGURE 4.5: Interledger payment screen

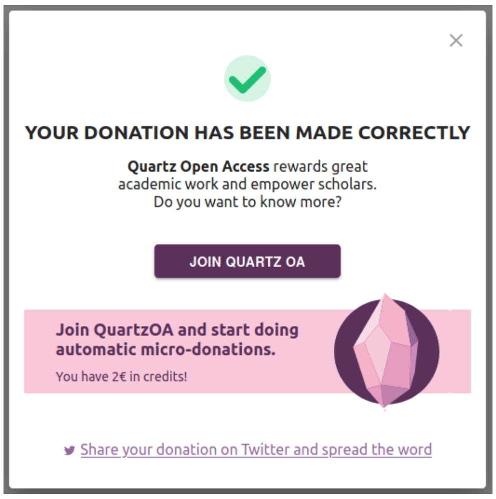
The user representing the journal could be a journal editor or the person in charge of the journal's web page. Their goal is to integrate the system in their page and start receiving donations.

In order to integrate the full system, they need to have one or more accounts in the various technologies used (although if they prefer to use only one of them, the code can be easily adjusted). This means the journal needs to have a PayPal account, and use this account to register in the Quartz PayPal platform. This is needed for the PayPal and credit card payments. The journal also has to create an account in an Interledger node, that could be the Quartz Interledger node. This way they obtain a payment pointer, that is used as an address to send them the money.

Once all the accounts are created, the button can be integrated in the web page[7], as shown in Figure 4.3.

When the donations are made by users visiting the website and interacting with the buttons, the journal receive the money in the corresponding account. The money sent via the PayPal buttons (PayPal, Sofort, and credit and debit card) is received in the Journal's PayPal account. The balance can be checked in their PayPal profile. The payments received via Interledger are received in the journal's Interledger wallet. This wallet could be in any node providing payment pointers.

[7]The button is implemented in React (JavaScript, HTML) and it can be integrated by inserting the button's code in the code of the website.


YOUR DONATION HAS BEEN MADE CORRECTLY

**Quartz Open Access** rewards great academic work and empower scholars. Do you want to know more?

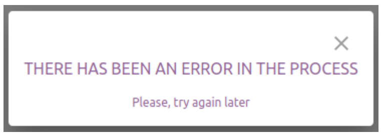
JOIN QUARTZ OA

**Join QuartzOA and start doing automatic micro-donations.**

You have 2€ in credits!

Share your donation on Twitter and spread the word

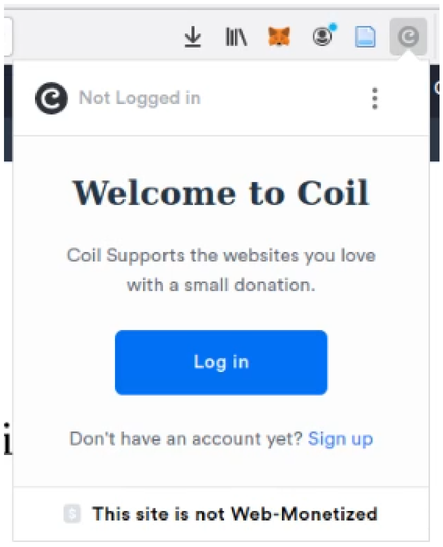
(A) Success dialogue


THERE HAS BEEN AN ERROR IN THE PROCESS

Please, try again later

(B) Error dialogue

FIGURE 4.6: Success and error dialogues


Not Logged in

**Welcome to Coil**

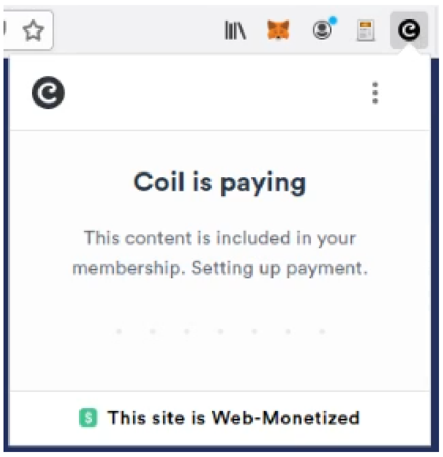
Coil Supports the websites you love with a small donation.

Log in

Don't have an account yet? Sign up

This site is not Web-Monetized

(A) Coil extension before logging in.

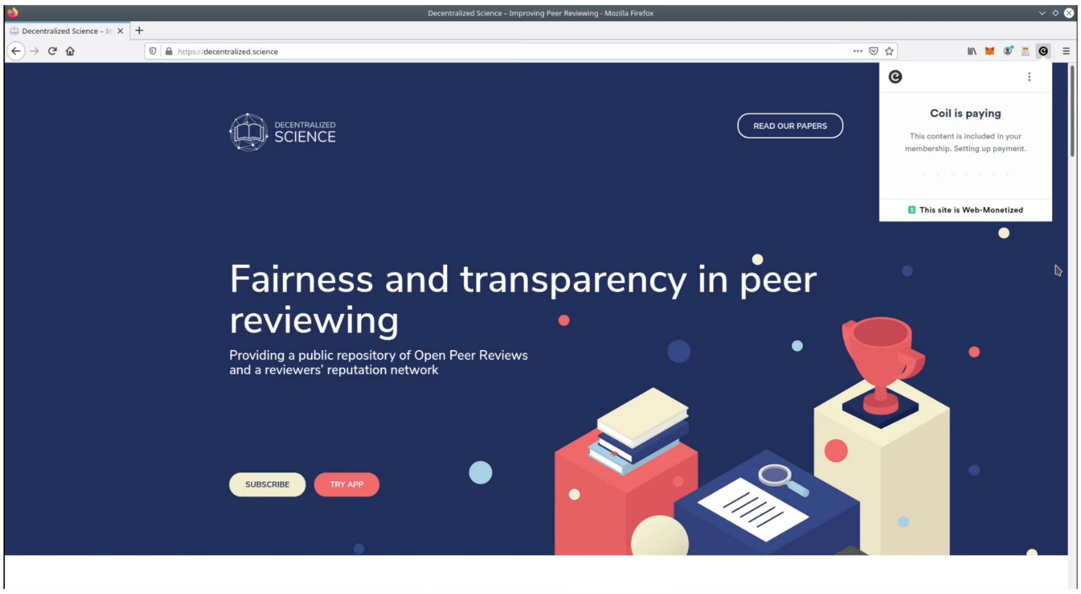

**Coil is paying**

This content is included in your membership. Setting up payment.

This site is Web-Monetized

(B) Coil extension while visiting a webmonetized site.

FIGURE 4.7: Success and error dialogues



Coil is paying

This content is included in your membership. Setting up payment.

This site is Web-Monetized

Fairness and transparency in peer reviewing

Providing a public repository of Open Peer Reviews and a reviewers' reputation network

Decentralized Science – Improving Peer Reviewing - Mozilla Firefox
Decentralized Science – In
https://decentralized.science
DECENTRALIZED SCIENCE
READ OUR PAPERS
SUBSCRIBE
TRY APP

FIGURE 4.8: Using the Coil extension in a webmonetized website.

# Chapter 5

# Centralized Implementation

In this chapter, details about the centralized implementation of the system are given. First, the core concepts of a PayPal application are introduced, as well as the different functionalities, components and APIs. Then, the architecture of the system is explained, followed by a detailed description of the implementation.

The centralized implementation is relatively simple, since all the management of the transactions is delegated to PayPal, who provides a high-level API. For this reason, the implementation of the centralized subsystem consists mainly in the integration with said API.

## 5.1 Introduction

Chapter 4 introduced the Quartz ecosystem, in terms of its external structure and how the user interacts with the different functionalities.

This chapter will study the centralized part of said ecosystem, by explaining how it is implemented.

The main component of this subsystem is the donation button, and the underlying payment system uses PayPal's infrastructure to send the payments. The user can choose to make the payments using a PayPal account or by credit card, but the differences between both channels are made transparent for both the user and the developer thanks to the simplicity of PayPal's API.

For this reason, in order to explain how this subsystem is implemented, it is helpful to understand the technology that is used for the implementation. This is the PayPal REST API for marketplaces and platforms. The implementation of the centralized donations consists in an integration of Quartz's UI with the functionalities PayPal provides.

## 5.2 PayPal

For the creation of the first version of the system, a centralized approach was taken and PayPal was used to allow payments and donations in a centralized way. PayPal was chosen for its interoperability and the functionalities it provides for the creation of marketplaces and platforms.

In brief, a platform can *onboard* a set of sellers to be part of the marketplace and then it can manage how the money is forwarded to said sellers.

PayPal offers a REST API to perform the different operations for the development. The REST API requests are executed combining an HTTP method, such as GET, POST, PUT, PATCH or DELETE, the URL to the API service, the URI of the resource to be updated, deleted or queried and optional HTTP headers. PayPal provides the developer with credentials to make the API calls, and the authorization is performed using the OAuth 2.0 protocol. These credentials consist of a **client-id** and a **secret**.

### 5.2.1 Concepts

There are a few concepts that will be used throughout this chapter and are particular to the PayPal API that has been used for the development of the centralized subsystem. Some of those concepts are the following:

- **Platform/marketplace.** A structure used for organizing a group of *vendors* (in this case, the journals) so that the payments made by the *buyers* (in this case, the donors) can be easily managed. It will be explained in more detail in Section 5.2.2.
- **Client-id and secret key.** These are credentials that identify a particular PayPal App. A user with a PayPal developer account can create apps that provide them with credentials to access the REST API for both testing and live transactions [35]. The *client-id* is the identifier and the *secret key* is the password. These credentials vary for each app and are also different for live and sandbox apps.
- **Access token.** It is an identifier or key that authorizes the user to use the PayPal REST API server [20]. It can be obtained by using the *client-id* and the *secret key*.
- **Attribution-id.** This key identifies the user as a *PayPal partner*. PayPal partners are developers that are allowed by PayPal to use all the functionalities the PayPal REST API provides. This attribution-id is received directly from PayPal and has to be used as an identifier every time a call to the API is made [2].

### 5.2.2 Platforms and marketplaces

PayPal offers an API for marketplaces and platforms.

A platform, as previously explained in Section 2.4.2, acts as an intermediate entity between the donor and the journal. A journal that wants to participate in a platform should, first, have a PayPal business account.

Next, by making calls to the PayPal API, using the Authorization token and the Attribution ID, it is possible to generate an *onboarding* link, which is specific for the platform. Onboarding is the process by which a journal, in this instance, integrates into a platform, so the platform can manage payments donors send to the journal. By using said link, any journal can onboard.

Figure 5.1 shows a simplified diagram of the structure of a platform.

## 5.3 Architecture

The architecture of the system has several layers (as illustrated in Figure 5.2). The user interface is common to both the centralized and decentralized part of the system,

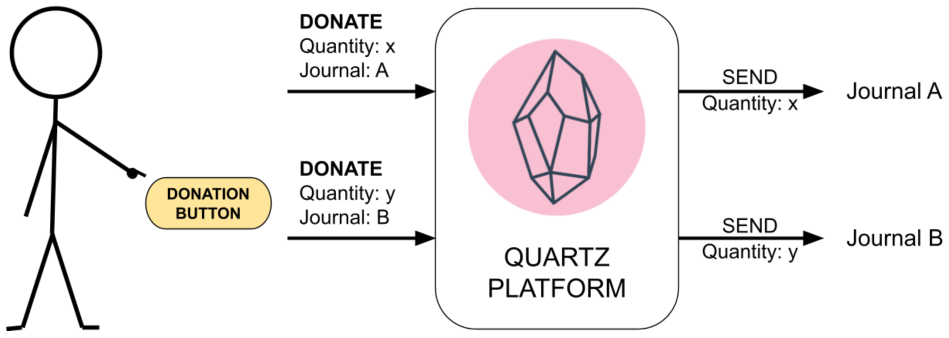


FIGURE 5.1: Quartz Platform

and then each of the subsystems has its own lower layers, in both cases based on a server-client model.

In this section, the centralized part will be analyzed.

As the Figure shows, the most external layer is the UI (user interface). It is the layer that allows the communication between the user and the application. This layer was implemented using React JS and following the directives of material design. In particular, the library material-ui for React was used.

This layer connects with the clients for both the centralized and the decentralized subsystems. In the case of the centralized system, the client receives the data inputs that the user enters via the UI and makes the POST requests to the server. The client is also implemented using React JS.

The server makes calls to the PayPal API using the information received from the client. The code for the server is implemented using the *Express framework* [13] for JavaScript and it runs on Node.js.

Thereby, these underlying client and server layers follow a theoretical structure, instead of a physical one. In particular, they follow the client-server architectural pattern [49]. As such, the *Centralized System Client Controller* works as an intermediate layer between the *Client UI* and the *Server*, by parsing the data received from the UI and forwarding the subsequent petitions to the server. The *Server*, on the other hand, manages the petitions and makes the necessary calls to the corresponding API (PayPal API, in this instance).

It is important to note that, for this is a theoretical structure, both the client and the server can be running in the same machine. The codes for both layers run independently, but there is no need for them to exist in separate machines. This is the case in the current deployment where, for the sake of facilitating testing, both structures are running in the same machine. However, the current implementation also allows for deployment in different machines. This separation would be ideal, since some of

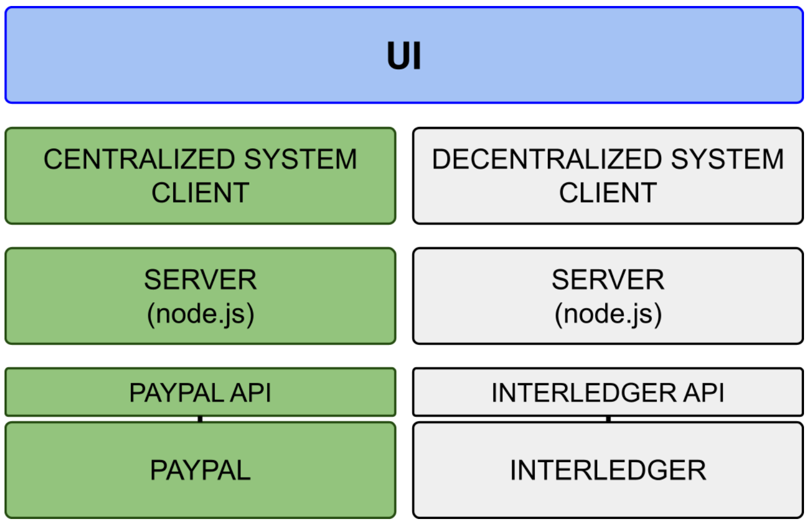


FIGURE 5.2: Architecture of the system. Highlighted the centralized part.

the data being managed by the application is sensitive information, and it is safer to process such data in an independent server.

It is also important to note that the Client Controller and Server for the centralized implementation are different from the ones for the decentralized implementation. The source code is different and independent, and they both run and work without any dependence from one another.

The PayPal API is the access layer through which the PayPal functionalities are accessed. The PayPal system is the deepest layer of the system, and it is the one in charge of managing the shipment of the money.

With this general knowledge of the architecture, it is now possible to explain in detail the sequence of calls among the different layers of the system.

### 5.3.1 Sequence diagram

The PayPal donation button generated for the sellers works following a server-client structure.

The server listens to petitions regarding the orders and makes the API requests. The client reads and manages the options of the user's order (in this case, a donation) and forwards the necessary information to the server to perform the calls.

Figure 5.3 shows a sequence diagram illustrating the flow for a transaction through the PayPal donation button. This applies for both payments using a PayPal account or a credit card. The flow for a payment would be as follows:

- The user sets the quantity they want to donate and press the "Pay" button.
- The client sends a *create-order* petition containing the quantity to be donated.
- The server listens to the petition and makes a call to the PayPal API. The access token and attribution ID of the platform are sent with the call, to authenticate the platform and assure it has the necessary permissions.

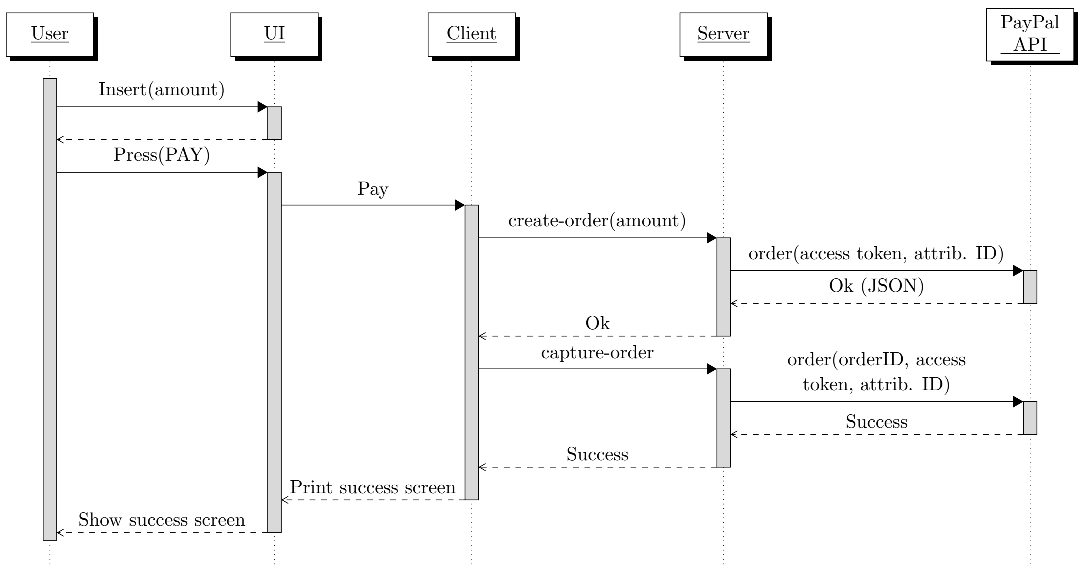


FIGURE 5.3: Sequence diagram for the PayPal donation button

- Through the API, PayPal does the corresponding operations to create the order, and returns a JSON containing either an error, if there is any, or some information about the operation.
- The client receives the response and prints in the prompt the error or the data received. Among this data, there is an order ID, identifying the order just created.
- If there is no error, the client sends a *capture-order* petition.
- The server listens to the petition and makes a call to the PayPal API. The access token and attribution ID of the platform are sent with the call, along with the order ID, that identifies the order.
- Through the API, PayPal does the corresponding operations to perform the order and returns whether there is an error or it has been a success.
- The server returns a state of success if everything worked properly or one of error if something went wrong.
- The client receives the response and fixes the system's state to either success or error, so the front-end knows which screen to show the user.
- The front-end prints the corresponding screen: a success screen if everything worked properly and an error screen if something went wrong.

## 5.4 Conclusions

The implementation of a centralized solution was the first approach of the project. It is an interesting approach, since centralized payments are most common nowadays in the context of online donations and payments in general. For this reason, they do not pose a big challenge for the users to overcome, since they are already used to this kind of systems (unlike newer or more innovative payment systems, where adaptation could be harder).

Another reason for this approach to be taken first was the fact that there is plenty of information, documentation and support for the development of systems that use these technologies (such as PayPal), for they are very popular and have a big user and developer base. For this reason, it is easier to create a first Minimum Viable Product (or MVP) using one of these technologies, instead of a decentralized one.

However, the centralized solutions also have some disadvantages. For example, due to its centralization, PayPal can ultimately decide who can integrate with their system. This poses several problems when trying to move the Quartz centralized system from a test environment into production, because to get the permissions needed to do the migration, PayPal imposes very strict criteria that is almost impossible to fulfill[1].

On the other hand, removing the centralized entity that controls the transactions, would make them more transparent and fair, and would also lower the costs, by lowering the number of intermediaries. Moreover, a system allowing payments and donations no matter what monetary systems the senders and the receivers use would help with the inclusion of any type of collectives.

All these goals can be achieved by implementing a decentralized system with technologies such as Interledger or Web Monetization. The next chapter is centered around the implementation of a system with these features.

[1]For example, a company needs to generate more than $1M per year in order to be granted access to the Live API.

# Chapter 6

# Decentralized Implementation

In this chapter, details about the decentralized implementation of the system are given. First, the Interledger Protocol is explained in detail, introducing its architecture and advantages. Then, the ERC20 tokens are defined and OpenZeppelin is introduced as a tool for creating such tokens. After that, the architecture of the system is explained, followed by a more detailed description of the implementation.

## 6.1 Introduction

Chapter 4 introduced the Quartz ecosystem, in terms of its external structure and how the user interacts with the different functionalities.

This chapter will study the decentralized part of said ecosystem, by explaining some of its main components and their implementation.

In short, the main parts of this ecosystem are:

- **The Quartz Button**, used to send direct donations to the journal it belongs to. In terms of structure, the button calls the underlying system -in this case, the ILP network -, and orders it to perform the needed transaction.
- **The Quartz Token** or coin, abbreviated as QTZ. It is an ERC20 token that can be sent or received in the donations, and it is meant to exist as an exchange token for academic publishing processes.
- **The Quartz ILP node**, a node inside the Interledger network that serves as a connector for exchanging QTZ with other currencies. It also stores users' wallets. It is connected to an Ethereum network, inside which the contract for the QTZ coin is deployed.
- **The Quartz wallet**, that lives inside the *Quartz node*, and stores a user's funds in QTZ (the Quartz coin).

In order to explain how these components are implemented, it is key to first understand the technologies that are used for the implementation. These are *Interledger* and *Web Monetization*. The implementation of the Quartz ecosystem consists in subtle but precise changes in the structure of these technologies, so introducing them in detail is imperative.

## 6.2 Interledger Protocol

The *Interledger Protocol* [27] is a set of publicly accessible rules for sending and receiving payments between different payment systems or ledgers. The goal is to allow the exchange of money among systems that have little to nothing in common, apart from the fact they are ledgers. They could be centralized or decentralized and operate with very different currencies.

In a general sense, this protocol works in a similar way the internet does. When sending a package through the internet, different interconnected routers find the best path to forward the package from the sender to the receiver. In a similar way, in Interledger there are intermediate nodes (the *connectors*) that find the best route (in terms of connection and price) to forward a certain quantity of money from a sender - operating in a currency A in A's ledger - to a receiver - operating in a currency B in B's corresponding ledger. The connectors have to find a way in which one currency can be transformed into the other and the money can reach the destination ledger, all while not exceeding a certain cost in transactions.

This technology is very interesting for a system that aims to allow donations to different collectives from very diverse backgrounds. In this sense, a technology as Interledger makes it easy to send donations from one country to others, without having to worry about abusive fees, nor dependence of powerful intermediaries. It also brings the possibility of the integration with new tokens or currencies, which is specially interesting in the context of academic donations, where some argue that monetary rewards could be harmful for the field, while other kind of rewards would be more effective [17].

In order to integrate with this technology, it is essential to understand how it works with detail. This section explains the Interledger protocol, so the next sections can analyze how it is used in the Quartz application.

### 6.2.1 Concepts

Before explaining the Interledger architecture in detail, it is necessary to define some concepts that are a core part of the protocol.

The protocol is centered around money shipment. The actor (person or entity) sending the money is called the *sender*, and the one receiving it is called the *receiver*. If the sender and the receiver are in the same monetary system, the shipment of the money is straightforward. However, if they are not, in order for the money to reach the receiver, a route has to be found between both ends. The intermediate parties that forward the money until it reaches the receiver are called *connectors*. As a general term, the word *node* is used to name any of these components of the network (a sender, a receiver or a connector). A simple graph of nodes can be seen in Figure 6.1.

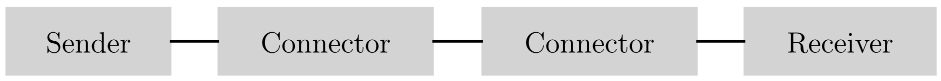


FIGURE 6.1: Node scheme in an Interledger Network [26]

Every node within the Interledger network can be referenced using an *Interledger address* or *ILP address*. They provide a universal way to address the nodes, independently of their internal nature. These addresses are strings composed by dot-separated segments that are hierarchically ordered. This means, the first segment is the most significant and the last one, the least significant.

In the frame of the Interledger protocol, connectors have *peers*, which are other connectors they consider trustworthy and transact with. A connector can fix a maximum credit limit for its peers. As the connector forwards packets sent by its peer, the liabilities of the peer to the connector accrue, and they could potentially exceed the credit limit, so the connector could reject subsequent packets.

For the connector to be able to send new packets, the peer has to settle their liabilities. To do so, they can send a payment to the connector using a settlement system they both have agreed on using. So, in this context, a *settlement* is the irrevocable discharge of a liability via an unconditional transfer of an asset or financial instrument [28]. These settlements take place in a *settlement system* (such as a bank, or a decentralized ledger or cryptocurrency) through which the funds are exchanged.

Now it is possible to define the *settlement engines*, which are services that are operated by two Interledger connectors that are peers and allow them to send and receive settlements from each other. A connector can have several peers that settle over the same system, and the settlement engine can manage multiple accounts, each one connecting to a different peer.

### 6.2.2 Architecture

In this section, the structure and architecture of an Interledger network will be studied. To simplify the explanation, a simple network with only three nodes - a sender, a receiver and one connector - will be used. The architecture explained for this simple network can be easily extended to a larger network with more nodes.

This simple network is composed by three nodes:

- **Alice**, the *sender*, that has an account with Ether in an Ethereum network.
- **Charlie**, the *receiver*, that has an account with XRP in an XRP network.
- **Bob**, a connector that has settlement engines for both Ethereum and XRP settlements.

The structure of the network is illustrated in Figure 6.2.

As the Figure shows, the nodes include different accounts and are interconnected in a specific way:

- **Alice node:** It is composed by the node itself and the Ethereum settlement engine. It contains Alice's account, and the Alice $\leftrightarrow$ Bob exchange account.
- **Bob node:** It is composed by the node itself, the Ethereum settlement engine and the XRP settlement engine. It contains the Alice $\leftrightarrow$ Bob exchange account and the Bob $\leftrightarrow$ Charlie exchange account.
- **Charlie node:** It is composed by the node itself and the XRP settlement engine. It contains Charlie's account and the Bob $\leftrightarrow$ Charlie exchange account.

### 6.2.3 Flow

In order to better understand the Interledger architecture, it is necessary to study the flow of the payments in the network. Figure 6.3 sows a sequence diagram of a payment performed in the three node network defined in the previous section (Section 6.2.2). The actors in the diagram are the nodes (Alice, Bob and Charlie) and the settlement engines and exchange rate provider that the connector interacts with.

Alice has a certain quantity of ETH in her Ethereum account, and also has an Interledger wallet associated to said account. Alice wants to send an amount of money to Charlie, who has an XRP account. Since they both work with different payment

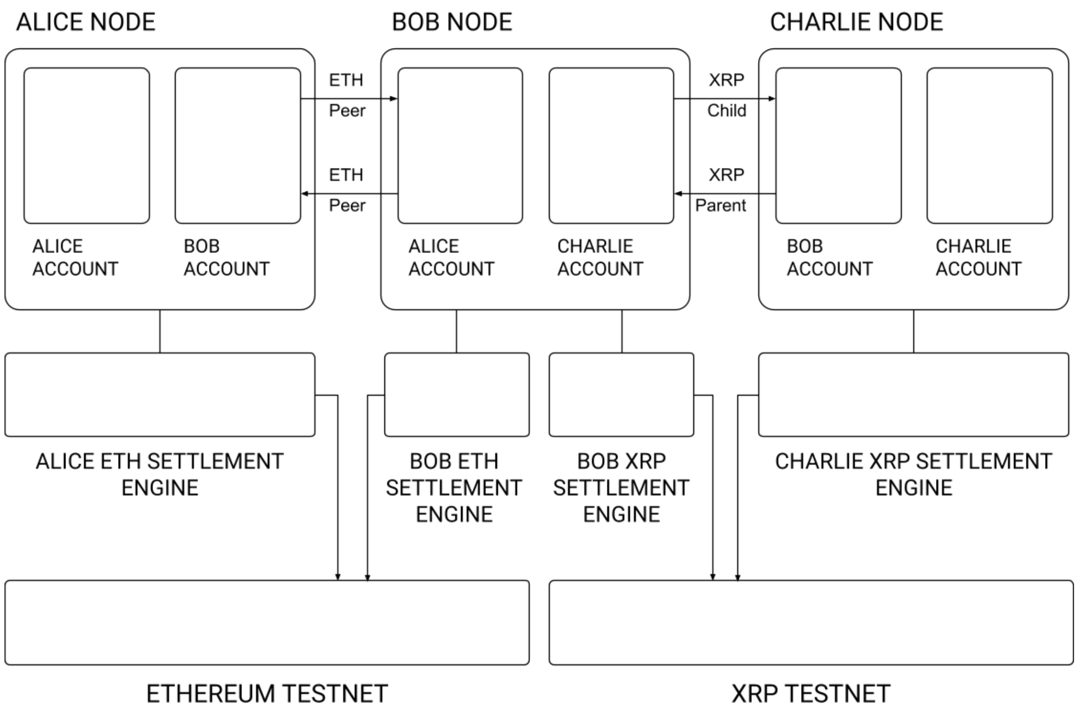


FIGURE 6.2: Simple Interledger network

systems, Interledger needs to find a route to send the money from one to the other. The flow for the payment in the Interledger network is as follows:

- Alice sends a *Prepare* packet, that represents a certain amount of money and has an associated *condition*, that has to be *fulfilled* in order for the payment to take place. Alice sends said packet to Bob, a connector with an Ethereum settlement engine who Alice can settle with. The packet is sent using the crossed accounts Alice and Bob have in each other's node.
- Bob, that has settlement engines for ETH and XRP, can settle with both Alice and Charlie. Bob calculates the exchange rate (in this example, consulting Coinbase[1]). It also decides on the rates it wants to charge (in this case, 0).
- Bob forwards the *Prepare* packet to Charlie, using the crossed accounts Bob and Charlie have in each other's node.
- Charlie gets the *Prepare* packet and evaluates if the quantity to receive (once the rates are applied) is correct.
- If he thinks it is correct, he *fulfills* the *condition*, by sending back a return *Fulfill* packet.
- The packet is sent from Charlie to Bob, and Bob confirms the change of balance in Alice's and Charlie's accounts, by using the corresponding settlement engine.

  If there were more than one connector, these confirmations would be done in order from last to first. This is, first the payment from the last connector to Charlie would be confirmed, then from the second to last connector to the last one, and so on until reaching the first connector, which would settle with Alice, thus ending the process. Since these settlements were approved when forwarding the *Prepare* packet, they are assured to take place when the *Fulfill* packet is being returned.

[1] https://www.coinbase.com/

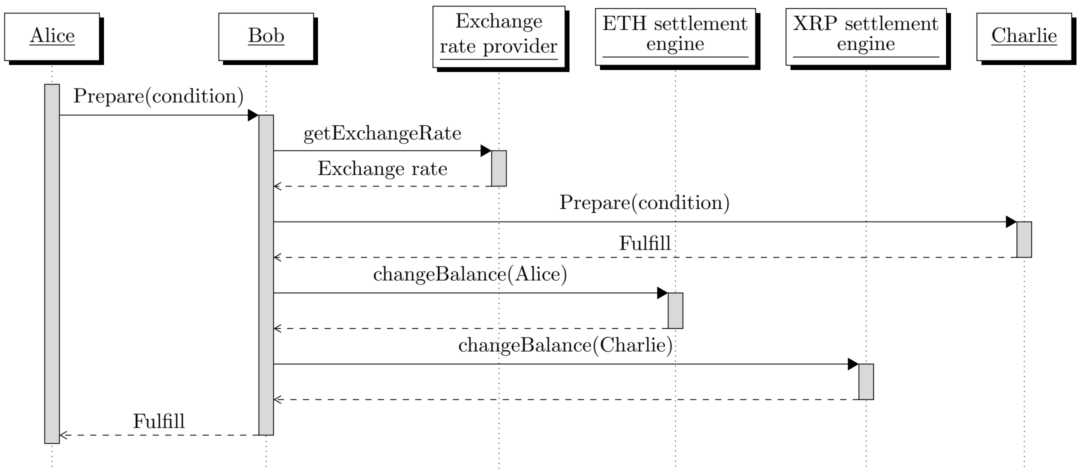
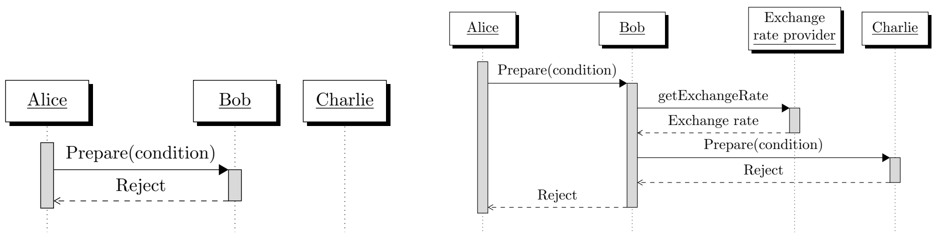


FIGURE 6.3: Sequence diagram of a payment in the Interledger network

- Finally, the *Fulfill* packet reaches Alice's node, thus confirming the payment has been made correctly.
- If Charlie does not want to receive the money, or if any connector can not or does not want to forward it, a *Reject* packet is forwarded back, until it reaches Alice.
- If for some reason the *condition* is never fulfilled nor the packet rejected, the payment *expires* after some time.

Figure 6.4 shows the sequence diagrams for the cases in which the packet is reject by either the connector or the receiver.



(A) The connector rejects the packet (B) The receiver rejects the packet

FIGURE 6.4: Sequence diagrams of a payment that is rejected

## 6.3 Web Monetization

Web Monetization[61] is an API that allows websites to request payments in small quantities via the browser and the Web Monetization provider.

This technology works over an Interledger network, which is the entity in charge of issuing the payments. Web Monetization manages the shipment of small payments (or micropayments) over periods of time.

The structure of an application using Web Monetization is showed in Figure 6.5.

As the figure shows, the central node in the structure is the website in which Web Monetization is enabled.

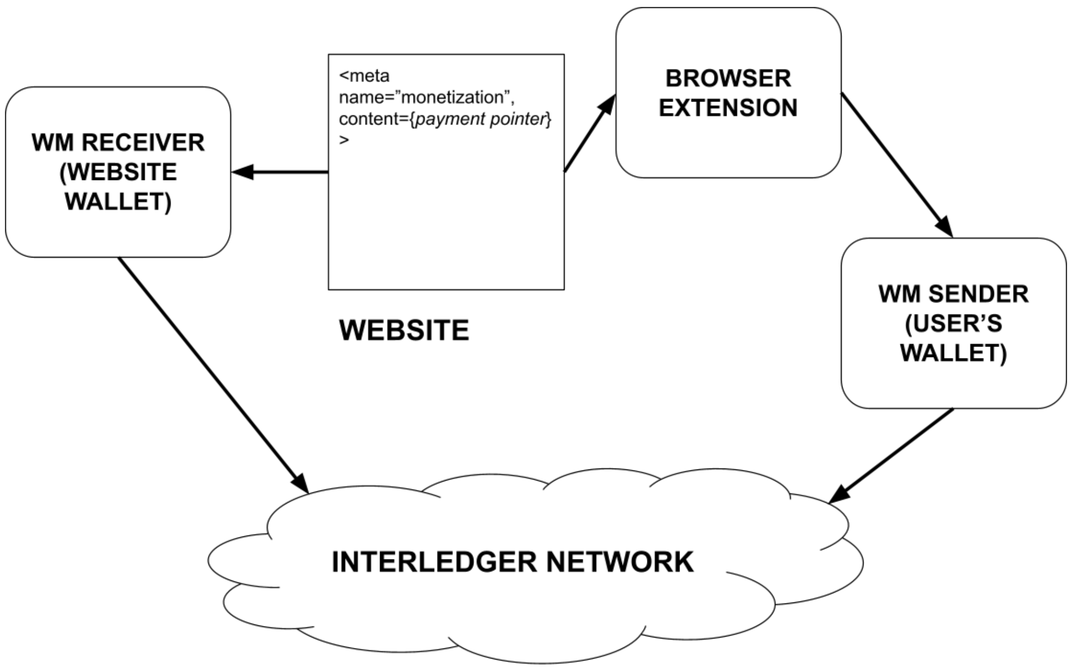

WM RECEIVER (WEBSITE WALLET)
<meta name="monetization", content={payment pointer} >
WEBSITE
BROWSER EXTENSION
WM SENDER (USER'S WALLET)
INTERLEDGER NETWORK

FIGURE 6.5: Web Monetization scheme

The website specifies in a meta tag in its html code that it will allow Web Monetization payments and the payment pointer that will receive said payments. This payment pointer points to the web's wallet, which is provided by a *Web Monetization receiver*. This receiver is an entity that can receive Interledger payments in behalf of the user. In this particular case, the Interledger payments are sent using Web Monetization.

This receiver is connected to the ILP network, so it is capable of receiving payments sent by any other entity in the network.

On the other side of the transaction is the user that visits the page. The user does so from a certain browser that has a browser extension installed which allows to send Web Monetization payments. This browser extension in connected to the user's wallet, which is provided by a *Web Monetizaiton sender* or *provider*. This sender is an entity that can send Interledger payments in behalf of the user. In this particular case, the Interledger payments are sent using Web Monetization.

As the receiver, this sender is also connected to the ILP network, so it is capable of sending payments to any other entity in the network.

When the user enters a website that is webmonetized, the browser, through the browser extension, identifies the meta tag that indicates that the website can receive Web Monetization payments and reads the payment pointer to which the payments need to be sent. The extension uses this information to make the WM sender issue a payment to that payment pointer.

### 6.3.1 Flow

This process of sending Web Monetization payments can be better understood by studying the flow of the process step by step. Figure 6.6 shows a sequence diagram of

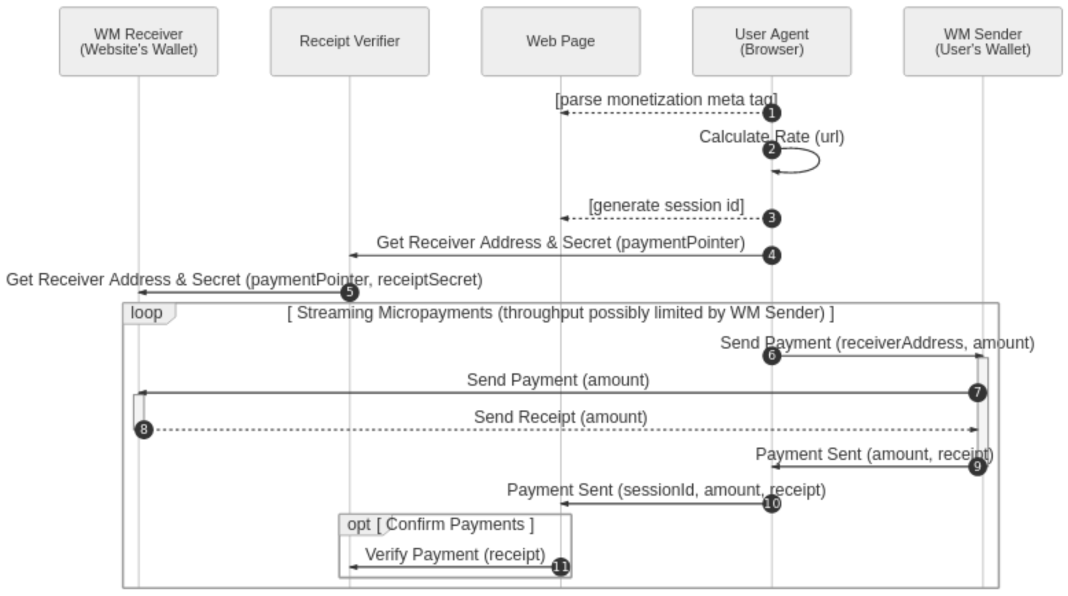

WM Receiver (Website's Wallet)
Receipt Verifier
Web Page
User Agent (Browser)
WM Sender (User's Wallet)
[parse monetization meta tag]
Calculate Rate (url)
[generate session id]
Get Receiver Address & Secret (paymentPointer)
Get Receiver Address & Secret (paymentPointer, receiptSecret)
loop
[ Streaming Micropayments (throughput possibly limited by WM Sender) ]
Send Payment (receiverAddress, amount)
Send Payment (amount)
Send Receipt (amount)
Payment Sent (amount, receipt)
Payment Sent (sessionId, amount, receipt)
opt [ Confirm Payments ]
Verify Payment (receipt)
1
2
3
4
5
6
7
8
9
10
11

FIGURE 6.6: Sequence diagram of a payment using Web Monetization[62]

a payment sent using Web Monetization.

It is assumed that the website has Web Monetization enabled. The flow for the payment is as follows [62]:

- The browser parses the code of the web page looking for the meta tag and, if it exists, for the payment pointer to where the money needs to be sent.
- The browser calculates the appropriate amount to send to the website per unit of time.
- In order to keep the anonymity of the user visiting the website, the browser generates a unique session ID for that specific session.
- To also keep the visited sites anonymous, the browser also gets a unique destination address for the website's payment pointer, as well as a shared secret for the session.
- Optionally, a payment receipt verifier can generate receipt of the operation.
- While the website is still in focus, the browser begins sending payments to the website (at the rate calculated at the beginning).
- The Web Monetization provider sends the money to the receiver via Interledger.
- Optionally, the receiver can generate a receipt and send it to the provider.
- The Web Monetization provider notifies the website the payment was sent successfully.
- The browser sends an event to the website, indicating the payment was successful.
- Optionally, the web page can send the receipt to the receipt verifier to confirm the payment.

## 6.4 QTZ: An ERC20 token

Interledger can provide connections to perform payments from any currency or payment system to any other one in existence, as long as they are correctly connected to the network. In this context, an ERC20 Quartz token was implemented (ERC20 tokens are defined in a previous chapter, in Section 3.2.2. The advantages and uses of the Quartz Token are also explained in a previous chapter, in Section 4.1.5).

In order to be able to either make donations using the token so that the receiver receives another currency or vice-versa, the token has to be made compatible with the Interledger network. For the scope of this project, tests were made in a simple Interledger network, the one described in Section 6.2.2.

Since this new Quartz coin (QTZ) is an ERC20 token, and ERC20 is a token standard, the process here described for including QTZ in an Interledger network can be trivially adapted to any other ERC20 token.

Also, being QTZ an ERC20 token, it is deployed over an Ethereum network. For this reason, it is possible to adapt the scheme described in Section 6.2.2 to fit this new coin. Ethereum settlement engines can be used to settle between nodes using QTZ.

The first step for integrating the token is creating and deploying the contracts of the token itself. The code has been adapted from the OpenZeppelin contracts [44]. In the configuration of the contracts it is necessary to fix the host to the IP of the Ethereum network where the token should be deployed. In the case of these tests, an Ethereum testnet. Then, the contracts are compiled and deployed to the network. Once this is done, the token will exist within the Ethereum network, being therefore possible to perform transactions using it within said network.

The next step is configuring the Ethereum settlement engines, so they are able to settle using the correct token. This is achieved by setting the token address of the settlement engine to the one of the token just deployed. This is, indicating the settlement engine that, from all of the tokens that may exist in the Ethereum network, QTZ is the one that it has to use for the settlements.

In order for QTZ to be exchanged for other currencies or tokens, the exchange rates have to be fixed. In the example network used in Section 6.2.2, the connector that exchanged ETH for XRP (and vice-versa) consulted the exchange rates from the exchange platform Coinbase[2]. This is not extensible for the new token, for none of these exchange platforms know its value. However, the rates can be set manually for the Interledger network using the Interledger API. In the test network of the project, only QTZ and XRP are used, so only those two rates have to be fixed.

When creating the accounts inside the Interledger nodes, the asset each account uses must be set. In this case, Alice will be using QTZ instead of ETH, as she was using in the previous test network.

Finally, the payments can be performed. To perform a payment from Alice (in QTZ) to Charlie (in XRP), it is only necessary to indicate the quantity, Charlie's payment pointer and Alice's password.

The precise process used for creating a token, deploying it and integrating it with the ILP network is explained in detail in Appendix A. It contains all the commands and steps necessary to recreate the network and the exchange.

## 6.5 Implementation details

Now that the decentralized technologies (Interledger and Web Monetization) have been properly explained, the implementation itself can be described in detail.

[2]https://www.coinbase.com/es/

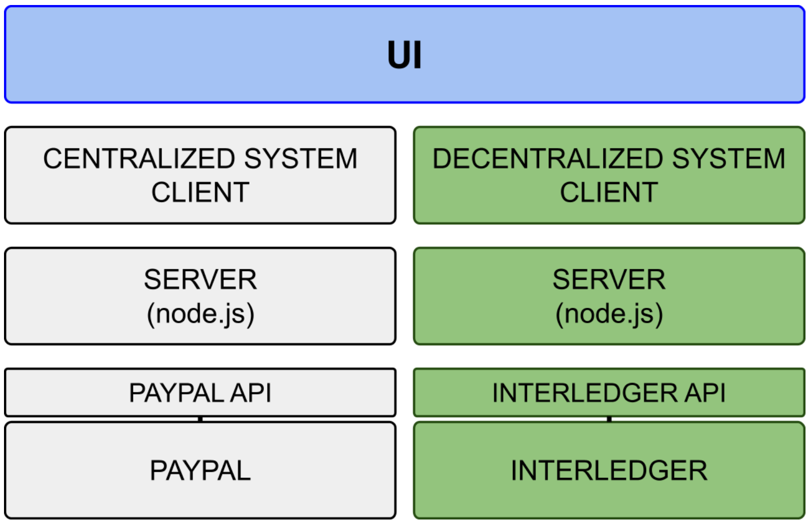


FIGURE 6.7: Architecture of the system. Highlighted in green is the decentralized part.

This section will explain the architecture of the decentralized system of the Quartz application, its flow and some other implementation details that are key for understanding the full implementation.

### 6.5.1 Architecture

The Interledger architecture (see 6.2.2) is a core part of the project, for the application is built over it and integrated with it. Nonetheless, the application has more layers, in order to allow a simple interaction between the user and the underlying system.

Figure 6.7 shows the different layer of the system. Highlighted in green are the ones corresponding to the decentralized subsystem. As explained in Section 5.3, the user interface is common to both the centralized and the decentralized part of the system, and each of the subsystems has its own lower layers, in both cases based on a server-client model.

In this section, the decentralized part will be analyzed.

As the figure shows, the most external layer is the UI (user interface), which is is the layer that manages the interaction between the user and the application. It is the exact same layer that was used in the centralized case, and it is implemented using React JS and material-ui.

This layer connects with the clients for both the centralized and the decentralized subsystems. In the case of the decentralized subsystem, the client receives the data inputs that the user enters via the UI and makes a GET request to the server, indicating the necessary data for the transaction. This code is also implemented using React JS.

The server makes calls to the Interledger API by sending POST requests, using the information received form the client. The code for the server is implemented using JavaScript and it runs on Node.js.

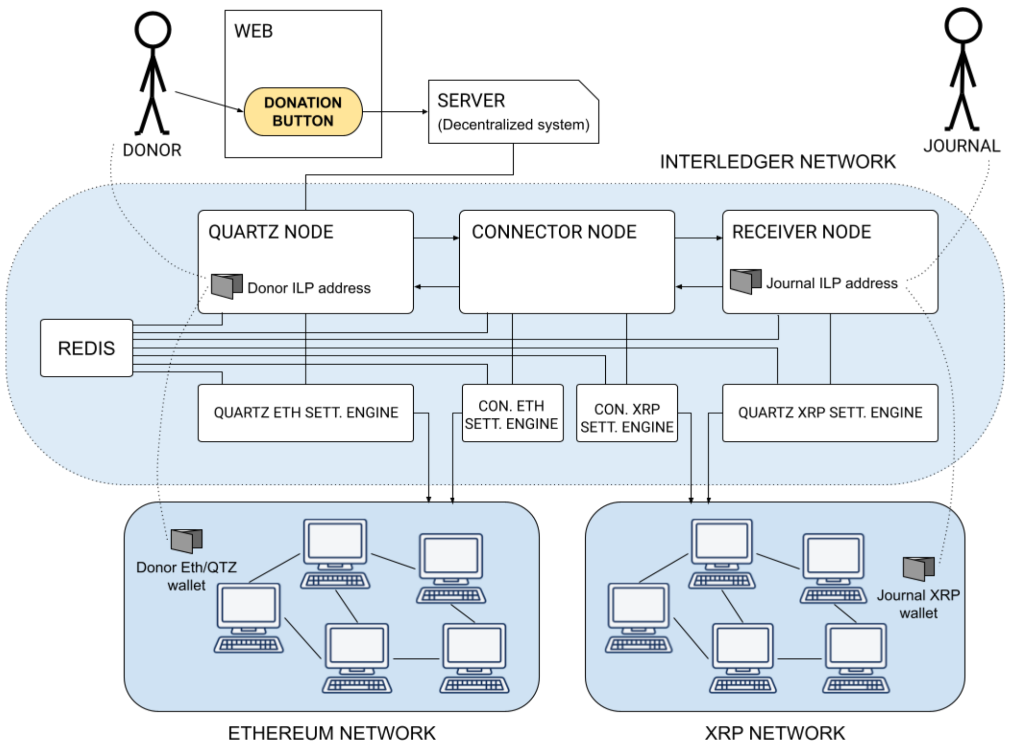


FIGURE 6.8: Overview of the decentralized subsystem.

The Interledger API is the access layer through which the functionalities of the ILP network are accessed. The Interledger system is the deepest layer of the system, and it is the one in charge of managing the shipment of the money, as well as the balance checks and other operations.

### 6.5.2 Implementation

In the context of the project, a test network was deployed to implement this architecture. The test network (which is a variation of the one described in section 6.2.2) simulates a small but complete Interledger network with a node that transacts using the QTZ token.

Figure 6.8 shows an overview of the entire decentralized subsystem, including the test network.

To simulate this test network locally, Docker (see Section 3.6.1) was used. Docker can be used to create a network locally by simulating several interconnected machines. Each machine will run a different service or application, thus creating a simulated Interledger network that can be run locally.

The detailed process and code for creating such network is explained in detail in Appendix A. The appendix shows the process in a low-level manner. Now this section will elaborate on the process at a high level.

First, a docker network is created. It will be referred as *local-ilp* from now on.

Then, several docker containers are deployed, each one serving a different purpose. A docker container contains *Redis*, which is a database for the settlement engines and the nodes, that will later be deployed. This database can be accessed by the services deployed in the rest of the containers.

Another docker contains an Ethereum testnet, that in this instance is initiated with a predefined seed, so that the public and private keys of the accounts are the same every time the network is redeployed. This is not necessary, but it simplifies the implementation and testing process.

Then, the QTZ token is implemented and deployed in this Ethereum network that has been created in a docker container. This way, the token can be accessed from the different nodes that are connected to this network. It is important to take into account that the token's contract has a certain address associated to it once it is deployed in the network. This address will be important later, when the nodes that use the coin are created.

The next step is deploying the nodes and their corresponding settlement engines. As explained in previous sections, settlement engines are entities that allow two interconnected nodes to settle with one another, this is, make an irrevocable exchange of funds between the two.

Three nodes are created. This text will refer to the nodes with the same names that were used for the deployment shown in Appendix A: Alice, Bob and Charlie. These names are used for simplicity, and represent the nodes as described in Figure 6.2 [3].

First, Alice's node is created. Alice's node only operates with QTZ. Therefore, two docker containers have to be deployed: one for the node itself and one for the Ethereum settlement engine (for the QTZ token runs over an Ethereum network).

In order to deploy the Ethereum settlement engine, it is necessary to tell the docker container to work inside the local-ilp network. Inside the container, the service corresponding to the Ethereum settlement engine is deployed and some important parameters are passed to the service, such as:

- The URL of the Ethereum testnet.
- The address of the QTZ token.
- The URL of the corresponding node (indicating the port to which the settlement engine will connect to). In this case, the URL for Alice's node and the corresponding port.
- The number of decimal positions the token is represented with.

Among others.

Then, Alice's node is deployed. In this case, the container runs a service meant for ilp nodes, and several parameters are passed to the service, such as:

- The ILP address of the node.
- A password for the node.
- The port through which it connects with the settlement engine.

Among others.

The procedure for Bob's and Charlie's nodes is similar. In the case of Charlie's node, it is a node that only operates with XRP (Ripple), so the settlement engine that has to be initiated is an XRP settlemet engine (it is different than the Ethereum one).

[3]When compared with Figure 6.8, *Alice* corresponds to *Quartz node*; *Bob* corresponds to *Connector node* and *Charlie* corresponds to *Receiver node.*

Bob's node, on the other hand, is a *connector* that operates with both QTZ and XRP, so it has to be able to settle with both currencies. Therefore, two settlement engines have to be deployed for Bob: one for Ethereum and one for XRP. Note that the settlement engines depend on the network, not the currency. For this reason, there is not an specific QTZ settlement engine, but an Ethereum one.

Charlie's and Bob's nodes are then created in a similar way Alice's node was.

For all this nodes and settlement engines, it is important to specify a fixed ip, that has to be different for each one of them. It is also important to indicate, for every one of them, that the docker network they work in is local-ilp.

Finally, the different accounts are created within the nodes. These accounts are the ones that allow the nodes to communicate with one another, send the packets and, eventually, make the economic exchanges.

Alice will be the node that issues the payment, Charlie the one that receives it and Bob will act as a connector between the two to make the currency exchange. This way, it is necessary to create in Alice's and Bob's nodes the respective sender and receiver accounts. This way, an *Alice* account will be created in Alice's node and a *Charlie* account will be created in Charlies node.

Now, in order for a node to be able to communicate with another one, it has to have an account inside the node it wants to communicate with. This accounts is not a usual account for storing funds, but its goal is to allow communication between the nodes. This way, a *Bob* account is created in Alice's node; an *Alice* and a *Charlie* account are created in Bob's node and a *Bob* account is created in Charlie's node. This is, in addition to the Alice account in Alice's node and the Charlie account in Charlie's node.

With this general knowledge of the architecture, it is now possible to explain in detail the sequence of calls among the different layers of the system.

### 6.5.3 Flow

The application follows a simple client-server structure: The client reads and manages the options for the user's donation and sends the server the necessary information to operate. The server listens to petitions regarding the payments and performs the API calls.

It works in a similar way to what was built for the PayPal payments, but instead of calling PayPal to perform the payments, the ILP process is called in its place.

Figure 6.9 contains a sequence diagram illustrating the architecture. The flow for a payment would be as follows:

- The user sets the quantity they want to donate and press the "Pay" button.
- The user is then asked to authenticate using its Interledger credentials (these being account and password). And there is also an option to choose the node in which the user has the account. For the purpose of the project, users are supposed to have an account in Quartz's node.
- The client sends a petition for a payment, containing the user, the receiver's payment pointer and the amount as parameters and the password as a header. The password is sent as a header instead of a parameter for security reasons.

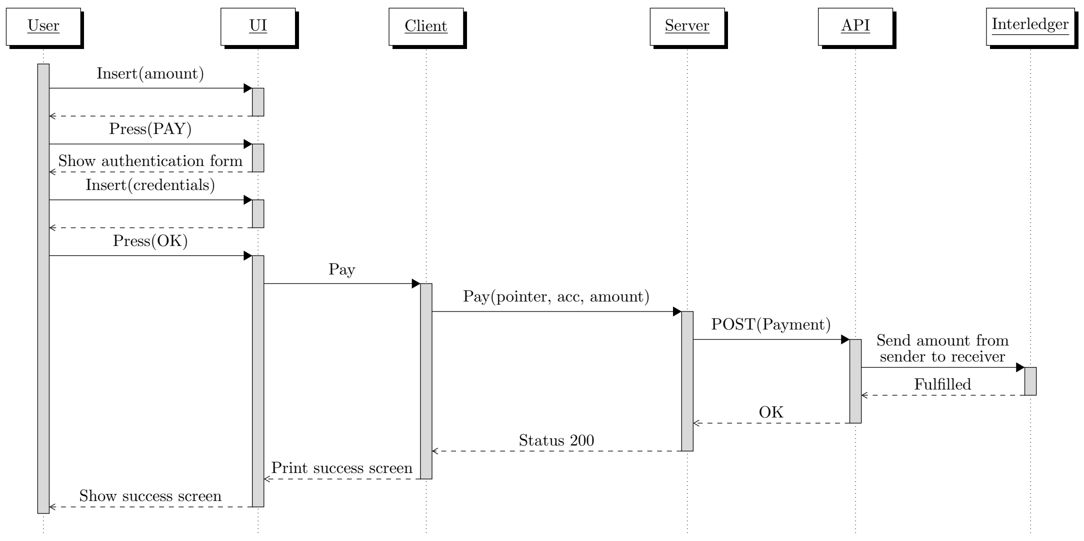


FIGURE 6.9: Sequence diagram for the Interledger payment button

- The server listens to the petition and makes a call - in the form of a POST petition - to the Interledger's API. The parameters and headers received from the client are used for this call.
- In case of an error, the server returns the headers and the type of errors, and prints them in the command-line.
- In case of success, it returns a status code of 200, which means everything worked correctly.
- The client receives the response and fixes the system state to success or error. This way, the front-end (or UI) can show the corresponding screen: error or success screen.

### 6.5.4 Web Monetization

The system also integrates with Web Monetization. The architecture for Web Monetization payments is explained in detail in Section 6.3.

This integration is done in two parts:

First of all, when the Quartz button is included in the web page of a journal, it automatically modifies the code of the head of the website to include the meta tag. This way, it enables Web Monetization payments in the website.

Moreover, there is a second layer to the integration, in terms of the Interledger network. The way this works is that the Quartz node in the ILP network can serve as a Web Monetization receiver or a Web Monetization provider.

For example, by specifying in the meta tag of the website a payment pointer provided by the Quartz node -this is, a payment pointer to an account that exists within the Quartz node -, the website, in this case the journal, can receive micropayments in QTZ.

# Chapter 7

# Conclusions and Future Work

Regarding all the previous work explained in prior chapters, conclusions are given and some suggestions are laid out to be carried out in the future.

## 7.1 Conclusions

This project was aimed to exploring how donations to journals and authors can be implemented in a practical way, and how decentralization can be applied to the task. In particular, exploring the Interledger Protocol as a technology to achieve these goals.

The tangible outcome of this project has been a functioning web app that provides functionality for sending donations by different means. As exposed in the Objectives (see 1.1), these means are:

- Via a centralized payment system (PayPal was used).
- Via a decentralized payment system, using any currency, including a specific coin created for donations to journals (Interledger was used).
- Via microdonations, using a browser extension (Web Monetization was used)

In this sense, the development goals were achieved.

A more intangible outcome, yet a main one, has been the study of different technologies and the design of an architecture for integrating both centralized and decentralized payment systems in a single application. Both approaches have advantages and drawbacks, but one interesting result of this work is that it is possible to implement a fully decentralized solution that allows to use different types of currencies and tokens.

The next subsections will provide detail about all these aspects.

### 7.1.1 Functionality and integration

From a practical point of view, the goal of the project was developing an applications that provides the functionalities previously described, by integrating with the different used technologies.

At a functional level, a button has been developed. This button can integrate within the web pages of the journals and provides them with the three main functionalities:

- With regard to the centralized payments, a button is presented that allows the user to make payments via PayPal, using a PayPal account or a credit card.
- With regard to the decentralized payments, a button is presented that allows the user to make payments via Interledger using the user's Interledger wallet.

- With regard to micropayments, the web is transparently integrated with the microdonations system and the donor can send microdonations using the Coil browser extension as a tool.

In each of these cases, a different integration was made from a technological point of view, not only mere functionality.

In the case of PayPal, the integration consists in the creation of a platform (the Quartz platform) within the PayPal system. This structure allows Quartz to act as an intermediary, being the entity in charge of managing all the donation buttons and forwarding the money to the corresponding journal in each case. Before being able to do so, it is necessary to register all the journals in the system, so that they become members of the platform.

In the case of Interledger, Quartz integrates as a new node in the network. In particular, this node has its own coin (QTZ). This node can also hold Interledger accounts for users using the coin and also serve as a connector to allow exchanges involving the coin. So the purpose is holding accounts and allowing these accounts to make exchanges (sending or receiving money) with the rest of the network without problems.

In the case of Web Monetization, Quartz integrates through the Interledger network. Coil's browser extension is the one in charge of sending the necessary funds, but the wallet of either the journal or the donor (or both) could be lodged in the Quartz node. This way, payments with QTZ are also integrated through microdonations.

### 7.1.2 Technology

One of the main focuses of this project was studying the adequacy of different technologies for the creation of a donation platform.

Providing a centralized option was seen as an essential first step to both understand the way donations and payments usually work and also to make it easier for new users to interact with the system. Most people are not familiarized with cryptocurrencies and having the option to pay with a credit card makes the usability easier.

However, despite the convenience of a technology such as PayPal, it proved to have some problematic restrictions due to its centralization. PayPal, as the central entity, has the last word in the decision of who can integrate with their systems and who can not.

For example, a system such as the one presented here, working perfectly in a test environment, finds difficulties when trying to bring it to a real-world environment (at a production level), for PayPal restricts the use of its live APIs to only companies that surpass a certain annual income threshold. This way, the functionality presented in this project can yet be implemented by someone not fitting the criteria, but some workarounds and adaptations are needed to avoid the blocking imposed by the centralized entity.

This kind of problems are some of the ones that are avoided by the use of decentralized technologies: a technology such as Interledger has proven to be more open to new participants, given there is no central entity deciding who can and who can not integrate with the system. Thereby, integrating with the system as a new member and even introduce new coins or currencies has no more limitations than the technological ones.

At a practical level, the technology has proven to serve correctly to provide the functionality that was initially intended, efficiently and effectively. It has also been proven that it is possible to integrate with the technology and use it as a bridge between different monetary systems to make donations, without this causing a big complexity wall for the users to use the system.

The case of Web Monetization is similar to this one. It has been proven that the technology can be integrated practically and transparently in the journals' websites, so that they can receive donations for the time the visitors expend in the web. Furthermore, since it works over an Interledger network, the previously mentioned integration of new coins or nodes works in the same way, and Web Monetization simply works in a higher layer. Using Coil as a tool has also be proven to be rather simple.

For these reasons, the technologies seem promising, given they also present a new range of possibilities in terms of donations, by allowing to make exchanges between any two monetary systems, without any intermediate barriers.

However, these technologies (Interledger and Web Monetization) are still very new technologies, that are still in a very early phase. Moreover, as for today there exist only a few projects and applications that are working in practice using them. For all of these reasons, the technologies are not yet fully developed, and using them in development is still quite complex. This makes the adaptation of these technologies to apply them in real systems harder.

So, to sum up, at a theoretical level, and in testing environments, both Interledger and Web Monetization are very promising technologies, that in the coming years may become essential tools for internet payments. As for today, however, it is still difficult to bring them out of testing environments - as the one tested in this project - to a real-life scenario.

Nonetheless, it seems likely that in the near future these technologies will be perfected enough to be used easily in production applications and systems.

## 7.2 Future work

Taking into account the work that has been done and the conclusions drawn from it, some modifications and improvements to the project arise. These improvements could be at a functional, technological or legal level.

Regarding functionality, there are several new aspects that can be added to the software:

- *Browser extension.* In the present state of the system, the integration with Web Monetization is done be storing the Interledger account o the journals in a node owned by Quartz. The donors issue their payments to said account using the Coil browser extension. For the purpose of a more global integration, a browser extension similar to the one from Coil could be implemented. A user with a Quartz account would be able to use the Quartz browser extension to send microdonations to webmonetized journals. This would have some advantages for the user, for they would not need to create an account in Coil, which is an external system, and they also would be able to have QTZ in their origin account.
- *Journal profile.* Another possible improvement is the creation of a profile page for the journals, in which they could consult their total funds and the donations

received from each source. The journals' web managers would be able to use this profile page to check the total balance of the account, in terms of money received from donations, as well as how much of that money has been received through microdonations and how much from direct donations. The specific origin of these donations could also be shown in the page. In the case of the PayPal donations, this kind of information is already available within the PayPal web page (in particular, in the user's dashboard), so this improvement being discussed is mainly referred to Interledger donations. The necessary data to implement this profile page can be consulted through the Interledger API.

On the other hand, there are also several improvements that can be implemented to the system from a technological point of view:

- *Encryption.* The current implementation, being a proof of concept for the system, does not apply any particular security measures when handling the user's data (which is test data for now). However, in order for the system to be used in a real environment, it is necessary to implement all the necessary measures. The main place where more security is needed is when the user makes a payment through Interledger. The password the user enters should be encrypted to protect it and sent to the server in a secure way.

  Moreover, some cryptographic and logical techniques could be used to protect the anonymity of the users when they make donations. Similarly to what Coil uses to protect the anonymity of its users [50, 14].
- *Centralized payments.* PayPal imposes severe restrictions in order to launch a live application using their API for marketplaces and platforms. It is possible to find workarounds for these barriers in PayPal's own APIs, for example using their "Simplest Chechout API"[1]. This API gives the possibility to create individual payment buttons for the journals, without relying in the marketplace or platform structure.

  Another possibility is using a different technology, not necessary PayPal. For example, Stripe[2] seems to be a promising technology for donations and payments. In this case, the modification would be replacing the centralized subsystem of the project, that currently relies solely on PayPal, with a new subsystem that manages the payments using Stripe.
- *Integrating new blockchains in the Interledger network.* For example, the Bloxberg [6] blockchain, where the QTZ coin could be deployed. For this integration to happen, it would be necessary to create a bridge with this new blockchain by using nodes that connect it to nodes that already exist within the Interledger network.

Lastly, in order to move this system, now a proof of concept in a test network, to a real-world environment, it is essential to study in detail the legal implications this could entail. Since the project revolves around money shipments and personal data management, it is essential to be careful, so there is still work to be done in this field:

- *How should money shipments be declared or managed?* The implemented system is a system for donations that moves money from the donors to the journals. The shipment of this money could be directly from sender to receiver, but in some cases, such as the PayPal platforms, the money goes through an intermediate entity managed by Quartz. In is necessary to study what limitations

[1]https://developer.paypal.com/docs/archive/checkout/integrate/
[2]https://stripe.com/

and implications exist for these type of structures. Similarly, in the case Quartz receives a fee for each payment through the application, it is necessary to study how that money must be declared and, again, if there are any limitations with respect to it.

- *How should the personal data be managed?* One of the essential parts of the system is working as a node in which users have accounts that can store and receive money. In order for this to be possible, it is necessary to to manage some of the user's personal data, as well as even the access to their funds. All this information is sensitive and might be subject to strict regulations regarding personal data (such as GDPR [19]). It is necessary to study carefully all these regulations before starting to store and manage the users' personal data.
- *How are the keys managed?* Related to the previous point, in order to make using blockchain more transparent for the user, it could be interesting to offer them to manage their blockchain account, as part of the data managed by the Quartz account (some users might find this useful, for blockchain technology is still quite cumbersome to use for users that are not used to it). In this case, it is necessary to study what kind of limitations and regulations exist with respect to storage and treatment of the users' keys.
- *How does a payment gateway work?* In some point of the development, for example, if a browser extension similar to the one Coil has is implemented, or in other cases, it could be necessary to be constituted as a payment gateway. As with the previous points, this is also a delicate process from a legal perspective, so it is necessary to study in detail all the existing regulation with respect to it.

# Appendix A

# Run a local ILP testnet using the Quartz token

## A.1 Introduction

This appendix contains the detailed steps for deploying the Quartz token contracts and running a local ILP testnet where the sender node operates with Quartz (QTZ).

The process is an adaptation of the one described in the ILP documentation. But there are a few subtle yet meaningful changes that have to be made in order for the new coin to be integrated.

The test network will be running using docker to simulate the different nodes (one sender, one receiver and one connector).

## A.2 Process

### A.2.1 Download docker images

First, install Dcoker and download the docker images.

```
docker pull interledgerrs/ilp-node
docker pull interledgerrs/ilp-cli
docker pull interledgerrs/ilp-settlement-ethereum
docker pull trufflesuite/ganache-cli
docker pull interledgerjs/settlement-xrp
docker pull redis
```

### A.2.2 Set up the environment

Start as explained in the ILP tutorial. The network, the redis container and the local Ethereum testnet must be initialized:

```
docker network create --subnet=192.168.128.0/24 local-ilp

docker run -d \
  --name redis \
  --network local-ilp \
  --ip 192.168.128.111 \
  redis
```

```
docker run -d \
  --name ethereum-testnet \
  --network local-ilp \
  --ip 192.168.128.2 \
  trufflesuite/ganache-cli \
  -m "abstract vacuum mammal awkward pudding scene penalty \
  purchase dinner depart evoke puzzle" \
  -i 1
```

### A.2.3 Create the token

The code for the Quartz token is available inside the repository for the TFG. If you download the code from the repository, this section is not needed, and you should skip to the next one, section A.2.4.

If you have not downloaded the repository, then follow the instructions in this section to create the contracts for the token.

The token must be deployed in the local ethereum testnet (that is running inside the container `ethereum-testnet`).

In the command line, do the following:

```
mkdir mytoken && cd mytoken
npm init -y

npm i --save-dev @openzeppelin/contracts

npm i truffle -g

npx truffle init
```

The `truffle-config.js` file must be modified adding a new network inside `networks`. This new network will be called `docker` and will be as follows:

```
module.exports = {
  networks: {
    docker: {
      host: "192.168.128.2",
      port: 8545,
      network_id: "*" // Match any network id
    }
  }
};
```

Where `host` is the ip of the Ethereum network that we are using (in this case, the testnet `ethereum-testnet`).

Create a new contract inheriting from ERC20 and implementing the new token.

Once that is done, write a script for the deployment of the contract and add it to the `/migrations` folder (in this instance, this will be for deploying the Quartz token). In this deployment script, the necessary parameters to initialize the contract must be provided (in this case, the amount of tokens that will be minted when the contract is deployed).

In `truffle-config.js`, change the compiler version to `0.6.12`. Keep an eye in the Solidity versions. The source files currently work for Solidity versions `>=0.6.0 <0.8.0`, but this could possibly change.

### A.2.4 Deploy the token

If the code for the token was downloaded from the GitHub repository, it is necessary to first do an `npm install`. If the code was generated as explained in the previous section, it is not necessary to do so.

Then, in both cases, proceed by compiling the contracts:

```
truffle compile
```

Then, migrate the contracts to the `docker` network:

```
truffle migrate --network docker
```

Copy the contract address that has been generated. This will be later used when initializing the dockers for the ethereum nodes.

Open the Truffle console in the `docker` network:

```
truffle console --network docker
```

Now that the contract has been deployed, it can be accessed and interacted with by assigning the contract instance to a variable.

```
let quartz = await ERC20Quartz.deployed()
```

This operation can only be performed by the issuer of the contract. For anybody else with access to the network to have an instance of the contract to interact with, they must be provided the address of the contract, so they can do:

```
let quartz = await ERC20Quartz.at(<contract address
                                        in quotes>)
```

The contract address can be consulted by doing:

```
quartz.address
```

To check the accounts existing within the blockchain (to make a transaction to one of them or consult their balance) they can be consulted from the Truffle console:

```
accounts
```

If the initialization of the contract worked correctly, the first of the accounts, which corresponds to the issuer (and will be later assigned to Alice) should have all the tokens that have been minted in the initialization.

```
let balance = await quartz.balanceOf(<public address of the
                                            first account>)
balance.toNumber()
```

`toNumber` is necessary so the information is displayed in a natural way.

### A.2.5 Start the ILP nodes

Following the ILP tutorial, initialize the nodes. A couple considerations have to be made:

- The `token_address` is the address of the `ERC20Quartz` contract.
- The ip's of the nodes are fixed, based on the ip of the local ethereum testnet (this initial ip can be checked from the console of the docker where the network is running). In this case, the ip of the network was set manually.
- The asset scale is fixed to 9.
- The exchange rate is provided by the API, so the parameter `exchange_rate.provider` is deleted.

```
docker run -d \
  --name alice-eth \
  --network local-ilp \
  -e "RUST_LOG=interledger=trace" \
  --ip 192.168.128.3 \
  interledgerrs/ilp-settlement-ethereum \
  --private_key 380eb0f3d505f087e438eca80bc4df9a7
                faa24f868e69fc0440261a0fc0567dc \
  --confirmations 0 \
  --poll_frequency 1000 \
  --ethereum_url http://ethereum-testnet:8545 \
  --token_address 0x770bC1820890415bB14a3B8f992c19caA74906aD\
  --connector_url http://alice-node:7771 \
  --redis_url redis://redis:6379/0 \
  --asset_scale 9 \
  --settlement_api_bind_address 0.0.0.0:3000

docker run -d \
  --name alice-node \
  --network local-ilp \
  -e "RUST_LOG=interledger=trace" \
  --ip 192.168.128.4 \
  interledgerrs/ilp-node \
  --ilp_address example.alice \
  --secret_seed 8852500887504328225458511465394229
                    327394647958135038836332350604 \
  --admin_auth_token hi_alice \
  --redis_url redis://redis:6379/1 \
  --http_bind_address 0.0.0.0:7770 \
  --settlement_api_bind_address 0.0.0.0:7771



docker run -d \
  --name bob-eth \
  --network local-ilp \
  -e "RUST_LOG=interledger=trace" \
  --ip 192.168.128.5 \
  interledgerrs/ilp-settlement-ethereum \
  --private_key cc96601bc52293b53c4736a12af9130abf3
                    47669b3813f9ec4cafdf6991b087e \
  --confirmations 0 \
  --poll_frequency 1000 \
  --ethereum_url http://ethereum-testnet:8545 \
  --token_address 0x770bC1820890415bB14a3B8f992c19caA74906aD\
  --connector_url http://bob-node:7771 \
  --redis_url redis://redis:6379/2 \
  --asset_scale 9 \
  --settlement_api_bind_address 0.0.0.0:3000

docker run -d \
  --name bob-xrp \
  --network local-ilp \
  -e "DEBUG=settlement*" \
  -e "CONNECTOR_URL=http://bob-node:7771" \
  -e "REDIS_URI=redis://redis:6379/3" \
  -e "ENGINE_PORT=3001" \
  --ip 192.168.128.6 \
  interledgerjs/settlement-xrp

docker run -d \
  --name bob-node \
  --network local-ilp \
  -e "RUST_LOG=interledger=trace" \
  --ip 192.168.128.7 \
  interledgerrs/ilp-node \
  --ilp_address example.bob \
  --secret_seed 1604966725982139900555208458637022
                    8755636914554293737193680533354 \
  --admin_auth_token hi_bob \
  --redis_url redis://redis:6379/4 \
  --http_bind_address 0.0.0.0:7770 \
  --settlement_api_bind_address 0.0.0.0:7771


docker run -d \
  --name charlie-xrp \
  --network local-ilp \
  -e "DEBUG=settlement*" \
  -e "CONNECTOR_URL=http://charlie-node:7771" \
  -e "REDIS_URI=redis://redis:6379/5" \
  -e "ENGINE_PORT=3000" \
  --ip 192.168.128.8 \
  interledgerjs/settlement-xrp

docker run -d \
  --name charlie-node \
  --network local-ilp \
  -e "RUST_LOG=interledger=trace" \
  --ip 192.168.128.9 \
  interledgerrs/ilp-node \
  --secret_seed 12323621311221399005552084586370228
                    75563691455429373719368053354 \
  --admin_auth_token hi_charlie \
  --redis_url redis://redis:6379/6 \
  --http_bind_address 0.0.0.0:7770 \
  --settlement_api_bind_address 0.0.0.0:7771
```

Next, change the exchange rates for the new coin using the API:

```
curl -X PUT "http://192.168.128.4:7770/rates" \
     -H "Authorization: Bearer hi_alice" \
     -H  "accept: application/json" \
     -H  "Content-Type: application/json" \
     -d "{\"QTZ\":1.00,\"XRP\":0.000283}"
curl -X PUT "http://192.168.128.7:7770/rates" \
     -H "Authorization: Bearer hi_bob" \
     -H  "accept: application/json" \
     -H  "Content-Type: application/json" \
     -d "{\"QTZ\":1.00,\"XRP\":0.000283}"
curl -X PUT "http://192.168.128.9:7770/rates" \
     -H "Authorization: Bearer hi_charlie" \
     -H  "accept: application/json" \
     -H  "Content-Type: application/json" \
     -d "{\"QTZ\":1.00,\"XRP\":0.000283}"
```

After that, create the accounts. Also here, a couple considerations have to be taken:

- In the Alice and Bob accounts that used `ETH` in the example, the asset code is now `QTZ`.
- The asset-scale is now 0.

```
alias   alice-cli="docker run --rm \
                    --network local-ilp interledgerrs/ilp-cli
                    --node http://alice-node:7770"
alias     bob-cli="docker run --rm \
                    --network local-ilp interledgerrs/ilp-cli \
                    --node http://bob-node:7770"
alias charlie-cli="docker run --rm \
                    --network local-ilp interledgerrs/ilp-cli \
                    --node http://charlie-node:7770"

alice-cli accounts create alice \
  --auth hi_alice \
  --ilp-address example.alice \
  --asset-code QTZ \
  --asset-scale 9 \
  --ilp-over-http-incoming-token alice_password

alice-cli accounts create bob \
  --auth hi_alice \
  --ilp-address example.bob \
  --asset-code QTZ \
  --asset-scale 9\
  --settlement-engine-url http://alice-eth:3000 \
  --ilp-over-http-incoming-token bob_password \
  --ilp-over-http-outgoing-token alice_password \
  --ilp-over-http-url http://bob-node:7770/accounts/
                                          alice/ilp \
  --settle-threshold 100000 \
  --settle-to 0 \
  --routing-relation Peer

bob-cli accounts create alice \
  --auth hi_bob \
  --ilp-address example.alice \
  --asset-code QTZ \
  --asset-scale 9 \
  --max-packet-amount 100000 \
  --settlement-engine-url http://bob-eth:3000 \
  --ilp-over-http-incoming-token alice_password \
  --ilp-over-http-outgoing-token bob_password \
  --ilp-over-http-url http://alice-node:7770/accounts/
                                              bob/ilp \
  --min-balance -150000 \
  --routing-relation Peer

bob-cli accounts create charlie \
  --auth hi_bob \
  --asset-code XRP \
  --asset-scale 6 \
  --settlement-engine-url http://bob-xrp:3001 \
  --ilp-over-http-incoming-token charlie_password \
  --ilp-over-http-outgoing-token bob_other_password \
  --ilp-over-http-url http://charlie-node:7770/accounts/
                                              bob/ilp \
  --settle-threshold 10000 \
  --settle-to -1000000 \
  --routing-relation Child

charlie-cli accounts create bob \
  --auth hi_charlie \
  --ilp-address example.bob \
  --asset-code XRP \
  --asset-scale 6 \
  --settlement-engine-url http://charlie-xrp:3000 \
  --ilp-over-http-incoming-token bob_other_password \
  --ilp-over-http-outgoing-token charlie_password \
  --ilp-over-http-url http://bob-node:7770/accounts/
                                          charlie/ilp \
  --min-balance -50000 \
  --routing-relation Parent
```

```
charlie-cli accounts create charlie \
  --auth hi_charlie \
  --asset-code XRP \
  --asset-scale 6 \
  --ilp-over-http-incoming-token charlie_password
```

### A.2.6 Perform the payment

Send a certain amount of tokens from Alice to Charlie (in this example, 20):

```
alice-cli pay alice \
  --auth alice_password \
  --amount 20 \
  --to http://charlie-node:7770/accounts/charlie/spsp
```

### A.2.7 Delete the nodes and start over

#### Only deleting the nodes, but keeping the network

Delete the accounts:

```
alice-cli accounts delete alice --auth hi_alice
alice-cli accounts delete bob --auth hi_alice
bob-cli accounts delete alice --auth hi_bob
bob-cli accounts delete charlie --auth hi_bob
charlie-cli accounts delete bob --auth hi_charlie
charlie-cli accounts delete charlie --auth hi_charlie
```

Remove the dockers:

```
docker stop alice-node bob-node charlie-node alice-eth \
            bob-eth bob-xrp charlie-xrp
docker rm alice-node bob-node charlie-node alice-eth \
            bob-eth bob-xrp charlie-xrp
```

#### Remove everything

Sometimes removing the nodes but keeping the network results in some issues when deleting the accounts. When this happens, the best option is to remove everything and start the process from scratch.

```
docker stop redis ethereum-testnet alice-node bob-node \
                  charlie-node alice-eth bob-eth bob-xrp \
                  charlie-xrp
docker rm redis ethereum-testnet alice-node bob-node \
                charlie-node alice-eth bob-eth bob-xrp \
                charlie-xrp
docker network rm local-ilp
```

# Appendix B

# Perform transactions using the server

## B.1 Introduction

This appendix contains the detailed steps for performing transactions through an ILP network using a server to handle the shipment petitions.

The server[1] accepts petitions to send a certain amount of Quartz from a given account to another. It is assumed that all the accounts that send the money are in the same host.

## B.2 Process

### B.2.1 Set up the environment

To start using the server, there has to be an ILP network running. The sender and the receiver accounts have to exist withing said network.

As a test, the local ILP docker network that was deployed in the previous appendix (see Appendix A) will be used.

It is assumed that the reader has already downloaded the code for the server from the repository.

### B.2.2 Configure the server

Copy the file `serverConfig.json.example` into a new file `serverConfig.json` and change the host and the port where the accounts that send the money are going to be.

As previously explained, it is assumed that the server will be managing transactions coming from accounts in a particular host, although the receivers could be anywhere.

For the test with the docker network, the hostname and port are:

```json
{
    "hostname": "192.168.128.4",
    "port": 7770
}
```

[1]The code for the server can be found in the GitHub repository of the project: https://github.com/ElenaPT/TFG_Inf_2021/tree/incoming_branch/src/token/ILP_server

### B.2.3 Run the server

Before running the server, the ILP network has to be up and running. Appendix **??** contains instructions for deploying a local network for testing using docker.

Once that is managed, simply do:

```
cd ILP_server
node transaction_server.js
```

### B.2.4 Perform a transaction

The server expects the sender, the receiver and the amount of Quartz to be sent as parameters. It also needs the authorization-token of the sender, and expects it as a header.

Taking this into account, to perform the transaction do a `curl` petition as follows:

```
curl "http://localhost:8080/?from=<sender-account> \
     &to=<receiver-payment-pointer>&amount=<amount>" \
     -H "Authorization: Bearer <sender-auth-token>"
```

For instance, to send *20 QTZ* from *Alice* to *Charlie* in the docker network, the petition would be:

```
curl "http://localhost:8080/?from=alice \
     &to=http://charlie-node:7770/accounts/charlie/spsp \
     &amount=20" -H "Authorization: Bearer alice_password"
```

# Bibliography

[1] Abdulla Alshamsi and Prof. Peter Andras. "User perception of Bitcoin usability and security across novice users". In: *International Journal of Human-Computer Studies* 126 (June 1, 2019), pp. 94–110. ISSN: 1071-5819. DOI: 10.1016/j.ijhcs.2019.02.004. URL: https://www.sciencedirect.com/science/article/pii/S1071581918301459 (visited on 09/04/2021).

[2] *API requests*. URL: https://developer.paypal.com/docs/api/reference/api-requests/ (visited on 08/08/2021).

[3] Carl T. Bergstrom and Theodore C. Bergstrom. "The costs and benefits of library site licenses to academic journals". In: *Proceedings of the National Academy of Sciences* 101.3 (Jan. 20, 2004). Publisher: National Academy of Sciences Section: Social Sciences, pp. 897–902. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.0305628101. URL: https://www.pnas.org/content/101/3/897 (visited on 09/02/2021).

[4] *BitGive Foundation - 1st Bitcoin and Blockchain Nonprofit*. BitGive Foundation. URL: https://www.bitgivefoundation.org/ (visited on 05/06/2021).

[5] Bo-Christer Björk. "Why Is Access to the Scholarly Journal Literature So Expensive?" In: *portal: Libraries and the Academy* 21.2 (2021). Publisher: Johns Hopkins University Press, pp. 177–192. ISSN: 1530-7131. DOI: 10.1353/pla.2021.0010. URL: https://muse.jhu.edu/article/787862 (visited on 09/04/2021).

[6] *Blockchain Infrastructure for Scientific Research*. Bloxberg. URL: https://bloxberg.org/ (visited on 08/30/2021).

[7] Vitalik Buterin. *Ethereum Whitepaper*. ethereum.org. 2013. URL: https://ethereum.org (visited on 05/30/2021).

[8] H. Frank Cervone. "Understanding agile project management methods using Scrum". In: *OCLC Systems & Services: International digital library perspectives* 27.1 (Jan. 1, 2011). Publisher: Emerald Group Publishing Limited, pp. 18–22. ISSN: 1065-075X. DOI: 10.1108/10650751111106528. URL: https://doi.org/10.1108/10650751111106528 (visited on 05/22/2021).

[9] *Coil - A new way to enjoy content*. URL: https://coil.com/ (visited on 04/29/2021).

[10] *Digital Wallet and Payment Pointers*. URL: https://webmonetization.org/docs/ilp-wallets (visited on 05/31/2021).

[11] Nadia Eghbal. "Roads and Bridges: The Unseen Labor Behind Our Digital Infrastructure". In: *Technical Report. Ford Foundation* (2016), p. 143. URL: https://www.fordfoundation.org/work/learning/research-reports/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure/.

[12] *Empowering App Development for Developers | Docker*. URL: https://www.docker.com/ (visited on 05/31/2021).

[13] *Express - Infraestructura de aplicaciones web Node.js*. URL: https://expressjs.com/es/ (visited on 08/29/2021).

[14] *Extended protocol design.* Privacy Pass. URL: https://privacypass.github.io/protocol/ (visited on 08/30/2021).

[15] Fabian Vogelsteller and Vitalik Buterin. "EIP-20: ERC-20 Token Standard". In: *Ethereum Improvements Proposals* no. 20 (Nov. 2015). URL: https://eips.ethereum.org/EIPS/eip-20.

[16] Flattr. *Flattr - Contributors.* Flattr. URL: https://flattr.com/ (visited on 04/29/2021).

[17] Armen Yuri Gasparyan et al. "Rewarding Peer Reviewers - Maintaining the Integrity of Science Communication". In: *Journal of Korean Medical Science* 30.4 (2015), p. 360. ISSN: 1011-8934, 1598-6357. DOI: 10.3346/jkms.2015.30.4.360. URL: https://jkms.org/DOIx.php?id=10.3346/jkms.2015.30.4.360 (visited on 05/31/2021).

[18] Armen Yuri Gasparyan et al. "Rewarding Peer Reviewers: Maintaining the Integrity of Science Communication". In: *Journal of Korean Medical Science* 30.4 (Mar. 19, 2015). Publisher: The Korean Academy of Medical Sciences, pp. 360–364. DOI: 10.3346/jkms.2015.30.4.360. URL: https://synapse.koreamed.org/articles/1022863 (visited on 09/02/2021).

[19] *General Data Protection Regulation (GDPR) – Official Legal Text.* General Data Protection Regulation (GDPR). URL: https://gdpr-info.eu/ (visited on 06/10/2021).

[20] *Get an access token.* URL: https://developer.paypal.com/docs/api/reference/get-an-access-token/ (visited on 08/08/2021).

[21] *Giveth Homepage.* URL: https://giveth.io/ (visited on 05/05/2021).

[22] *GiveTrack™ - The future of philanthropy built upon Bitcoin and Blockchain.* URL: https://www.givetrack.org/view/13/chandolo-primary-school-water-project/updates (visited on 05/05/2021).

[23] Toby Green. "Is open access affordable? Why current models do not work and why we need internet-era transformation of scholarly communications". In: *Learned Publishing* 32.1 (2019), pp. 13–25. ISSN: 1741-4857. DOI: 10.1002/leap.1219. URL: https://onlinelibrary.wiley.com/doi/10.1002/leap.1219 (visited on 09/04/2021).

[24] Dick Hardt. *The OAuth 2.0 Authorization Framework.* Request for Comments RFC 6749. Num Pages: 76. Internet Engineering Task Force, Oct. 2012. DOI: 10.17487/RFC6749. URL: https://datatracker.ietf.org/doc/rfc6749 (visited on 08/09/2021).

[25] *Interedger's GitHub repository for the rafiki.money software.* Apr. 29, 2021. URL: https://github.com/interledgerjs/rafiki.money (visited on 04/29/2021).

[26] *Interledger: Interledger Architecture.* URL: https://interledger.org/rfcs/0001-interledger-architecture/ (visited on 05/10/2021).

[27] *Interledger: Overview.* URL: https://interledger.org/developer-tools/get-started/overview/ (visited on 05/10/2021).

[28] *Interledger: Settlement Engines.* URL: https://interledger.org/rfcs/0038-settlement-engines/ (visited on 05/16/2021).

[29] Weiling Ke and Ping Zhang. "The Effects of Extrinsic Motivations and Satisfaction in Open Source Software Development". In: *J. AIS* 11 (Dec. 1, 2010). DOI: 10.17705/1jais.00251.

[30] *Ko-fi | Donations and subscriptions from fans for the price of a coffee. No fees.* Ko-fi. URL: https://ko-fi.com/ (visited on 04/29/2021).

[31] Sandeep Krishnamurthy. “Monetary donations to an open source software platform”. In: *Research Policy* 38 (Mar. 1, 2009), pp. 404–414. DOI: 10.1016/j.respol.2008.11.004.

[32] Vincent Larivière, Stefanie Haustein, and Philippe Mongeon. “Big publishers, bigger profits : how the scholarly community lost the control of its journals”. In: *Media trope*. Media trope. Vol. 5. Accepted: 2020-04-20T18:20:05Z Issue: 2, Libraries in crisis. Carleton University. Department of English language and literature, 2015, pp. 102–110. URL: https://papyrus.bib.umontreal.ca/xmlui/handle/1866/23285 (visited on 09/03/2021).

[33] Vincent Larivière, Stefanie Haustein, and Philippe Mongeon. “The Oligopoly of Academic Publishers in the Digital Era”. In: *PLOS ONE* 10.6 (June 10, 2015). Publisher: Public Library of Science, e0127502. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0127502. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0127502 (visited on 05/19/2021).

[34] Evgeniya Lupova-Henry and Ámbar Tenorio-Fornes. *Quartz OA White Paper*. quartz.to. 2021. URL: https://quartz.to/wp-content/uploads/2021/07/Quartz-OA-White-Paper-.pdf (visited on 09/13/2021).

[35] *Manage your apps*. URL: https://developer.paypal.com/docs/api-basics/manage-apps/ (visited on 08/08/2021).

[36] *Material-UI: A popular React UI framework*. URL: https://material-ui.com/ (visited on 05/31/2021).

[37] *MetaMask*. URL: https://metamask.io/ (visited on 08/09/2021).

[38] E. Rajpert-De Meyts, S. Losito, and D. T. Carrell. “Rewarding peer-review work: the Publons initiative”. In: *Andrology* 4.6 (2016), pp. 985–986. ISSN: 2047-2927. DOI: 10.1111/andr.12301. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/andr.12301 (visited on 09/02/2021).

[39] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008. URL: https://bitcoin.org/bitcoin.pdf (visited on 05/30/2021).

[40] Peter Newmark. “Peer review and the rewards of open access”. In: *Nature* 422.6933 (Apr. 2003), pp. 661–661. ISSN: 1476-4687. DOI: 10.1038/422661b. URL: https://www.nature.com/articles/422661b (visited on 09/04/2021).

[41] Node.js. *Node.js*. Node.js. URL: https://nodejs.org/es/ (visited on 05/31/2021).

[42] Michael Nofer et al. “Blockchain”. In: *Business & Information Systems Engineering* 59 (Mar. 20, 2017). DOI: 10.1007/s12599-017-0467-3.

[43] Silas Nzuva. “Smart Contracts Implementation, Applications, Benefits, and Limitations”. In: *Journal of Information Engineering and Applications* (Oct. 9, 2019). DOI: 10.7176/JIEA/9-5-07.

[44] *OpenZeppelin*. OpenZeppelin. URL: https://openzeppelin.com/ (visited on 05/13/2021).

[45] Cassandra Overney et al. “How to not get rich: an empirical study of donations in open source”. In: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. Seoul South Korea: ACM, June 27, 2020, pp. 1209–1221. ISBN: 978-1-4503-7121-6. DOI: 10.1145/3377811.3380410. URL: https://dl.acm.org/doi/10.1145/3377811.3380410 (visited on 05/30/2021).

[46] *Plaudit · Open endorsements from the academic community*. URL: https://plaudit.pub/ (visited on 04/29/2021).

[47] *Rafiki Money*. URL: https://rafiki.money/ (visited on 04/29/2021).

[48] *React – Una biblioteca de JavaScript para construir interfaces de usuario*. URL: https://es.reactjs.org/ (visited on 05/31/2021).

[49] School of Computing Universiti Utara Malaysia Kedah, Malaysia and Haroon Shakirat Oluwatosin. "Client-Server Model". In: *IOSR Journal of Computer Engineering* 16.1 (2014), pp. 57–71. ISSN: 22788727, 22780661. DOI: 10.9790/0661-16195771. URL: http://www.iosrjournals.org/iosr-jce/papers/Vol16-issue1/Version-9/J016195771.pdf (visited on 08/08/2021).

[50] Sharafian. *Doubling Down on Privacy*. May 11, 2020. URL: https://write.as/sharafian/doubling-down-on-privacy (visited on 08/30/2021).

[51] David J. Solomon and Bo-Christer Björk. "A study of open access journals using article processing charges". In: *Journal of the American Society for Information Science and Technology* 63.8 (2012), pp. 1485–1495. ISSN: 1532-2890. DOI: 10.1002/asi.22673. URL: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/asi.22673 (visited on 09/02/2021).

[52] Richard M. Stallman and Joshua Gay. *Free software, free society: selected essays*. 1st. ed. OCLC: 253840339. Boston, Mass: Free Software Foundation, 2002. 220 pp. ISBN: 978-1-882114-98-6.

[53] Helperbit Team. *Home*. Helperbit. URL: https://app.helperbit.com/ (visited on 05/05/2021).

[54] Technische Universität München, Germany et al. "Code Reuse in Open Source Software Development: Quantitative Evidence, Drivers, and Impediments". In: *Journal of the Association for Information Systems* 11.12 (Dec. 2010), pp. 868–901. ISSN: 15369323. DOI: 10.17705/1jais.00248. URL: http://aisel.aisnet.org/jais/vol11/iss12/2/ (visited on 05/30/2021).

[55] Ámbar Tenorio-Fornés et al. "Decentralizing science: Towards an interoperable open peer review ecosystem using blockchain". In: *Information Processing & Management* 58.6 (Nov. 1, 2021), p. 102724. ISSN: 0306-4573. DOI: 10.1016/j.ipm.2021.102724. URL: https://www.sciencedirect.com/science/article/pii/S0306457321002089 (visited on 09/21/2021).

[56] Stefan Thomas and Evan Schwartz. "A Protocol for Interledger Payments". In: *interledger.org* (2016), p. 25.

[57] Elena Perez Tirador and Antonio Tenorio-Fornes. "Decentralizing peer reviewing to increase transparency, quality and reliability". In: *Blockchain for Science Conference* (2019), p. 3.

[58] Arvind Tripathi. "Acceptance of monetary rewards in open source software development". In: *Research Policy* (2013). URL: https://www.academia.edu/24309028/Acceptance_of_monetary_rewards_in_open_source_software_development (visited on 05/30/2021).

[59] Uphold. *Uphold - Compra, Vende y Envía BTC, XRP y MÁS en Segundos*. Uphold. URL: https://uphold.com/es/ (visited on 04/29/2021).

[60] Richard Van Noorden. "Open access: The true cost of science publishing". In: *Nature* 495.7442 (Mar. 1, 2013), pp. 426–429. ISSN: 1476-4687. DOI: 10.1038/495426a. URL: https://www.nature.com/articles/495426a (visited on 09/02/2021).

[61] *Web Monetization*. URL: https://webmonetization.org/ (visited on 05/12/2021).

[62] *Web Monetization Explainer*. URL: https://webmonetization.org/docs/explainer (visited on 05/27/2021).

[63] *Web Monetization Providers (Sending Payments)*. URL: https://webmonetization.org/docs/sending (visited on 05/12/2021).

[64] John Willinsky. *The unacknowledged convergence of open source, open access, and open science.* First Monday, ISSN 1396-0466. Archive Location: 1996 - 2005 Publisher: Valauskas, Edward J. Aug. 1, 2005. URL: https://firstmonday.org/ojs/index.php/fm/article/download/1265/1185?inline=1 (visited on 05/30/2021).

[65] Monica Aniela Zaharie and Marco Seeber. "Are non-monetary rewards effective in attracting peer reviewers? A natural experiment". In: *Scientometrics* 117.3 (Dec. 1, 2018), pp. 1587–1609. ISSN: 1588-2861. DOI: 10.1007/s11192-018-2912-6. URL: https://doi.org/10.1007/s11192-018-2912-6 (visited on 09/04/2021).