École Polytechnique Fédérale de Lausanne

# Columbus IV
# Implementation of an intuitive and insightful blockchain explorer (cont.)

by Pilar Marxer and Rosa José Sara

## Master and Bachelor Project Report

Prof. Bryan Ford
Project Director

Noémien Kocher
Project Supervisor

Decentralized Distributed Systems Laboratory (DEDIS)
EPFL

June 11, 2021

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

Imagine your bank gives for free to everyone of its' subscriber a book of transactions that everyone can check and write on, but nobody can delete lines or destroy it. This analogy completely illustrates the concept of blockchain. A blockchain is a type of data storage tool, that is starting to be used in a wide set of domains. It is used because of its transparency (decentralized) and its verifiability (every block is linked to the previous one). However in practice without a blockchain explorer, users have no adequate tool to verify their transactions. The main goal of this project is to offer different classes of users a way to visualize the Byzcoin blockchain for different purposes and understand it's underlying concepts.

## 1.2 Byzcoin

Byzcoin [3] is a distributed ledger (database) implemented by the DEDIS lab at EPFL. Byzcoin uses a blockchain called Skipchain [4]. As in regular blockchains every node has its local copy of the ledger, the particularity of Skipchains is that they add forward and backward multi-hop links to the blockchain, these enable to efficiently and securely traverse short or long distances in the chain.
Every change to the ledger is packed into a block that is cryptographically chained to the previous one, the forward links represent consensus among the verifiers that these changes belong in the database.
In Byzcoin the data is organized as instances of smart contracts. To update the database, users can spawn (create), invoke (apply a command or function), or delete a smart contract. Every request to update the ledger is a transaction made up of one or more instructions, which is then sent to a node. A quorum of nodes (roster) must then verify and sign the transaction. Thanks to

the Cothority [2] framework, nodes can communicate safely with each other and form a collective authority (cothority). Columbus [6] is the Byzcoin explorer. It gives Byzcoin subscribers a visual tool to monitor the evolution of Byzcoin and verify its transactions.

## 1.3 Previous work on the Explorer

The Project started in December 2019. Julian von Felten and Anthony Iozzia, in the realm of their Bachelor project started to develop a tool that would centralize all the features of the ByzCoin explorer. V0.0.1 was then released in June 2020, and later in August after some refactoring and documentation of the code v0.0.2 was released. Sophia Artioli along with Lucas Trognon, continued to work on the explorer during the fall semester 2020, they added new features and reworked the whole design of the explorer. We continued working on what they started, by adding new features and continuing to enhance the interaction with the explorer.

# Chapter 2

# Initial UX survey and Workflow

In the first weeks of the project we assessed what the timeline of our project would be and what features were needed. We did this by surveying a set of users and comparing Columbus with other blockchain Explorers such as Etherscan[1]. From their insights, combined with ours and a list of goals that Noémien Kocher gave us, we were able to develop our own feature list and plan how we were going to proceed.

## 2.1 Feedback and insights

Initially, we took on the set of questions (slightly adjusted), that Sophia and Lucas used in there own UX survey [11]. We were able to collect feedback of a set of 6 users, most of them were novice to the concept of a blockchain explorer, and all of them weren't familiar with Byzcoin.
The main drawback novice users were encountering, was the fact that they could not understand the information that was shown to them (e.g. the block details, and the transaction details). Furthermore, some of the interactions with the visual interface of the chain were not intuitive enough and not all users discovered them right away such as the click-and-drag on the chain.
Intermediate users, usually understood the visual representation of the blockchain, but were lost on the notions related to the Skipchain and were curious to know more about it.
Overall, the main insights were:

- Lack of information about the content of the main elements shown to the users.
- Aesthetically, the font-size is too big, and one can not display on the whole page the information of the block details and transaction details when scrolling down.
- Some features such as the instruction search are not used.

[1] https://etherscan.io/

Our first impressions were solely based on insights from users unfamiliar with Byzcoin, at this phase of the project we had not the opportunity to interview lab members.

## 2.2 Workflow and the initial list of features

### 2.2.1 Stack

At the technical level, the project used the following stack:

- Typescript, as the frontend language
- NPM, as the package manager
- Webpack, as the bundler
- D3, as the visualization library [8]
- RxJS, as the reactive programming library [13]
- UIkit, as the CSS framework [14]
- Intro.js, as the guided tour library [10]

### 2.2.2 Features

After the first weeks of research, from the feedback we received, we came up with a list of features. We organised our features by priority and categorize them so that each of us would tackle different tasks independently. We separated our semester in three phases.[Figure 2.1] In the kick-off phase we tackled small visual aspects of the Explorer,this phase was useful for us to get familiar with the stack we were going to work with during the project.
Our main goal after the feedback we received was to facilitate user's understanding of the Explorer. This is what the first implementation phase was dedicated to. We planned to implement :

- Improve visual elements on the page (logo, block demarcation, [...])
- Tutorial and tooltips
- Instance search improvement
- Data exportation feature

Next, we planned to not only focus on adding new features but also highlighting the existing ones, correct bugs and implement suggestions received from the mid-semester UX survey. Our plan for the second implementation phase was :

- Correct small errors users detected while doing the mid-semester survey tasks.
- Add a scrollbar and zoom button for the chain
- Display the nodes' status
- Display some statistics about the chain
- Give users the possibility to change the roster
- Make the instance instruction search more appealing

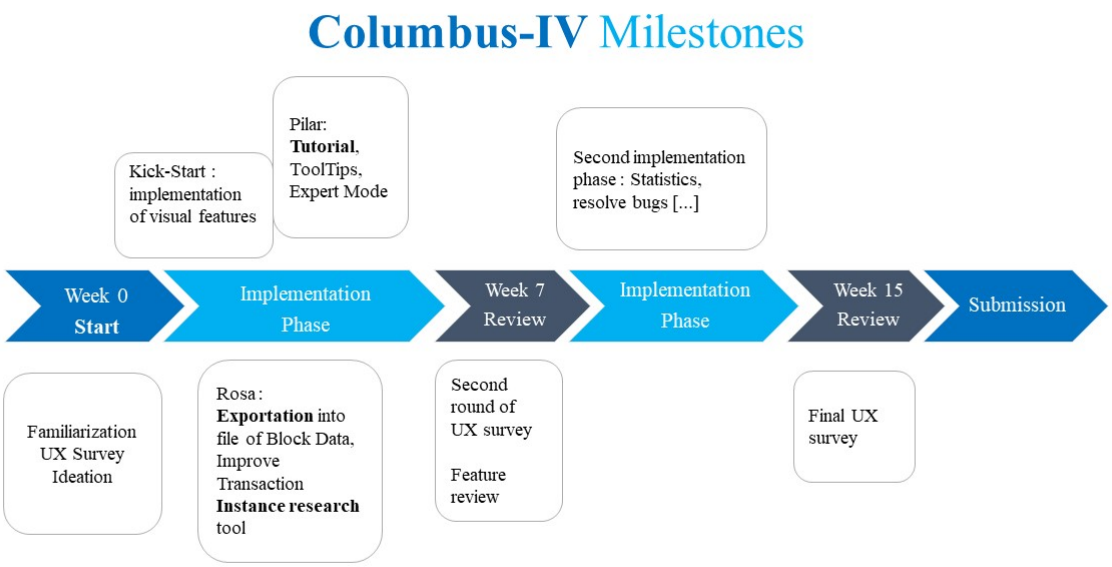



Figure 2.1 – Project Timeline

# Chapter 3

# First implementation phase

## 3.1 General changes

At the outset, we started by implementing simple visual features.

### 3.1.1 Logo and demarcation of the detail container

One of the first things we noticed when browsing the Explorer for the first time was that the logo color was not in accordance with the blocks color. We then unified the logo of Columbus, so that it was more in accordance with the color palette of the webpage.[Figure 3.1] The block details demarcation wasn't visible enough. We made those small changes for aesthetic purposes.


(a) Old (b) New

Figure 3.1 – Logo

## 3.2 New features

### 3.2.1 Instructions search

Each instruction deletes, creates or changes an instance. As explained earlier, an instance is an object of the class *Smart Contract*. So tracking an instance is similar to tracking a contract. Last

semester, Lucas Trogon implemented the Instance Tracking feature so that users can track the evolution through the blocks of a contract. In the first implementation phase of the project, our goal was to improve the waiting time when launching a query.
In the previous version of Columbus, we had the possibility to search for the 10, 50, 100 first or all instructions related to an instance but searching could be very slow when the first instruction of an instance were far from the beginning of the chain. We then came up with the idea to search for the X previous instructions related to a instance instead and launch the query from the current clicked block.
This update made the instances browsing more fluid. Accessing the instance tracker interface is easier since the loading time was reduce and users can then track instances as they are supposed to. This feature will be further improved during the second implementation phase.

### 3.2.2 Data Exportation

Data exportation was frequent others Explorers but not in Columbus. So we decided to implement it in our Explorer. Users would be able to access and download blocks data through an extra JSON file.
First we argued on the icon and visualization aspect. We wanted the icon to be meaningful so that any user knows exactly what would happen if he clicks on it. We also wanted to place it somewhere one would find it easily.
Secondly we talked about the generated file format. We needed this file to be as representative as possible of what is important to know about a block (e.g index,hash, transactions,instructions and so on...).[Figure 3.2]

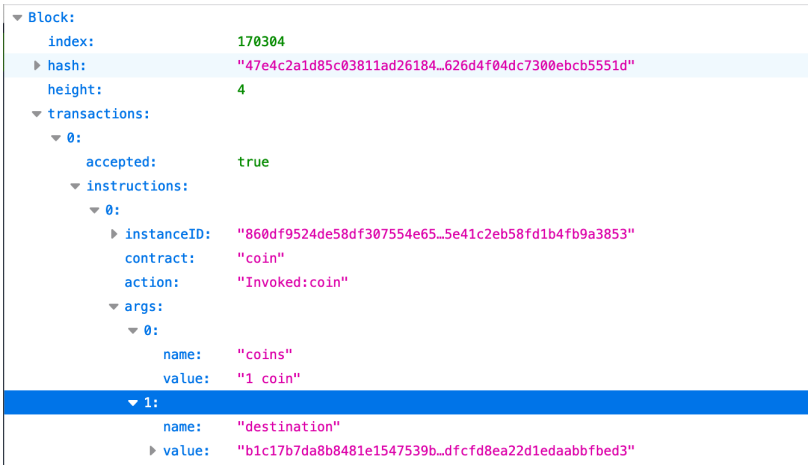



Figure 3.2 – JSON file format for the exported block data

Finally, the idea of extending this feature to the browsed instructions related to an instance came up. That way, advanced users would have the possibility to keep a record of the evolution of an instance by inspecting the downloaded JSON file and by going through the instruction blocks visually on Columbus.

### 3.2.3 Tutorial

One of the main issue Columbus was facing was the onboarding of new-comers. To address it we decided to add a guided tour that takes the user around the page, showing him the individual parts and what functionalities are available.
We used the *Intro.js* [10] library, which turned to be the easiest to include in the existing implementation, furthermore the visual aspects were in accordance with the Explorer.
The tour starts when clicking on the button *Get Started*, which was named *About* before [1]. The user is then introduced to the different parts of the page : the Skipchain visualization along with its interaction, the block and transaction details, the instruction search [...].[Figure 3.3] We invite curious users to visit the Github Repository of Byzcoin to learn more about the distributed ledger. Overall, the main goal of the tutorial was to present how to use the key functionalities of the Explorer.

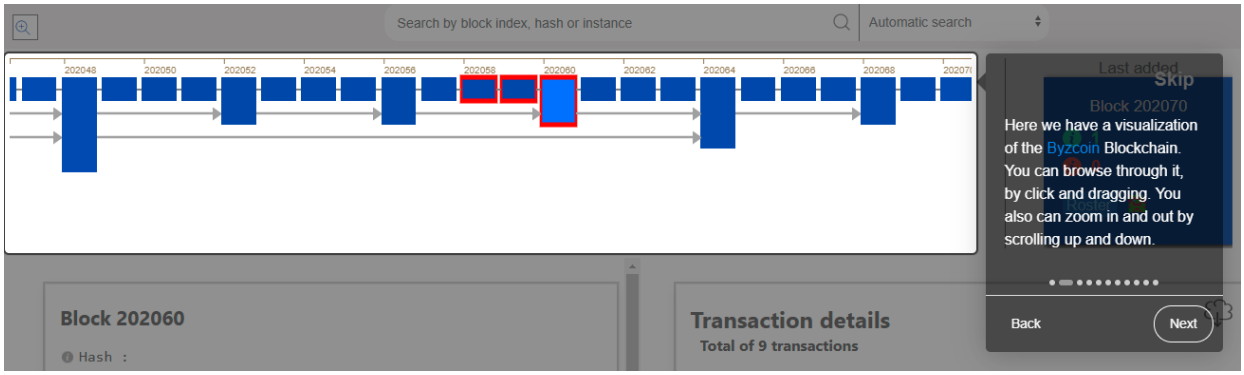



Figure 3.3 – Step 1 of the tutorial.

### 3.2.4 Tooltips

In continuation to what had been done in the tutorial, we wanted to help novice users have an understanding of the technical terms related to blockchain (e.g."Hash", "Block Height") in general as well as the terms specific to Byzcoin (e.g. "Forward Link", "Invoke"). We decided to use tooltips [3.4] that appear when hovering over the information logo, since they are unobtrusive to expert users (who are already familiar with the technical terms of blockchain).

[1] The latter gave a brief explanation of what was the purpose of Columbus and how to use it.


Figure 3.4 – Definition of a Hash

# Chapter 4

# Intermediate UX research and adjustments

## 4.1 UX survey

To evaluate our first round of implementation, we created a Google Form and invited the DEDIS lab members along with fellow students to test the interface.
First we asked them questions related to their background and their knowledge of Byzcoin and blockchain in general. We also wanted to know if they had some experience with blockchain Explorers (be it Columbus or others) and what type of tasks they would do using them. We then asked them to perform a set of tasks including :

- Going through the guided tour.
- Searching a block and downloading its information; doing the same but when browsing related instructions to a Contract.

Participants would then evaluate the interaction, and we invited them to give feedback on their general user experience. Finally, we asked them questions related to new features that would be "nice to have".

### 4.1.1 Summary of Feedback

In the following points we summarize the main insights we received.
The most common activities users would do when using Columbus are :

- "Check the content of blocks and transactions."

- "Browse (main search bar, instances), debug contracts, etc."
- "Get familiar with the blockchain."

The most common frustrations:

- "Font-size and how the information is displayed takes too much place, can not see information in one shot, doesn't always work with chrome for example."
- "When searching for a block, wouldn't it be better to make it appear "in the middle" of the interface ?"
- "a lot of "Cannot find backward link" errors fill the screen."[1]

The proposal for new features that were the most up-voted were : display of the status of the roster's node, some statistic on the transaction history of the Skipchain, and some statistic on the time spent mining the last block. Users also uttered that giving more flexibility to the instance search would be useful.

## 4.2 User scenarios

### 4.2.1 An interface for different type of users

One of the main challenge in the beginning of the project, was the lack of knowledge about who would be using the Explorer and what were the goals of such a user.
In human-computer interaction design it is important to understand the goals a user has when using the product you design, in order to create a interface satisfying his needs.
From the feedback we received in the different phases of the project, we could start building more precise user scenarios, and implement adapted features.

### 4.2.2 Scenarios

In interaction design we work with *Personas*, they are a user model that is represented as a specific individual human being [1]. They are not actual people but are synthesized directly from observations of real people, they represent a class or a type of user. During this project we focused on the experience of two classes of users: novice and experts.

A novice user, is someone that just discovered Byzcoin and may start using it for a personal goal. He does not have a technical background on the mechanics of Byzcoin, thus the

[1] Error of deployment, was resolved in the course of the survey.

explorer should help him have a better understanding. His main task on the explorer is checking some transactions (related to one of his personal goals). The guided tour and tooltips, help him get around the page and understand its individual parts.

An expert user, is typically an engineer working with Byzcoin. When he uses the explorer he can use all functionalities without needing a proper introduction to them. He knows the technical terms. His goal when using the explorer is to debug contracts and blocks, or tracking instances. Thus he needs a maximum of technical information about the blocks along with advanced features to perform his tasks easily.

One of the main insights we gained from the user surveys, is that it is important to interview people that have self-interest in the explorer. Interviewing fellow students for example was useful to evaluate visual elements, but not for proper functionality testing. It would have been great to be able to have more elaborated interviews. In person for example, and observe live their interaction with the explorer. Since Noémien knows about the functionalities that are needed by the engineers he helped us finding our way and implementing new functionalities for them.
Regarding intermediate users, we imagine them to use Byzoin as a service for one of their tasks (e.g. at their workplace). They have a mental model of how blockchains work but no technical background on them, they are not necessarily engineers. We did not have a chance to interview this type of user, but this could typically be an idea for what could come next for Columbus.

## 4.3 Adjustments

At the end of this phase, we could update our initial feature list. We included the suggestions we received from the participants, and we decided to focus on implementing a more flexible instruction search along with advanced features to make the experience of expert users more valuable.

# Chapter 5

# Second Implementation Phase

## 5.1 Implementations related to the UX survey

### 5.1.1 Block Search

A few users pointed out, that when searching a block from the search-bar the block was not loaded in the middle of the Chain (but on the left), which was violating their expectation. Thus, we corrected the position of the searched block in the chain.

### 5.1.2 Page Layout

As multiple users suggested how information is displayed was taking to much place, and users were not able to see in one shot the block details. We changed the font-size of secondary information and additionally changed the display of the block such that users can have a full overview of the block details. Furthermore some features such as the tooltip icones were not showing up when using Chrome, this was due to the fact that the *svg* format was not supported and we could correct it easily.

### 5.1.3 Additional information for blocks

As one of the users suggested to us, we added the name of the command that was executed on the contract in case of *Invoke* (in the transaction details). Note that we do not show it when the command is *transfer* of a *Coin*, because the information shown is self-explanatory and adding the command would overcrowd the information shown.
Finally, we also display the set of nodes (roster) that validate the transactions of a block in the

block details.

## 5.2 Implementations related to the Instruction Search

One of the main goals of the second implementation phase was to improve the Instruction Search : the search interface , the loading screen and instructions block interface(instance tracker interface).

### 5.2.1 Instance search interface improvement

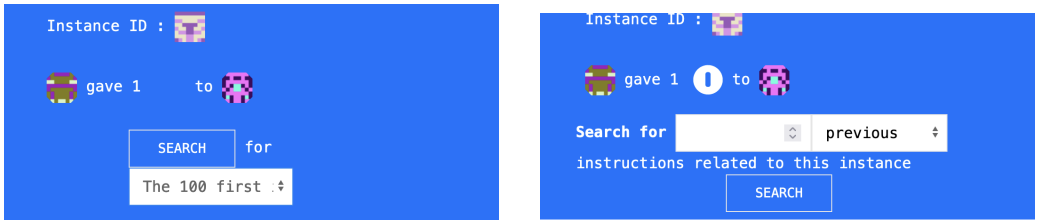
From the feedback we received, many users raised the same points :

- Searching is too restrictive ( the research is limited to the 10,50 or 100 first instructions related to an instance)
- The design of the search design is not optimal, many users launched a query without knowing what they searched for.

In the updated version of the Explorer, we give more freedom to users when launching a query. Users can now :

- select the exact query number
- track the next or previous instructions of an instance form the current clicked block
- track first instructions of an instance

All these possibilities improved the search flexibility and the user experience with the Explorer since he is not limited by the interface anymore (i.e. the search is completely customizable). The search interface design was updated according to users' feedback and is now more appealing.[Figure 5.1]



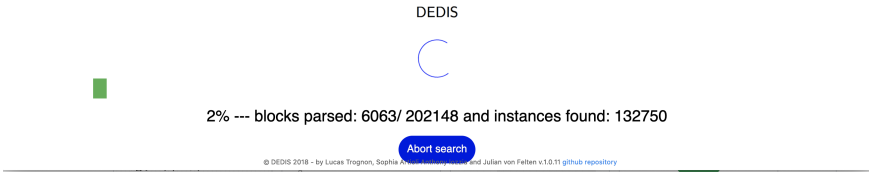

(a) Old loading interface

(b) New loading interface

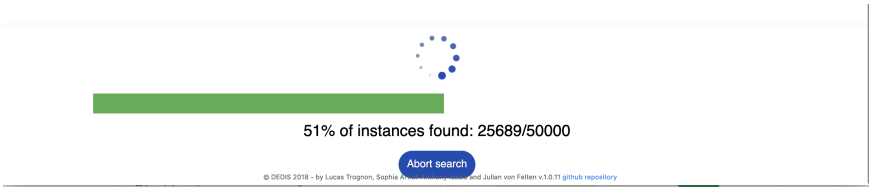
Figure 5.1 – old versus updated search interface

### 5.2.2 Loading screen improvement

The previous loading screen did not correctly represent the evolution of the query. The green loading bar level was based on the total number of block parsed which is not an important information one wants to have when searching for a specific number of instructions of a contract. We then changed the loading screen so that it looks less odd for beginners.[Figure 5.2]
First we changed the spinner and the abort search button color for aesthetic purposes.
Secondly we based the waiting time on the number of instructions found by the browser.
These are small changes but thanks to it users are less tempted to quit the loading screen when the loading is too slow.




(a) Old search interface




(b) New search interface

Figure 5.2 – old versus updated search interface

### 5.2.3 Instance tracker interface improvement

The third part of the instance search interface improvement concerned the instance tracker interface. The idea was to make the instance history look more dynamic similarly to what can be done in the Skipchain visualization.

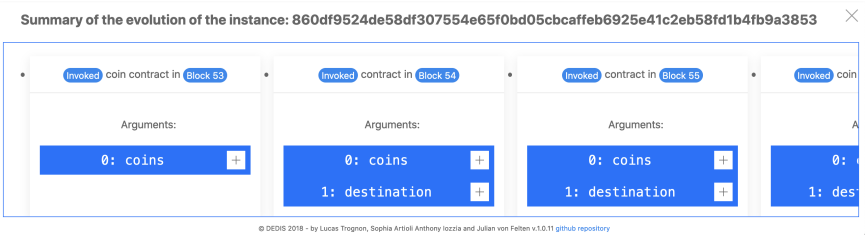
After launching a query and waiting for the browser to find every requested instruction, the user is automatically directed to the tracker interface at the bottom of the page (i.e. automatic scroll down). There he can visualize each instruction of a contract as a block. Each block contains relevant information about an instruction such as the block where it was found, the action type and its arguments.
Finding a more appealing and dynamic way to present these information was not easy. We completely changed this interface three times.
At first we didn't touch to the the instruction block's aspect. We make the visualization more dynamic by adding blocks one by one according to the user scrolling position. We were not satisfied with this first proposition. It was a bit minimalist and didn't improve the instance tracker visibility as it was supposed to. We then transformed the interface entirely and came up with the following design:

- To make the visualization look less uniform, we represented each contract type by a color. There is a legend in the visualization to support this idea.
- Blocks are rectangular shaped as in the Skipchain and have the color assigned to their contract type.
- On click on a block, the information it contains are displayed on the right of the page.
- User can zoom and scroll left and right to load more blocks.

This design was more challenging to implement but we got the feedback from our supervisor that it was too similar to the Skipchain visualization and that it can confuse the user. So we proposed a third and last design:

- Instruction blocks are more squared shape and its information are directly appended on it.
- There is always a color code
- User can still play with the zoom and scroll to load more blocks.
- When blocks are too small (zoom out), users are able to temporarily display the block information by hovering on it and make the information fixed on click.
- The number of loaded blocks is displayed so that any user knows when he is at the end of the chain according to the number of instructions he has requested.

This third design convinced everyone. It is dynamic, aesthetic and visually appealing. Users are more tempted to play with the interface and interested to check the instructions block information.[Figure 5.3]
A new class called `InstructionChain` has been added to the code. It entirely describes the instance tracker interface and the content of each instruction block.




(a) Old instance tracker interface


(b) New instance tracker interface

Figure 5.3 – old versus updated instance tracker interface

## 5.3 New Features

### 5.3.1 Tools for the Chain

In the first phase of evaluation of the project, we realised that novice users sometimes omitted the zoom (done by scrolling horizontally) and drag functionalities on the chain. So we added a scrollbar and zoom button (on the top left). These elements help users notice that they can interact with the chain, be it with through buttons or through the initial behavior i.e scrolling and click-and-dragging.

### 5.3.2 Roster

Columbus should be able to connect to different Skipchains of the Cothority. We decided to add a functionality that enables the user to connect to a Skipchain from another roster. This functionality can also be found in the Skipchain explorer *https://status.dedis.ch*.
User can click on the *Roster* button and enter a roster.toml file [Figure 5.4], by default the first Skipchain (index 0) of the roster is loaded. If necessary, in the future a second prompt could be added to select a specific chain.

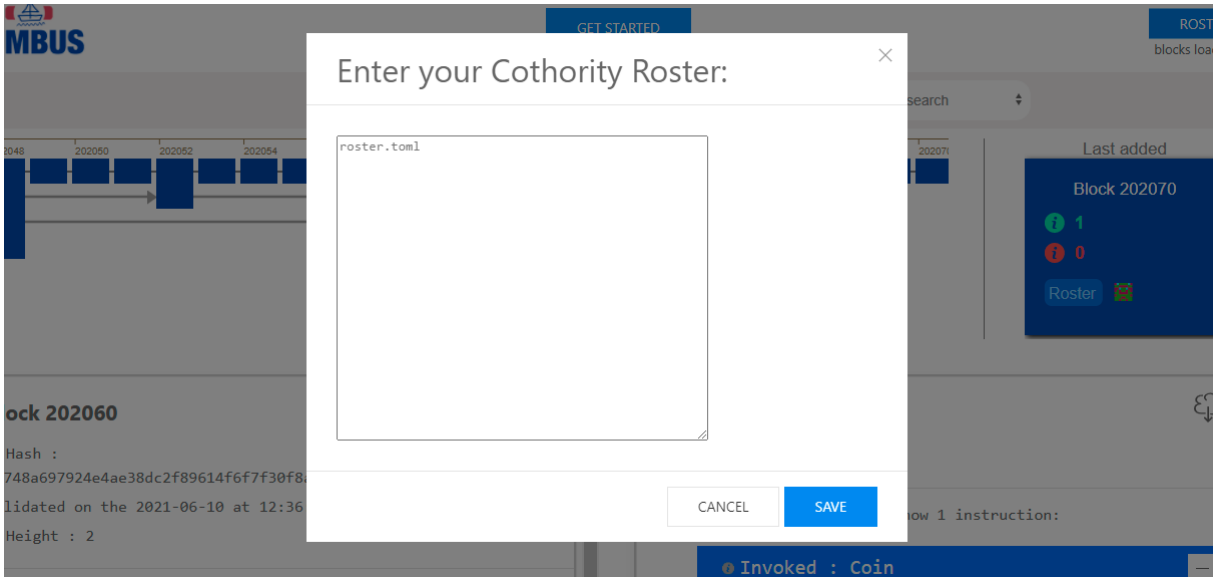



Figure 5.4 – Pop-up to connnect to a new Roster

### 5.3.3 Status of the Skipchain

To complete the information shown to the user, we wanted to provide information about the status of the Skipchain and it's current state along with statistics on the blocks of the Skipchain.[Figure 5.5]
To do this we added a new section underneath the chain visualization, this also gives a sense of "fullness" to the page (before half of the page was blank on arrival). When a user clicks on a block, the statistics go down the page, and thus are not bothering when checking the block and transaction details. All elements related to the status of the Skipchains were implemented in the new class *status*.

#### Status of the Roster's nodes

In the survey lab members confirmed that it would be helpful to them to be able to have the status of the roster's node. Again,similar to what is done on *https://status.dedis.ch/*, we implemented a table showing the *name* of the node, as well as the name of it's *host* and the *uptime* (rounded to days, hours or minutes). The name of the nodes that are up is written in green, whereas the nodes that are down are written in red. For expert users, we added a tooltip when hovering over the name, so that they can see the details of the status (connection type, port, version and traffic rate). The details are updated every 10 seconds, this is especially useful to keep track of the traffic rate (tx/rx).

### Statistics

Along with the status of the nodes, we show a graph plotting the number of transactions of the last 1000 blocks (x-axis being the block-id and y-axis the number of transactions). This gives the user another perspective of the content of the Skipchain. Underneath we give give the number of accepted and rejected transactions, along with what type of contracts were used.
We chose to show 1000 blocks, because this was feasible with the current implementation. In order to have more elaborated statistics (e.g. about the whole chain), Columbus must keep track of them all the time or receive them from a third-party, both are not possible yet given the nature of the code and how the information is retrieved.[Figure 5.5]

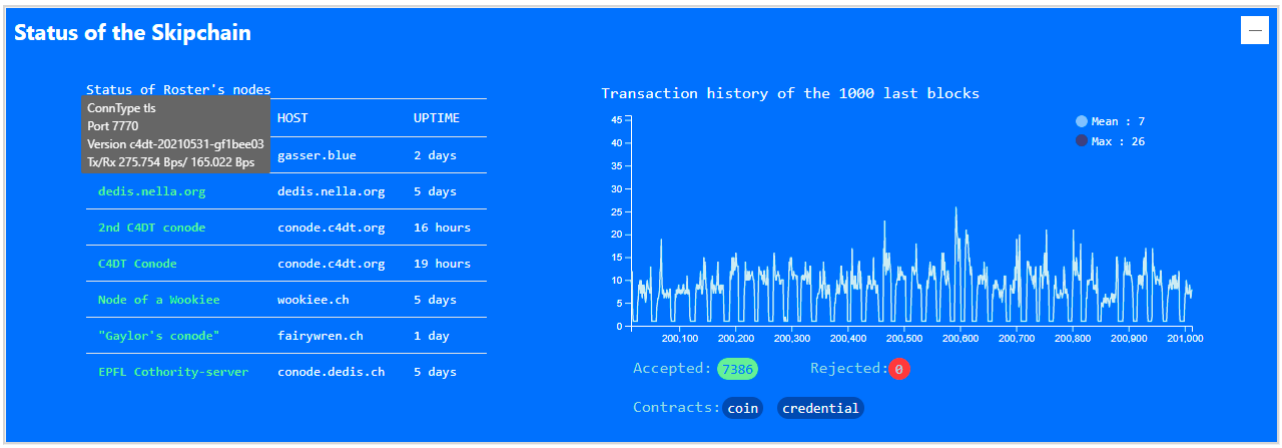



Figure 5.5 – Status of the Skipchain

# Chapter 6

# Conclusion

## 6.1 UX validation

To validate our second implementation phase we created a second Google From, and again we invited lab members and fellow students to test the new interface. Unfortunately, due to the short time span and the exam session, we did not receive any feedback from the lab members. Thus we evaluated the visual aspects with fellow students. We showed them the old Columbus interface for comparison, and explained to them the new features we implemented. Their main feedback was that the interface is now more appealing to use, and they are able to understand the main functionalities easily.

## 6.2 Ideas for the Future of Columbus

We managed to implement all the features we had planned in the beginning of the semester. Still we think that this project could be enhanced further especially for users outside of the DEDIS lab, if Byzcoin would be used in a larger scope for example. As mentioned in the user scenarios, the features we implemented focused on the experience of novice and expert users. One could imagine to have more features for intermediate users such as :

- having a *User tracker*, where they could check their transactions and thus also having a search mode by *User ID*
- Some sort of overview of their browsing history
- User-centric statistics

We think that to assess the real needs of such users, it would be best to interview 4 to 5 members

and assess together what would improve their experience with Columbus.

## 6.3 Personal retrospectives

### 6.3.1 Rosa

I started this project without any knowledge about Blockchain nor front-end development. When browsing Columbus for the first time, I clearly was lost and asked all questions a beginner can ask such as : what do these blocks represent? Why do they have different height? What is a transaction? and so on... I needed to read the project documentation many times to get familiar with Byzcoin and the coding utils. It was very hard to get into the project but coming up with an organized list of features and seeing our work validated and valued by engineers through feedbacks were a real motivation source. Furthermore, Noémien(assisted by Sophia Artioli) gave us enough freedom during the project which made us gain in confidence and allowed us to develop our creativity and front-end skills. Working on this project in DEDIS lab was an enlightening experience. It gave me an overview of what working as engineer would look like. It is not only about having front-end coding skills we also need social interaction with the users and to consider our collaborators feedbacks. Collaborating with Pilar this semester was enjoyable. We were complementary enough to learn from each other.
One thing I would have done if I first had front-end knowledge is that I would have made a representative draft of my ideas and put on paper also the steps to implement it using the stack. As novice we don't realize the time the implementation of each feature can take. Some features implementation can be very time consuming this is the reason why organizing our ideas can be more than helpful.

### 6.3.2 Pilar

Prior to working with Columbus I had no experience with front-end development, but I had a vague idea of how blockchains work. When discovering the first time Columbus, I was a bit lost alike the novice users we interviewed. I then started to get familiar with Byzcoin and the Cothority, reading the report of the previous students that worked on the project was also really helpful. In parallel to this project I was taking the course *Interaction Design (CS-486)*, which was really helpful for the UX research phases of this project.
I would say that in the first part of the project (first UX research and first implementation phase), it took me some time to understand the goal of our work and to get accustomed to the interface and the programming language we were working with. I think that the intermediate feedback really helped us evaluate our work and understand what was missing. Collecting feedback from members of the lab, was not easy, very few of them answered our surveys. Maybe it would be worthy to find another method to evaluate the work we have done. From my experience in

other projects, personal interviews are the most effective way, even if they can be more time consuming. The second implementation phase was much more fluid for my part. The help of Noémien and Sophia throughout the course of the project was really valuable and helped me move along. Overall, this project was really exciting and seeing the new features appear along the way was motivating. I really learned a lot be it from the technical side as well as from the evaluation of the UX research, this project really was versatile. Surprisingly, implementing new features turned out to be much easier and less time consuming than adding simple features to the already existing functionalities. Working along with Rosa was a pleasure, we had a really good communication and collaboration.

# Bibliography


[1] Alan Cooper, Robert Reimann, David Cronin. *About face 3, the essentials of interaction design* Wiley Publishing, Inc, Indiana, 2007.

[2] DEDIS lab. *Cothority repository*, https://github.com/dedis/cothority

[3] DEDIS lab. *ByzCoin repository*, https://github.com/dedis/cothority/tree/main/byzcoin

[4] DEDIS lab. *Skipchain repository*, https://github.com/dedis/cothority/tree/main/skipchain

[5] DEDIS lab. *Conode repository*, https://github.com/dedis/cothority/tree/master/conode

[6] DEDIS lab. *Columbus IV repository*, https://github.com/dedis/columbus-united

[7] DEDIS lab. *Development branch of Columbus explorer*, https://wookiee.ch/columbus-dev/

[8] *D3 repository*, https://github.com/d3/d3/wiki

[9] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, Bryan Ford. *OmniLedger*. https://eprint.iacr.org/2017/406.pdf

[10] *IntroJs documentation*, https://introjs.com/docs

[11] Sophia Artiolli, Lucas Trognon. *Columbus United*, https://www.epfl.ch/labs/dedis/wp-content/uploa

[12] *Introduction to reactive programming*, https://gist.github.com/staltz/868e7e9bc2a7b8c1f754

[13] *RxJS documentation*, https://rxjs-dev.firebaseapp.com/guide/overview

[14] *Uikit documentation*, https://getuikit.com/docs/introduction