Columbus - Implementation of an intuitive and insightful blockchain explorer

Anthony Iozzia

Computer Science

Decentralized and Distributed Systems (DEDIS) lab

Bachelor Project Report

June 2020

Responsible
Prof. Bryan Ford
EPFL / DEDIS

Supervisor
Noémien Kocher
EPFL / DEDIS

Contents

1 Abstract

The goal of this project was to implement an intuitive and insightful blockchain explorer and use it to visualize the ByzCoin. We wanted to build an intuitive interface, as easy as possible to use for every user. This visualizer should implement basic functionalities to efficiently browse a blockchain, such as blocks visualization, blocks navigation, easy breakdown of blocks contents and following instances.

2 Vision of the project and prototype

2.1 Current tools and limitations

Currently, DEDIS has the following tools to explore the blockchain:

- Columbus explorer (https://wookiee.ch/columbus): a simple interface that shows a specific number of blocks with or without details
- Skipchain explorer (https://status.dedis.ch): a more elaborate interface with measurements and a vertical graph
- CLI (command-line interface) bcadmin (https://github.com/dedis/cothority/tree/master/byzcoin/bcadmin): navigation using commands

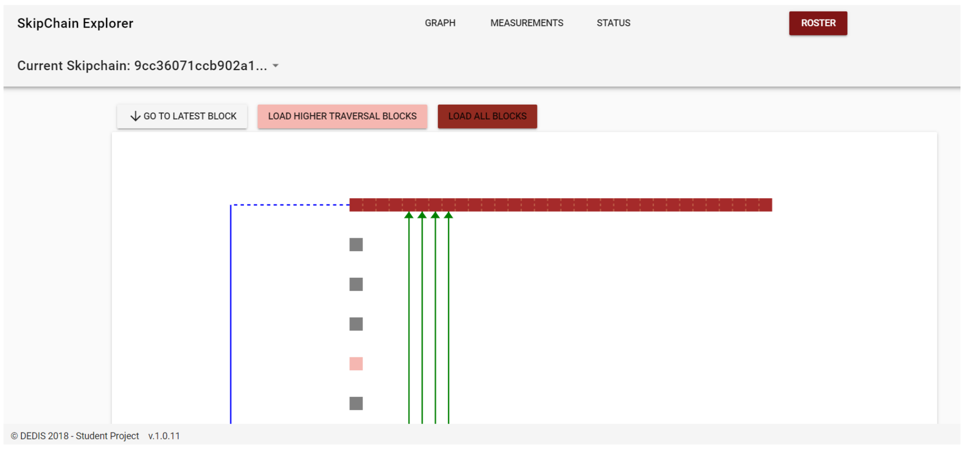

SkipChain Explorer
GRAPH
MEASUREMENTS
STATUS
ROSTER
Current Skipchain: 9cc36071ccb902a1...
GO TO LATEST BLOCK
LOAD HIGHER TRAVERSAL BLOCKS
LOAD ALL BLOCKS
© DEDIS 2018 - Student Project v.1.0.11

Figure 1: SkipChain Explorer

The current tools are limited in the following ways:

- Performance
- Fluidity

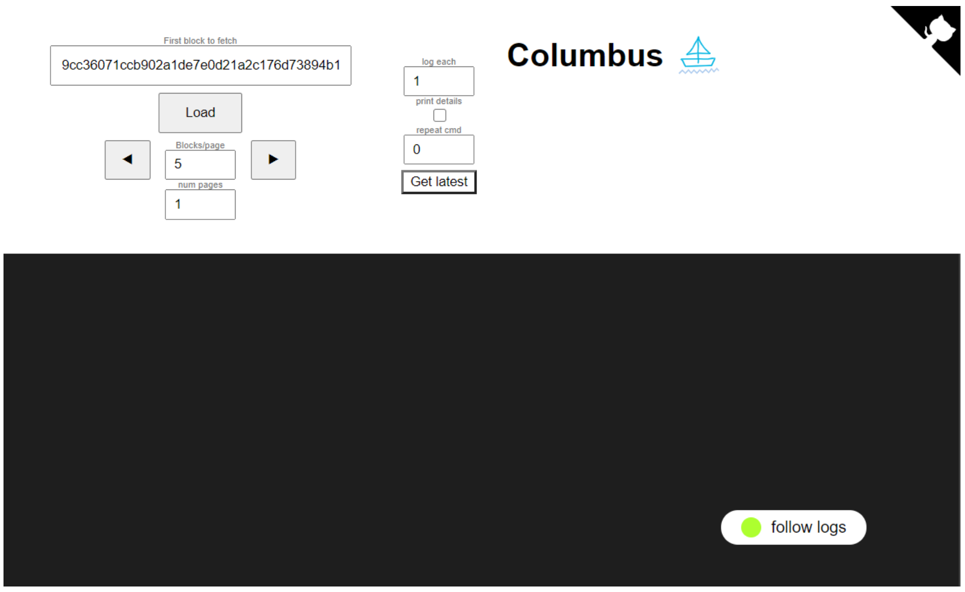


Figure 2: Wookiee

- Usability
- Interactivity
- Functionalities (global visualization, following instances)

2.2 Inspiration

The Ethviewer blockchain explorer inspired us for the design of our project.

2.3 Structure of the interface

The structure of the interface is described in figure 4. At the top, we have the diagram of the blockchain, then the blocks which contain transactions which contain instructions. The instructions contain the directives that manipulate a smart contract. I worked on the blockchain diagram, while Julien worked on the blocks informations visualizations.

2.4 Assets of this project

The goal is to provide the missing features in an all-in-one intuitive interface of the blockchain. Here are some functionalities that we wanted to implement: Anthony's features:

- A dynamic and intuitive visualization of the blockchain that is "infinite": load blocks while going through the blockchain; and show dynamic information

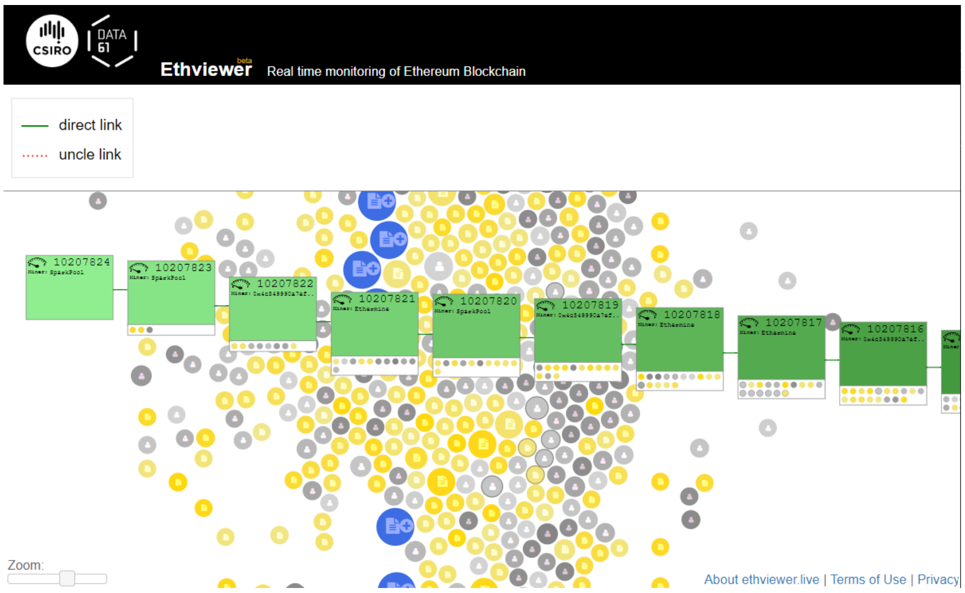


Figure 3: Ethviewer

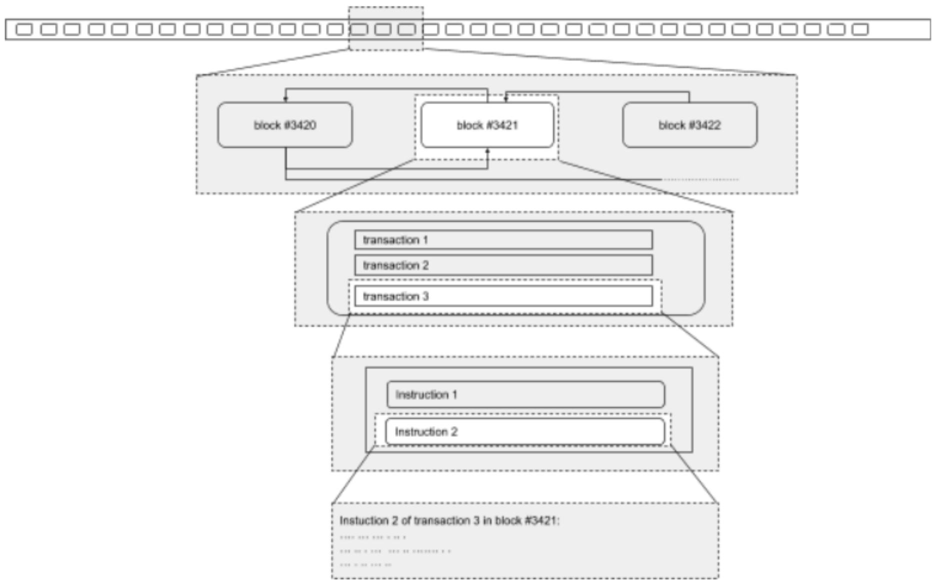


Figure 4: Desired structure of the interface

- Zooming through and navigating in the blockchain
- Visualization of the backlinks and forward links

Julien's features:

- Following instances: e.g. highlight all the blocks involved in a specific smart contract

- Show details of the blocks

Also, we thought of some ”nice to have” functionalities:

- Search (by block index, by hash, by instance, etc.)
- Undo and repeat
- Statistics
- Dark mode

2.5 Tools used

This project is implemented in TypeScript, using the d3js library.

2.6 Description of the interface

Here is my vision at the start of the project:

The interface is composed of the following elements:

- A top bar with the name of the app, menus and search
- The search bar will be able to search block numbers, hashs, verifiers IDs, instances...
- The horizontal visualization of the last created blocks, with the most recent block at the left
- Some controls such as:
  - A zoom slider with + and - buttons
  - Reset zoom
  - A slider to scroll through the horizontal blockchain
  - Undo and repeat
  - Go to first block
  - Go to latest block
- Some statistics with nice data visualization (nice to have)

Operating mechanics of the blocks visualization:

- The block number and date of creation are displayed on each block
- Ability to zoom and unzoom with the mouse-wheel

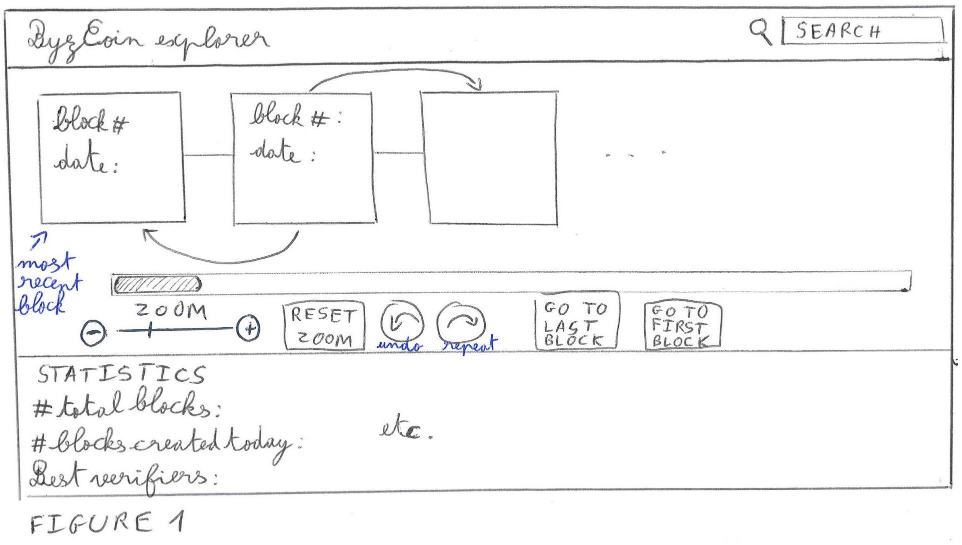


Figure 5: General look

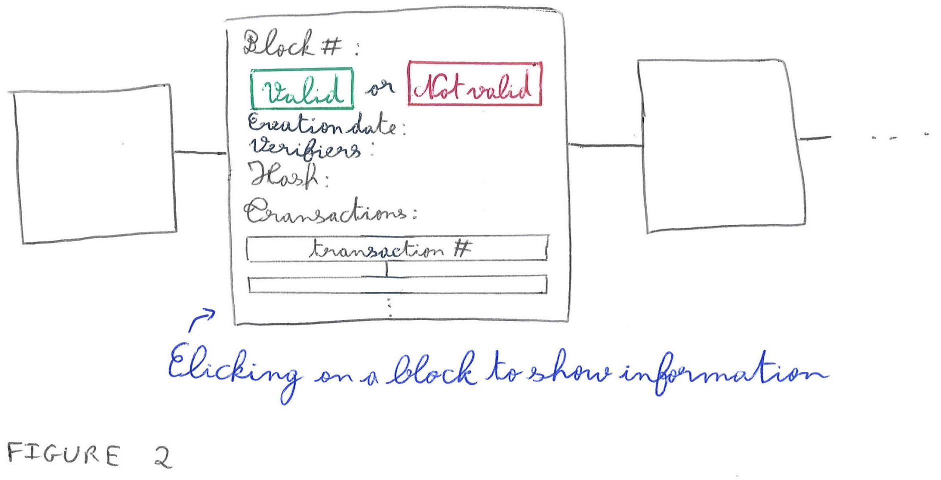


Figure 6: Block look

- Clicking on a block will horizontally expand the block and show informations:
  - block number
  - date of creation
  - hash
  - verifiers
  - transactions with IDs
- At first, we should be able to see block information by clicking on

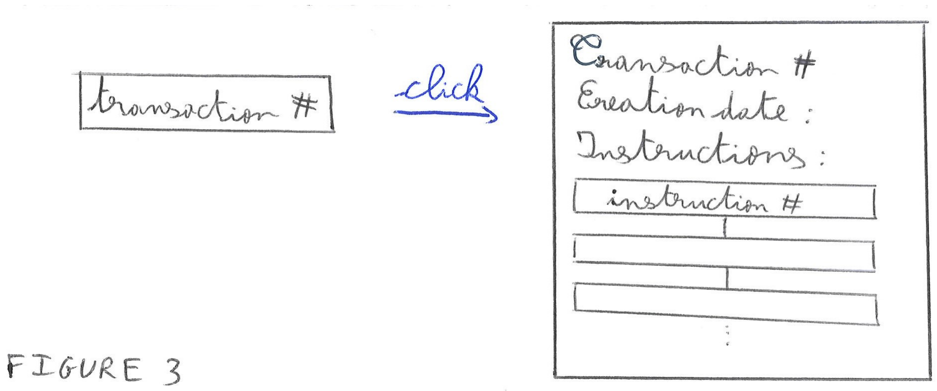


Figure 7: Transaction look

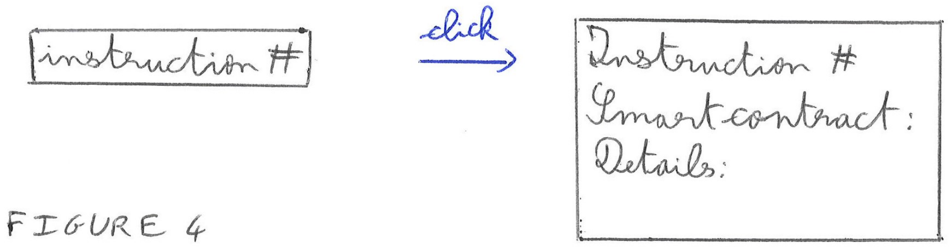


Figure 8: Instruction look

it. In the future, the information display should be automatic when zooming.

- Similarly, clicking on a transaction (in the future: zoom on a transaction) will vertically expand the block and show:
  - the transaction ID
  - the creation date
  - the instructions that compose this transaction
- Similarly, clicking on an instruction will expand the block and show:
  - the instruction ID
  - the name of the instance of the smart contract that is being manipulated
  - the details of this instruction
- Following instances: highlight all the blocks involved in a specific smart contract (for example: follow an instruction that manipulates a specific smart contract to highlight all similar instructions, transactions and blocks). For example, in the 9 the user chose to follow the instruction 2 of the transaction 3 of the before-last block.

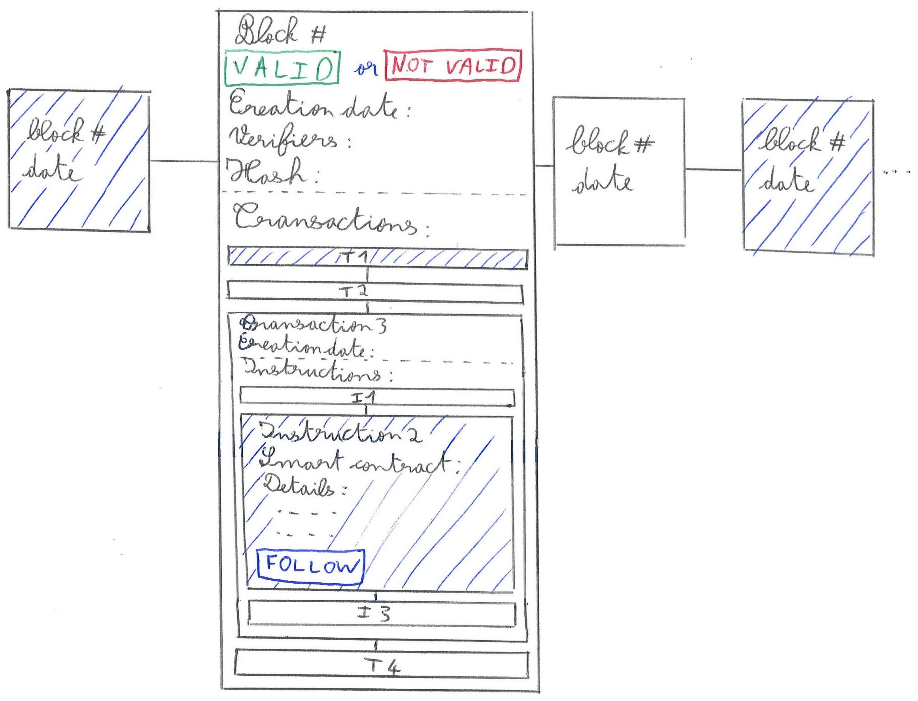


FIGURE 5

Figure 9: Detailed block look

3 Implementation

I worked the blocks diagram and Julien von Felten worked on the blocks informations and the search of instances. Here, I will explain the Implementation of the blocks diagram.

3.1 index.ts

Everything starts in index.ts. We set the initialBlockIndex and a function will browse the blockchain from the block 0 and search for the block with the desired index and return it. Once it is returned, we instantiate all the classes, including blocksDiagram.ts.

3.2 blocksDiagram.ts

Here is how it works. I used d3 to create a zoomable and scrollable SVG, which contains a chain of rectangles. First, in the constructor, I created the SVG by telling d3 to selects the class .blocks (defined in the index.html) which is a container for the SVG. I set the width and height. The width is simply the width of the window, and the height is 1/3 of the height of the window (this is modulable in the function computeBlocksHeight()). Then, I call the function zoom() on this SVG, where everything will happen. Each time the user translates (moves) the SVG or zooms or unzooms, it will trigger the zoom function. In it, I computed the indexes of the leftmost and rightmost blocks on screen in order to request new blocks to the blockchain when the user approaches an extremity of the already loaded blocks. To request new blocks, I use the function getNextBlocks. At the end of the class constructor, I created a subscriber that receives the new blocks from the server and calls the function displayBlocks (there is an attribute ”backward” that specify whether the blocks need to be displayed on the left or on the right). The function displayBlocks appends rectangles to the SVG using a for loop. These rectangles have the dimension (svgHeight * svgHeight) by default so that they fit in the SVG. Of course, when the user zooms or unzooms, this size changes. I also display the block index, hash and number of transactions on each block. Finally, a loader is displayed while we retreive new blocks. If the user has a slow Internet connection, the new blocks will be delayed but there is a loader animation to inform the user that the application is still working.

3.3 utils.ts

This class contains some useful functions:

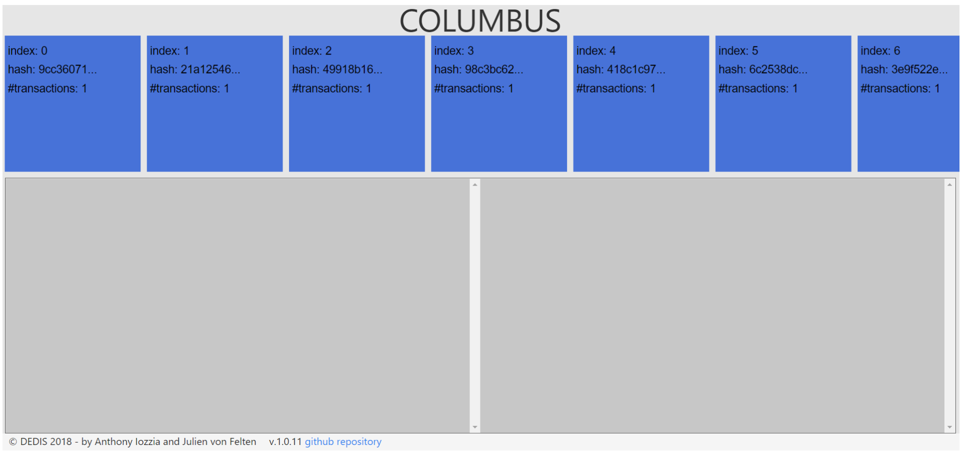
- bytes2String which converts a buffer to a string (useful for converting the hash of a block)
- hex2Bytes which is the inverse conversion
- getRandomColor which generates a random color (not used, but if we want rainbow blocks, we have a function for it)
- getBlockFromIndex which browses the blockchain to find a specific block
- getLeftBlockHash which retrieves the hash of the previous (left) block
- getLeftBlockHash which retrieves the hash of the next (right) block

4 Result

The figures 10, 11, 12, 13 show the final result of the explorer.

5 User tests

I did some user tests on my entourage (family and friends) and they globally enjoy the application (even if some of them did not understand what is a


COLUMBUS

index: 0
hash: 9cc36071...
#transactions: 1

index: 1
hash: 21a12546...
#transactions: 1

index: 2
hash: 49918b16...
#transactions: 1

index: 3
hash: 98c3bc62...
#transactions: 1

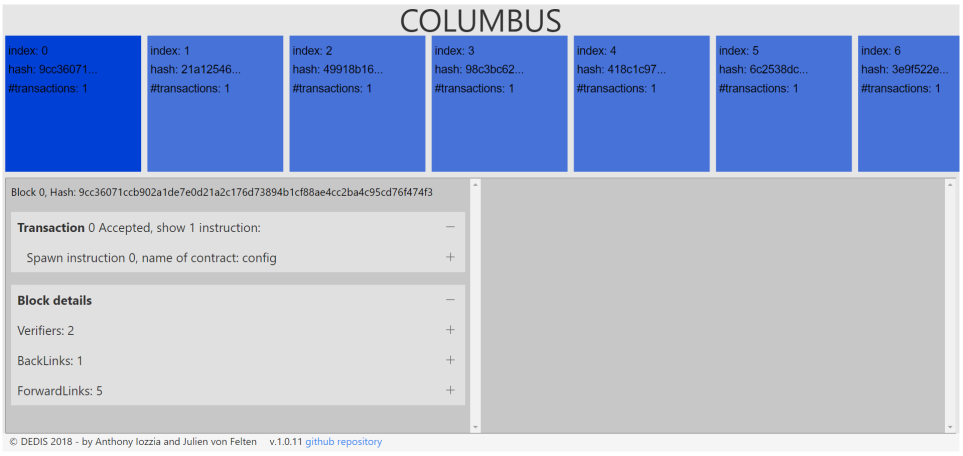
index: 4
hash: 418c1c97...
#transactions: 1

index: 5
hash: 6c2538dc...
#transactions: 1

index: 6
hash: 3e9f522e...
#transactions: 1

© DEDIS 2018 - by Anthony Iozzia and Julien von Felten v.1.0.11 github repository

Figure 10: General view


COLUMBUS

index: 0
hash: 9cc36071...
#transactions: 1

index: 1
hash: 21a12546...
#transactions: 1

index: 2
hash: 49918b16...
#transactions: 1

index: 3
hash: 98c3bc62...
#transactions: 1

index: 4
hash: 418c1c97...
#transactions: 1

index: 5
hash: 6c2538dc...
#transactions: 1

index: 6
hash: 3e9f522e...
#transactions: 1

Block 0, Hash: 9cc36071ccb902a1de7e0d21a2c176d73894b1cf88ae4cc2ba4c95cd76f474f3

Transaction 0 Accepted, show 1 instruction: −

Spawn instruction 0, name of contract: config +

Block details −

Verifiers: 2 +

BackLinks: 1 +

ForwardLinks: 5 +

© DEDIS 2018 - by Anthony Iozzia and Julien von Felten v.1.0.11 github repository

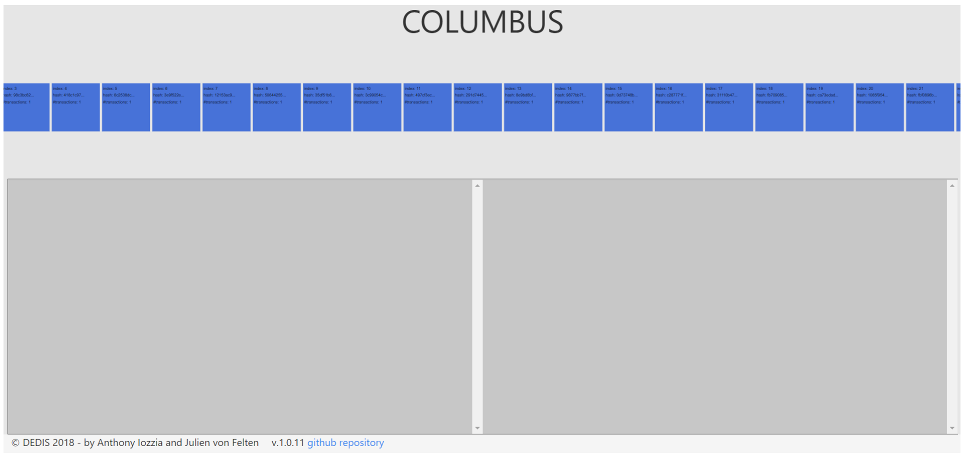
Figure 11: Block 0 with details

blockchain). Here are a compilation of their remarks:

- the highlighted blocks should be in a different color to see them better
- the text that popups when the mouse passes on the Columbus title stays on the screen for too long
- it seems that the application does not display correctly on the Firefox browser

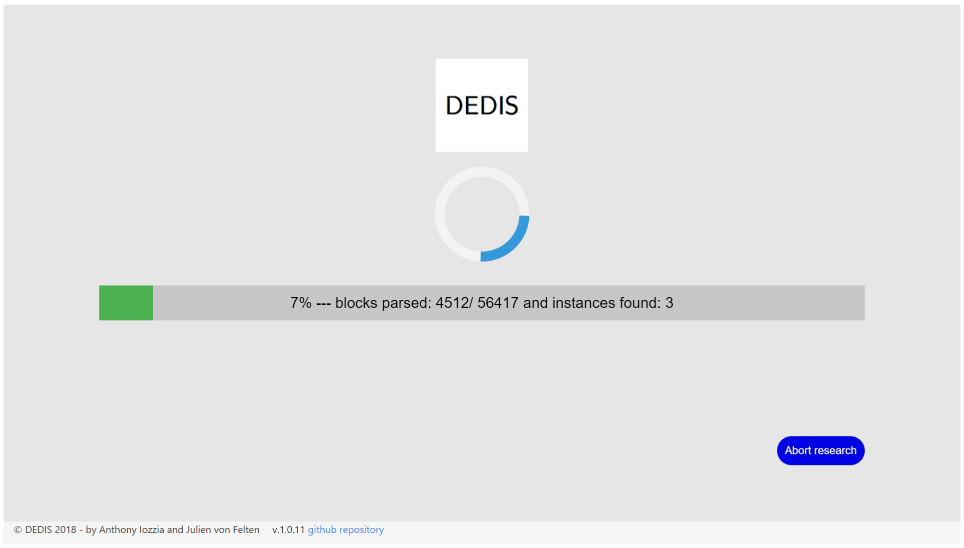
They also expressed that

- the minimalistic design is good
- the style is consistent
- the users controls are intuitive



© DEDIS 2018 - by Anthony Iozzia and Julien von Felten v.1.0.11 github repository

Figure 12: Unzoom



7% --- blocks parsed: 4512/ 56417 and instances found: 3

© DEDIS 2018 - by Anthony Iozzia and Julien von Felten v.1.0.11 github repository

Figure 13: Searching instances

6 Future of the project

Unfortunately, I did not manage to do every feature that I wanted because of time contraints. I would be happy that this project continues with future students because I really like the application that Julien and I did, and there is potential. For example, here are some missing features:

- forward links and backlinks
- search bar
- horizontal zoom like this: https://bl.ocks.org/larsvers/95115f57fb67ac8c0a568fdd28ae8c00

7 Conclusion

I would like to thank Noémien Kocher very much for proposing and supervising this project and also Julien von Felten that did a great work and also provided me his latex report template. I really enjoyed working on this project and I think it has a great potential. I will follow with great interest the GitHub page!

References

[1] Project repo (Columbus United): `https://github.com/dedis/columbus-united`

[2] Data-Driven Documents (D3js): `https://d3js.org/`

[3] Initial repo of my part of the project (merged in the columbus-united repo): `https://github.com/dedis/student_20_columbus_react`

[4] Julien's initial repo of his part of the project (merged in the columbus-united repo): `https://github.com/dedis/student_20_columbus_vue`

[5] Cothority repo: `https://github.com/dedis/cothority`

[6] Columbus CLI: `https://github.com/dedis/columbus-cli`

[7] Wookiee: `https://wookiee.ch/columbus/`

[8] SkipChain Explorer: `https://status.dedis.ch/#/9cc36071ccb902a1de7e0d21a2c176d73894b1cf88ae4cc2ba4c95cd76f474f3/status`

[9] Ethviewer - Example blockchain visualizer: `http://ethviewer.live/`

[10] ByzCoin - an innovative solution: `https://actu.epfl.ch/news/byzcoin-an-innovative-solution/`