University of Zurich[UZH]

BACHELOR THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

A Blockchain Explorer for Bazo

Luc Boillat
Zurich, Switzerland
Student ID: 14-715-577

Supervisor: Dr. Thomas Bocek, Bruno Rodrigues, Hamza Bedrija
Date of Submission: February 1, 2018

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

Bachelor Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
URL: http://www.csg.uzh.ch/

Abstract

Das Belohnungssystem eines Finanzdienstleisters besteht aus Bonuspunkten, welche beim gebrauch von Kredit- und Debitkarten gewonnen werden. Der Karteninhaber kann diese Punkte im Online-Shop des Dienstleisters gegen Waren und Gutscheine umtauschen. Dies bringt einen hohen administrativen Aufwand mit sich, da für jeden neuen Händler, welcher im Online-Shop seine Waren gegen Bonuspunkte verkaufen möchte, ein speziell abgestimmter Vertrag erstellt werden muss. Zusätzlich ist die Popularität des Services nicht erwartungsgemäss, da die Punkte nur in diesem einen Online-Shop benutzbar sind. Zusammen mit der Universität Zürich wurde darum die Bazo Kryptowährung entwickelt, welche eine dezentralisierte Verwaltung der Punkte und Konten ermöglicht. Dies hat den Vorteil, das Händler an ihrem eigenen PoS ihre Waren gegen Bazo Coins verkaufen können. Der einzige Kontakt, welcher die Händler mit dem Finanzdienstleister haben werden ist das Umtauschen von Bazo Coins in Fiat Währung. Die Bazo Software besteht aus zwei Kommandozeilen-Tools welche die verarbeiteten Daten der Blockchain zwar speichern, jedoch nur bedingt dem Benutzer lesbar präsentieren. Diese Arbeit dokumentiert das Design, die Entwicklung und die Evaluation eines Blockchain Explorers für die Bazo Blockchain. Der Explorer ermöglicht dem Benutzer über einen Webbrowser die Blockchain-Daten zu durchsuchen und grafisch darzustellen. Ebenfalls verfügt der Explorer über eine Benutzeroberfläche für Administratoren, damit Systemparameter für die Blockchain gesetzt werden können.

The reward system of a financial service provider, consists of bonuspoints, which can be amassed by using credit- and debit-cards. These points can be exchanged for goods and coupons in the online reward shop of the service provider. This causes significant administrative overhead for the provider, since for every merchant that wants to sell its products in the reward shop, a tailored contract has to be made. Additionally popularity of the shop is not as expected, due to the bonus points being only useable in this specific shop. Jointly with the University of Zurich, the Bazo cryptocurrency was developed to counter these disadvantages of the bonus point system. This enables a decentralized management of points and accounts, and permits merchants to sell their products at their own PoS for Bazo Coins. The only contact merchants now have with the financial service provider, is when they exchange their amassed Bazo Coins for fiat money. The Bazo software consists of two command-line interfaces which, by design, save the processed data of the blockchain. However only limited access to this data is possible. This thesis covers the design, development and evaluation of a blockchain explorer for the bazo cryptocurrency. The blockchain explorer enables users to display and browse through the blockchain data via a web-browser. Additionally, the explorer contains an admin-panel, where administrators of the system can set certain system parameters of the blockchain.

Acknowledgments

Optional

Contents

Chapter 1

Introduction

1.1 Motivation

The bazo blockchain application consists of two command-line tools called bazo_miner and bazo_client. Both tools are needed to run and interact with the blockchain. Every bazo_miner stores all the blockchain and state data in its built-in storage component, however there is no way for a user to browse through that data using a GUI. Information about the health and productivity of the system are not available either. This is why a blockchain explorer is needed, a seperate service that lets users examine the blockchain data, without directly taking part in the network.

1.2 Description of Work

This thesis documents the design and implementation of a blockchain explorer for the private blockchain bazo and its corresponding cryptocurrency bazocoin. The explorer allows users and admins of Bazo to inspect and analyze data regarding the bazo blockchain. Blocks, transactions and accounts are being displayed in an informative and well-structured manner, with the explorer acting as a visualizer for the blockchain. Statistical information about the blockchain will also be made available to the user. Furthermore the explorer features admin-only functionality, serving as a GUI for setting various system parameters from the web via a gateway to the Bazo network.

1.3 Thesis Outline

Chapter 2 introduces the bazo blockchain and analyzes existing blockchain explorers and statistics analysis platforms. Chapter 3 focuses on the design of the bazo explorer, consisting of both, the frontend and its corresponding backend. Chapter 4 documents the implementation of the web application, followed by an evaluation in chapter 5. A summary and conclusions are presented in chapter 6.

Chapter 2

Related Work

This chapter gives an overview of the bazo cryptocurrency and its underlying blockchain technology. Additionally it presents an analysis of blockchain explorers for 2 different cryptocurrencies, highlighting both similarities and differences in the implementation and functionality of the applications. The analysis plays a major role in the specification of the Bazo Blockchain Explorer, as it helps making design decisions for requirements.

2.1 The Bazo Blockchain and Cryptocurrency

Developed in 2017 at the University of Zurich, the Bazo cryptocurrency is a private blockchain that aims to reduce administrative overhead, as well as extend the functionality of a financial service provider's bonus point reward system. Traditionally, for each merchant who wants to sell its products in the rewards shop of the service provider, specific contracts between the two parties need to be made. This makes expanding the bonus point system a time and resource consuming process. Bazo eliminates this restriction by introducing a cryptocurrency which allows to directly make transactions between merchants and users or even between users itself using Bazo Coins. The merchants itself do not need to form contracts with the service provider anymore, they can offer their products in exchange for Bazo Coins even at their own PoS. The only interaction between the service provider and merchants consist of the exchange of Bazo Coins for fiat currency. Users of the bonus point system can exchange their current bonus points for Bazo coins. In order to access the blockchain, client and miner applications are available. Simultaneous to the development of the Bazo Blockchain Explorer, a light client and a payment app were developed as well.

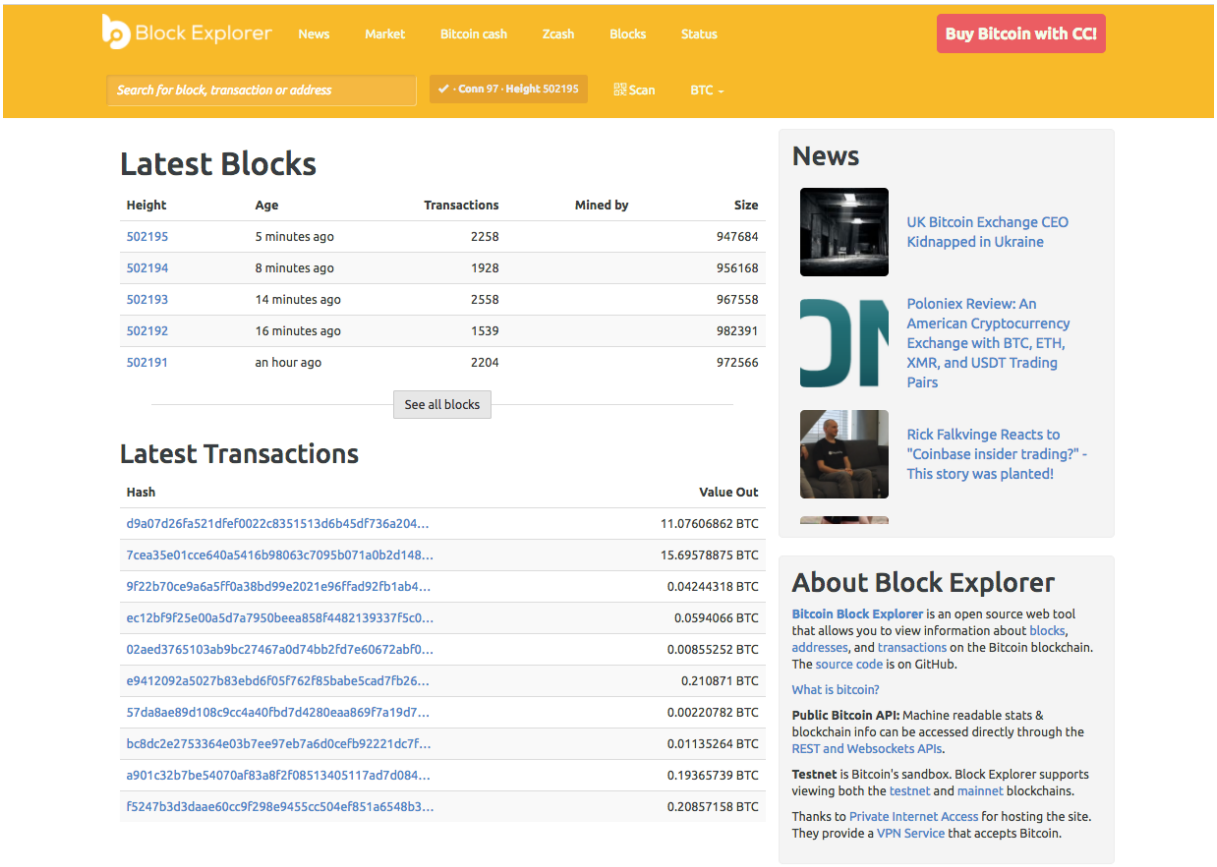

Figure 2.1: Landing page of blockexplorer.com

2.2 Blockchain Explorers and Analytics Platforms

2.2.1 Blockexplorer.com

This blockchain explorer was built for both the bitcoin and bitcoin cash blockchain. The frontend of the web application is called Insight UI and is built using AngularJS, a javascript framework. It interacts with the Insight API, the corresponding backend. Insight API consists of a REST and websocket API for Bitcore Node, a query and indexing service for the bitcoin blockchain. The source code for both frontend and backend are available on GitHub.

2.2.2 Etherscan.io

EtherScan is a block explorer and statistics analysis platform for the Ethereum blockchain. It uses Go Ethereum, an implementation of the Ethereum protocol in the Go language, in combination with Parity, a client for interacting with the Ethereum blockchain. EtherScan is a closed source project.

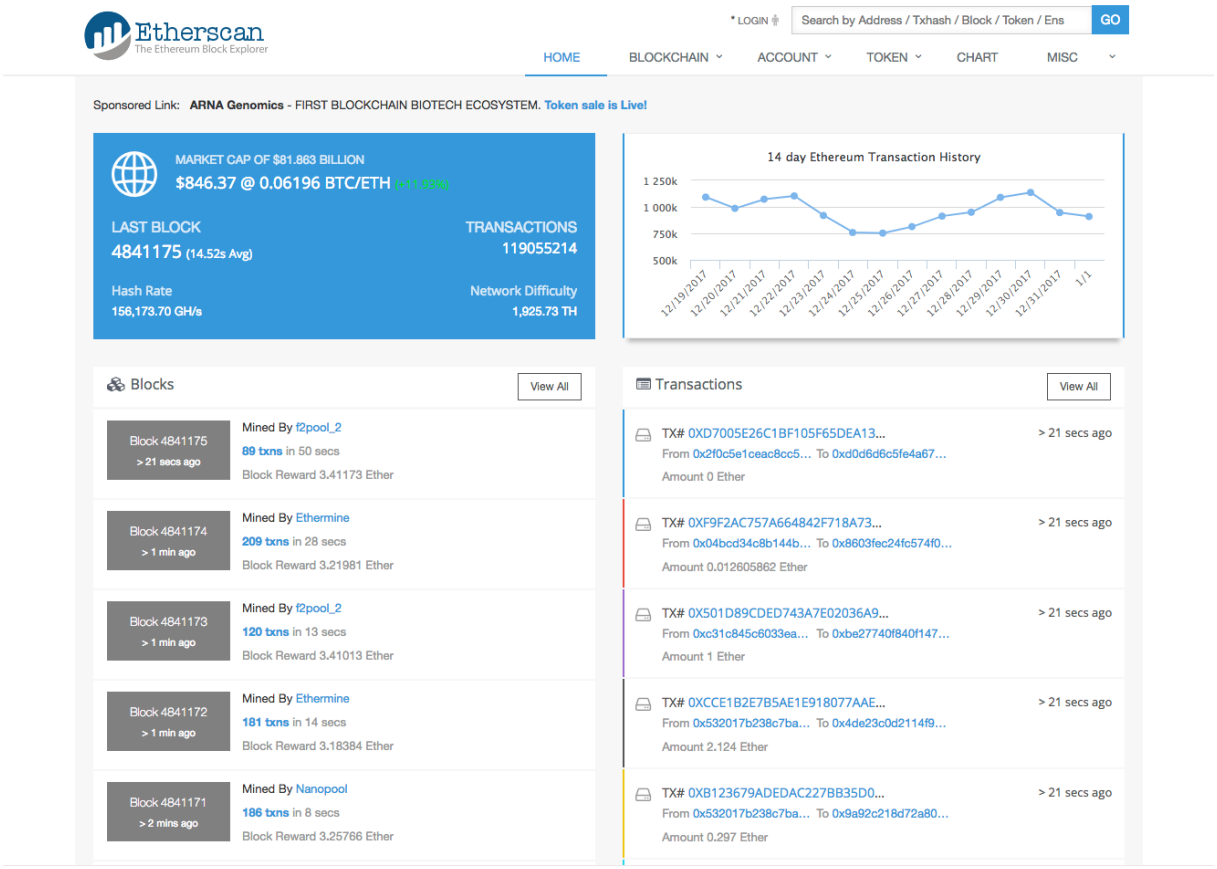

Figure 2.2: Landing page of etherscan.io

2.3 Analysis

This analysis omits features of the explorers that do not relate to the blockchain itself, such as newsfeeds of blockchain-related topics or social media links. Both explorers offer similar functionality as their core-feature: Structured views of blocks and transactions. The landing pages display the most recently mined blocks and transactions, with blockexplorer offering real-time updates. EtherScan also displays statistical data about the chain, such as the market cap, mining difficulty and hash rate. A search feature is present on both sites, offering the user to search for transactions, blocks and accounts via their respective hashes. To browse the chain, links are used extensively (e.g. every block on the landing page links to its respective detailed block page). When presenting multiple objects on the same page, such as a list of blocks, the data is structured using tables, in EtherScan's case using pages with a predefined length and in blockexplorer's case using a date picker that displays all blocks which have been mined on the chosen date. When multiple items are displayed using lists, less information about the items is given, compared to when a single item is viewed.

Chapter 3

Design

This chapter covers the design of the blockchain explorer and the additional components necessary to run the application.

3.1 Requirements for the Bazo Blockchain Explorer

Based on the analysis performed in 2.3 and meetings held with members from the financial service provider and the University of Zurich, the following functional requirements were elicited:

- **Blocks** A user should be able to view all validated blocks of the blockchain and the information they contain. In a list-view, the most recent blocks are being displayed, identified by their respective hashes and timestamps. If a user wants to get more comprehensive information about a block, he can display one block in a detailed-view, where additional information, such as all the transaction hashes contained in this block or the address hash of the block-reward beneficiary are presented.

- **Transactions** Similarly to the requirement above, a user should be able to view all validated transactions that have been broadcasted by him or other users of the blockchain. Due to the Bazo system having 3 different types of transactions (Funds Transactions, Account Creation Transactions and System Configuration Transactions), different implementations for each of them have to be made. A list-view displays the most recent transactions of each type, offering information such as the sender, receiver and the amount of the transactions. Detailed views for all transaction types are needed as well, presenting more information, for example each transaction's signature or block it is contained in.

- **Accounts** With Bazo using an account-based model, every user that actively interacts with the blockchain owns an account. The application should maintain a state of all accounts, that gets updated with every newly mined block. A list with the most affluent accounts should be made available, as well as a detailed view of a

single account, which displays all transactions this account was either the sender or receiver of.

- **Search** A user should be able to search for blockchain data using hashes. These hashes can be identifiers of blocks, transactions or accounts, with accounts having both addresses and address-hashes. A user should not need to choose what he actually searches for, meaning the application searches through all data it has collected and, in case of a hit presents the data associated with the hash to the user, and in case of a miss, notifies the user of not finding any relevant data.
- **Navbar** Featured on every page of the application, a navbar, that lets a user access all functionality of the website, is required. This functionality includes, among others, links to lists of blocks, transactions and accounts. The search-functionality is also located here, being available at all times to the user.
- **Statistical Information** The system should calculate statistical information about the network and make it available to users. This includes information such as a graphical history of transactions, or the total amount of Bazo Coins currently in the system.
- **Status**
- **Admin Panel** Only available to admins of the systems, a panel that lets them change system parameters using System Configuration Transactions has to be implemented. These transactions get sent via an interface to the network, meaning no bazo_client is running on the server of the website.
- **Fetch and Store Data** The application needs to automatically gather the latest Bazo Blockchain data and save it independently from the blockchain. This data needs to be processed, in order to make it publicly available.

3.2 Structure of the Service

The program that users and admins use to view blockchain data is a website, which is made up of a front- and a backend. However, to run the blockchain explorer on its own, additional components beside the front- and backend programs are required. The website fetches blockchain data from an SQL database that runs independently from the blockchain. A separate database was chosen, because additional data like statistical information needs to be calculated and stored as well, which would bloat all miner?s built-in databases with information, if implemented in the miner. The fact that there are unified data structures and the possibility of having stored procedures in the database makes SQL a preferable database model, since the website always queries predefined statements. This also makes running the website possible without having a miner running in the backend. However this requires a program that copies data from a running Bazo mining node's database and stores it in the new SQL database. As mentioned above, the backend accesses this database by making queries for relevant data and sending the results to the frontend to be displayed to the user.

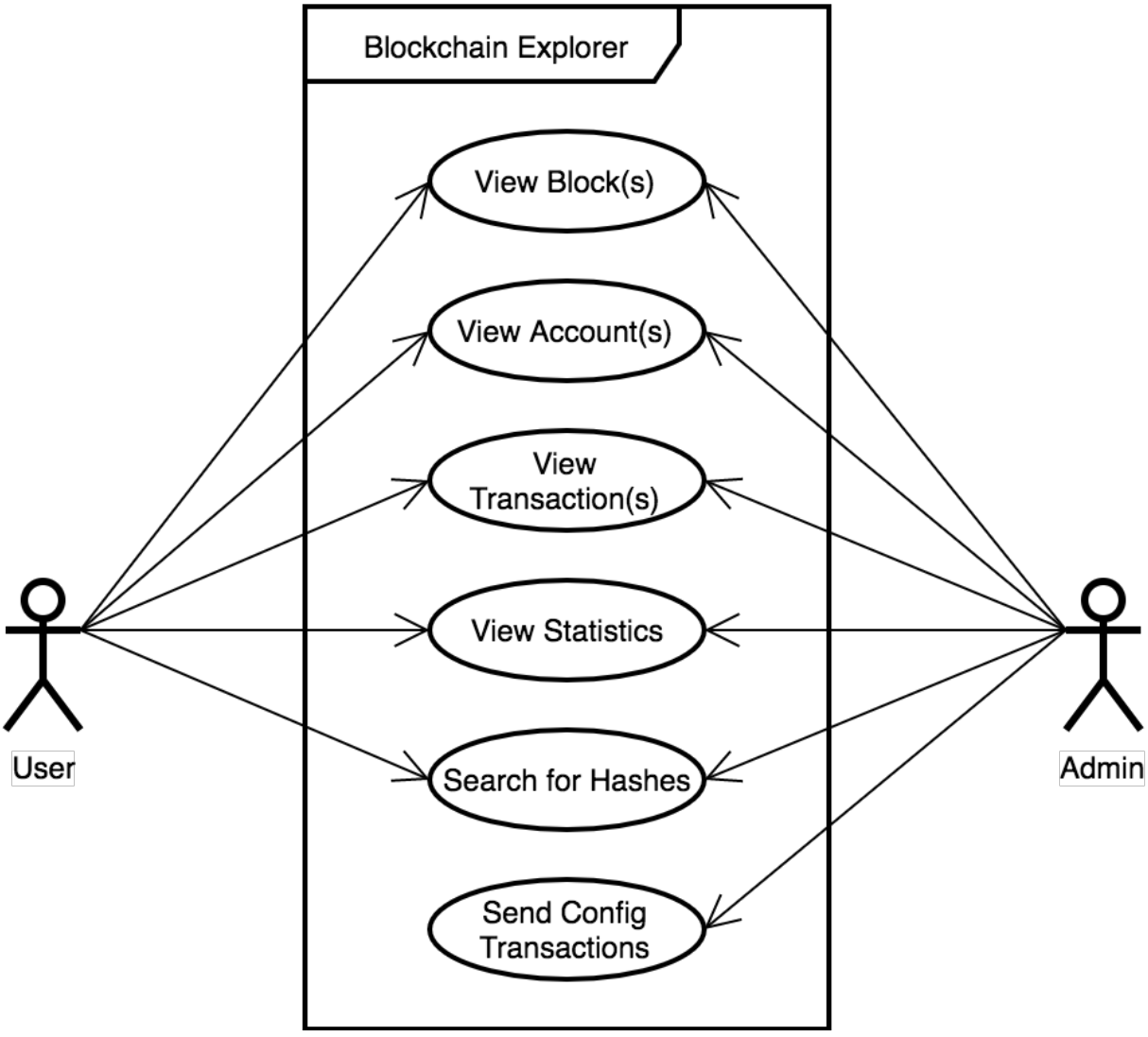

Figure 3.1: Use Cases of the Block Explorer

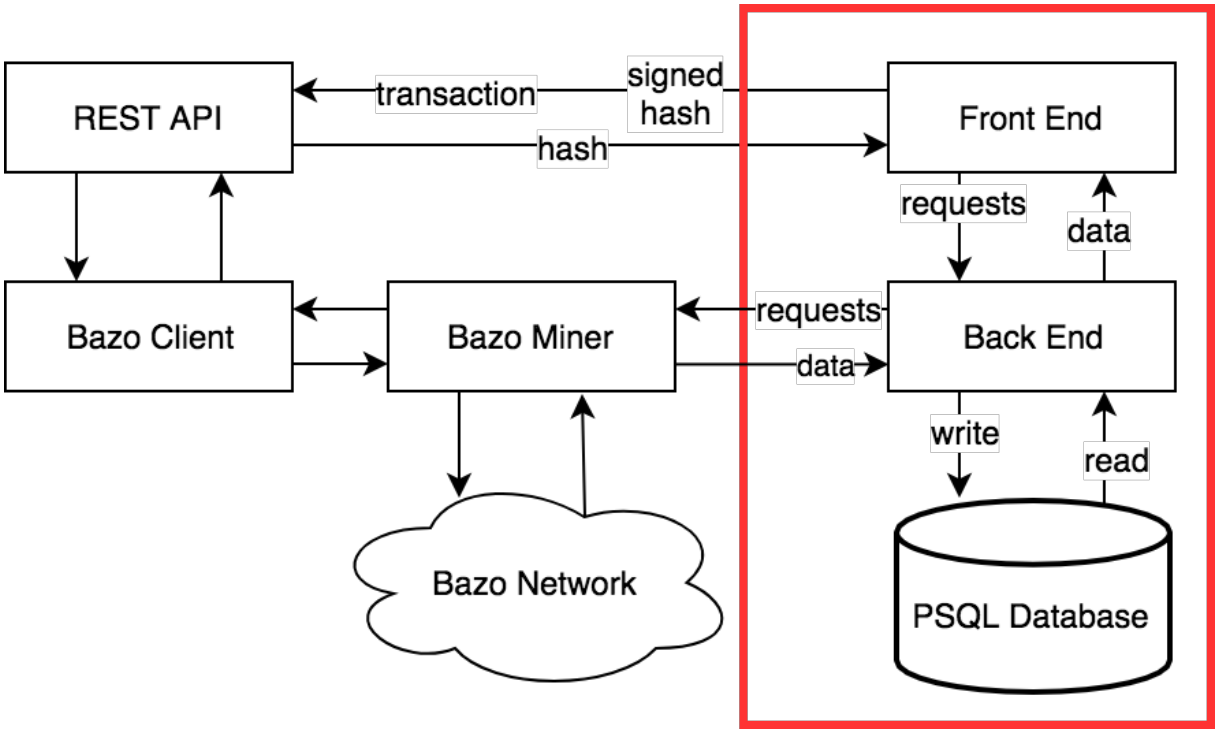

Figure 3.2: Structure of the Whole System

3.3 Structure of the Website

Chapter 4

Implementation

This chapter documents the implementation of the components listed in chapter 3; Frontend, Backend and the SQL database. An additional section concerning hosting of the application on the internet is also included.

4.1 Frontend

XXX

4.1.1 HTML Templates

To interact with the Golang-Backend of the application, Gohtml templates have been used to define the markup of the application. One one hand, this makes way for a modular view component by letting programmers define reusable HTML modules such as headers, footers or table-templates. Using templates can minimize code duplication, which in turn makes maintenance on the code an overall less risky task. On the other hand, Golang templates allow for some limited logic in the Gohtml files, which is needed for handling variables that get passed from the backend component. For example, if the passed variable is an array of integers and the goal is to display all variables in a list, golang can create a new <li> tag for every value in the array. The HTML code that gets passed to the end user after making a request contains no Golang code, since the backend renders the Gohtml template files and passed variables to a useable HTML file that can be displayed by a web browser. Except for some Bootstrap functionality mentioned in 4.1.2 and one case discussed in section 4.1.3, all rendered pages of the application are static HTML files, because for every action a user may make on the website, a request has to be made to the backend. The website aims to always display the most recent data, so storing information that may not be up to date on the client's machine in order to save bandwidth, is outweighed by the possibility of having more recent data.

Reusable Modules

lskjdflkj

4.1.2 UI Frameworks

4.1.3 Client-Side Logic

Chapter 5

Evaluation

Chapter 6

Summary and Conclusions

Bibliography

[1] Autoren: Titel, Verlag, `http://...`, Datum.

Abbreviations

AAA	Authentication, Authorization, and Accounting

Glossary

Authentication

Authorization Authorization is the decision whether an entity is allowed to perform a particular action or not, e.g. whether a user is allowed to attach to a network or not.

Accounting

List of Figures

List of Tables

Appendix A

Installation Guidelines

Appendix B

Contents of the CD