

# Blockchain - Diaries

Paolo Bettelini, Giacinto Di Santis, Gianni Grasso

## Contents

|          |                |          |
|----------|----------------|----------|
| <b>1</b> | <b>Diaries</b> | <b>2</b> |
|----------|----------------|----------|

# 1 Diaries

## 2022-01-27

- Created GitHub repository
- Added Readme
- Created Gantt
- Initial analysis
- Documentation
- Gradle init

Today we started to take a look at the blockchain technology. We started planning which pieces of software we will need to write in order to make everything work. We also initialized the project using Gradle.

## 2022-02-03

- Created node module
- Created seeder module
- Imported SQLite dependency
- Tested SQLite database
- Added LaTeX files to `.gitignore`
- Seeder packets
- Byte utils classes

Today we created the node and seeder module using gradle. We focused on defining the protocol for the seeder application (to allow for peer discovery). We wrote the code for the peer discovery packets. We also tested SQLite, creating and interacting with a test database.

## 2022-02-05

- Design change
- Written seeder service
- Tested connection to seeder

This design change is related to how a user interacts with the blockchain.

Our initial idea was to develop a *client* software alongside the *node* software. The *client* would connect to a single node as an entry to the blockchain to broadcast its transactions. The user would then be able to broadcast transactions to the blockchain using a web-based application.

This is a poor design choice since it has a number of problems.

- **Problem 1** Dual-functionality: Each node would need to be able to handle and distinguish both a connection from a node and a connection from a client.
- **Problem 2** WebAssembly: Both the *client* and *node* software would share the same protocol implementation, written in Java. Since the application is web-based, we would need to compile the *client* code to WebAssembly (WASM) to run on the browser.
- **Problem 2.1** Sockets: It would be challenging to allow for *socket connections* from a Java-based WASM.

- **Problem 3** Peer discovery: The client software can't store the addresses of other nodes since it is a web-based application. Even by caching nodes using *cookies*, clients will generally need to rely on *seeders* more than they should.

The solution is to completely remove the client. Only nodes can broadcast transactions throughout the network. The web application will send requests to a server, which is also a node. This server will route the user transactions through its node. No WebAssembly is needed. Connection to the seeder will be established (possibly) only the first time the web application server's node connects to the blockchain. Users can still host their own node and use them as an entry to the network, and multiple users can host their own web application for the same purpose.