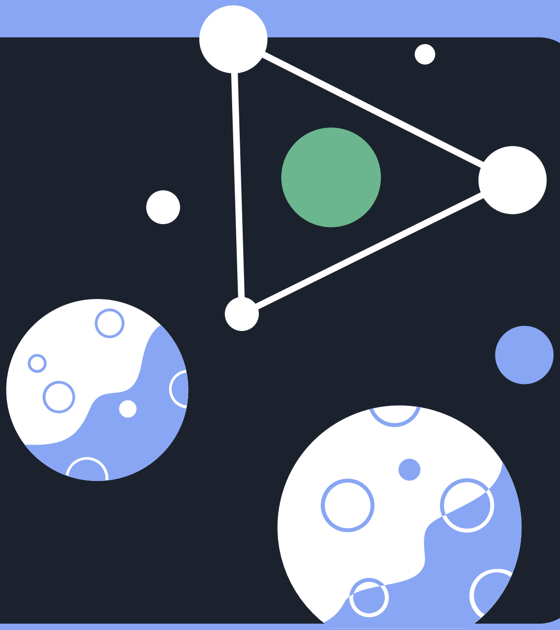
Peer to Peer Social Network



A decentralized social network

Bruno Gomes
up201906401

Fernando Rego
up201905951

José Silva
up201904775

Rui Moreira
up201906355

Table of contents

01 Project Description

02 Technologies

03 Design Choices

04 Joining the network

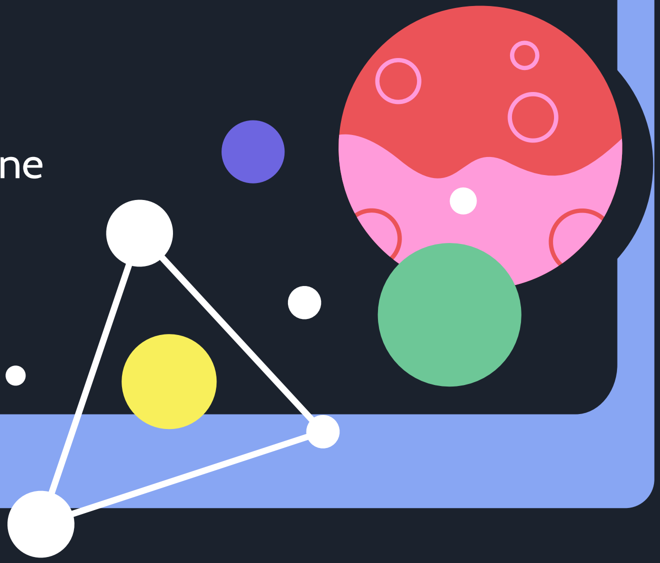
05 Future Improvements

01

Project Description

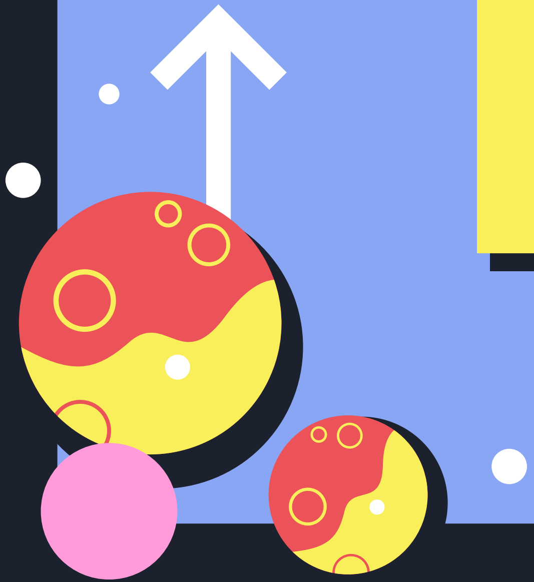
Project Description

- Social network similar to **Twitter**
- Users can **publish** text posts on their timeline
- **Following** users adds their posts to your timeline
- The **timelines** should always be up-to-date

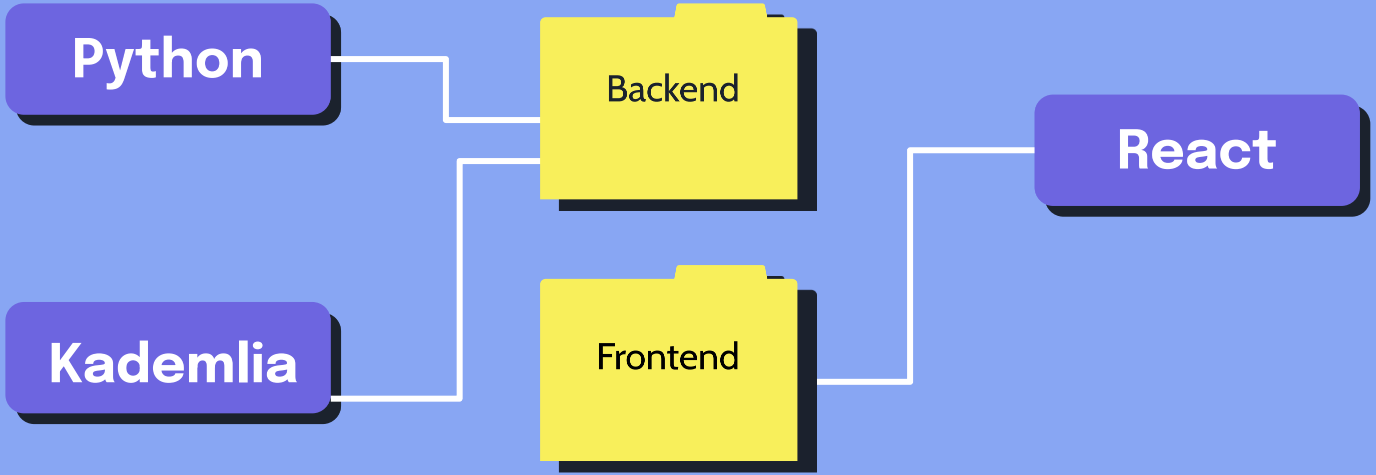


02

Technologies



Technologies



Python

Kademlia

Backend

Frontend

React

03

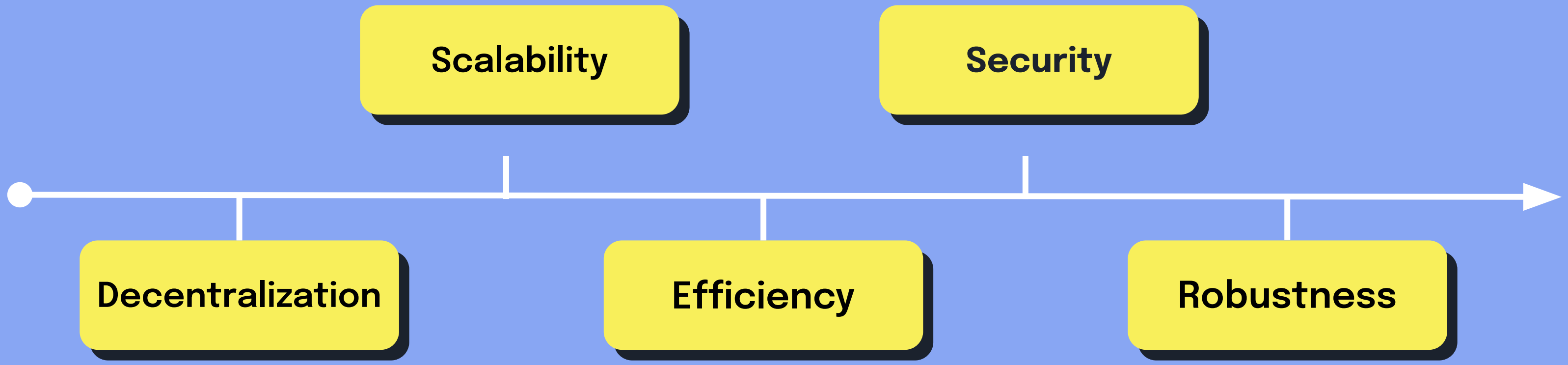
Design Choices

Design Choices

- Distributed Hash Table using Kademlia algorithm
- Be decentralized to the point where any user can run the program by just opening it (no server setup needed)
- Be resource efficient;
- Authenticity of every message is important to provide trust

DHT’s WHY?

Why use DHT



Scalability

Security

Decentralization

Efficiency

Robustness

Why we use it?

Decentralization

DHTs allow for decentralized networks. This makes the network more resistant to failure, as the loss of a single node does not compromise the entire network.

Scalability

DHTs can support a large number of nodes, allowing the network to grow and expand without encountering performance issues.

Robustness

DHTs are designed to be resilient to the failure of individual nodes, as the network can route messages around failed nodes using alternative paths.

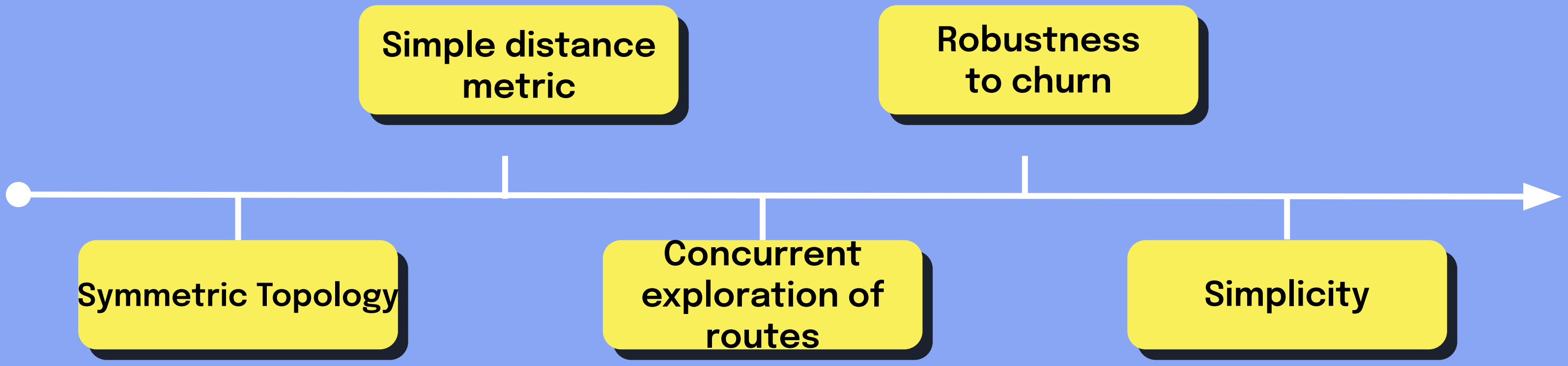
Efficiency

DHTs allow for efficient routing of messages, as each node only needs to maintain information about a small number of other nodes in the network.

Security

DHTs can be designed to provide secure communication between nodes, using encryption and other security measures.

Why use Kademlia Algorithm



Symmetric Topology

Simple distance metric

Concurrent exploration of routes

Robustness to churn

Simplicity

Why use Kademlia Algorithm

- **Symmetric topologies:** Kademlia uses a symmetric network topology, where each node has the same number of connections to other nodes. This makes the network more resistant to failure and allows for more efficient routing

- **Simple distance metric:** Kademlia uses a simple XOR distance metric, which allows for fast and efficient routing of messages.

- **Concurrent exploration of routes:** Kademlia allows for concurrent exploration of multiple routes to a destination, which can improve the speed and reliability of message delivery.

- **Robustness to churn:** Kademlia is designed to be robust to the churn of nodes, as nodes can be added or removed from the network without compromising its structure or performance.

- **Simplicity:** Kademlia is a simple algorithm, which makes it easier to implement and maintain in real-world systems.

04

Joining the Network

01

Bootstrap Node

At the start of the program, the peer connects to one well-known node.

02

Authentication

Users can register themselves if they do not have an account, or log into their accounts.

03

PKI

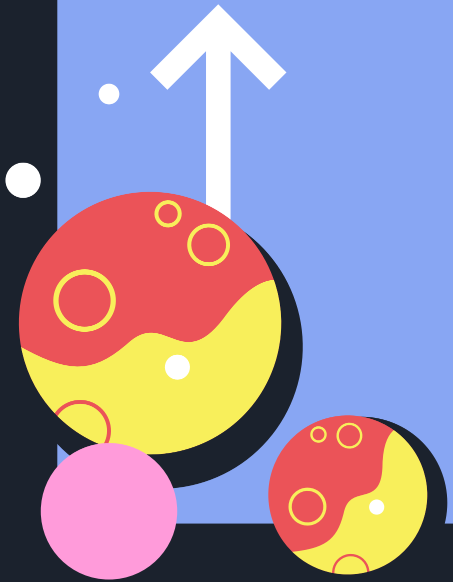
After authentication/registration, the user has access to the private key.

04

Connecting to DHT

Authentication and security

- **Bootstrap** nodes also serve as a **database**
- New users register themselves on a **node**, by providing a **username** (unique) and **password** that is encrypted using **bcrypt** algorithm
- Upon user authentication a private/public key pair is generated, the **private key** is stored in the user's browser cache, and the **public key** is stored in the **database**
- On the network, each post travels **authenticated** (signed) by the author's **private key**
- All messages containing posts, where the public key is unable to verify the authenticity of the message, are discarded
- This makes it so only registered users can participate in the network



05

Future Improvements

Network Time Protocol

- The **Network Time Protocol** (NTP) is a protocol used to **synchronize** the clocks of computers over a **network**
- **NTP** is typically used in a hierarchical structure, with **highly accurate** reference clocks providing time information to **less accurate** clocks
- It is also possible to use **NTP** in a **peer-to-peer** configuration, where each node in the network acts as both a time **server** and a **client**
- In a **peer-to-peer NTP** network, each **node** maintains a local clock that is synchronized with the reference time, and also acts as a **time server** for other **nodes**

Demo

Thanks!