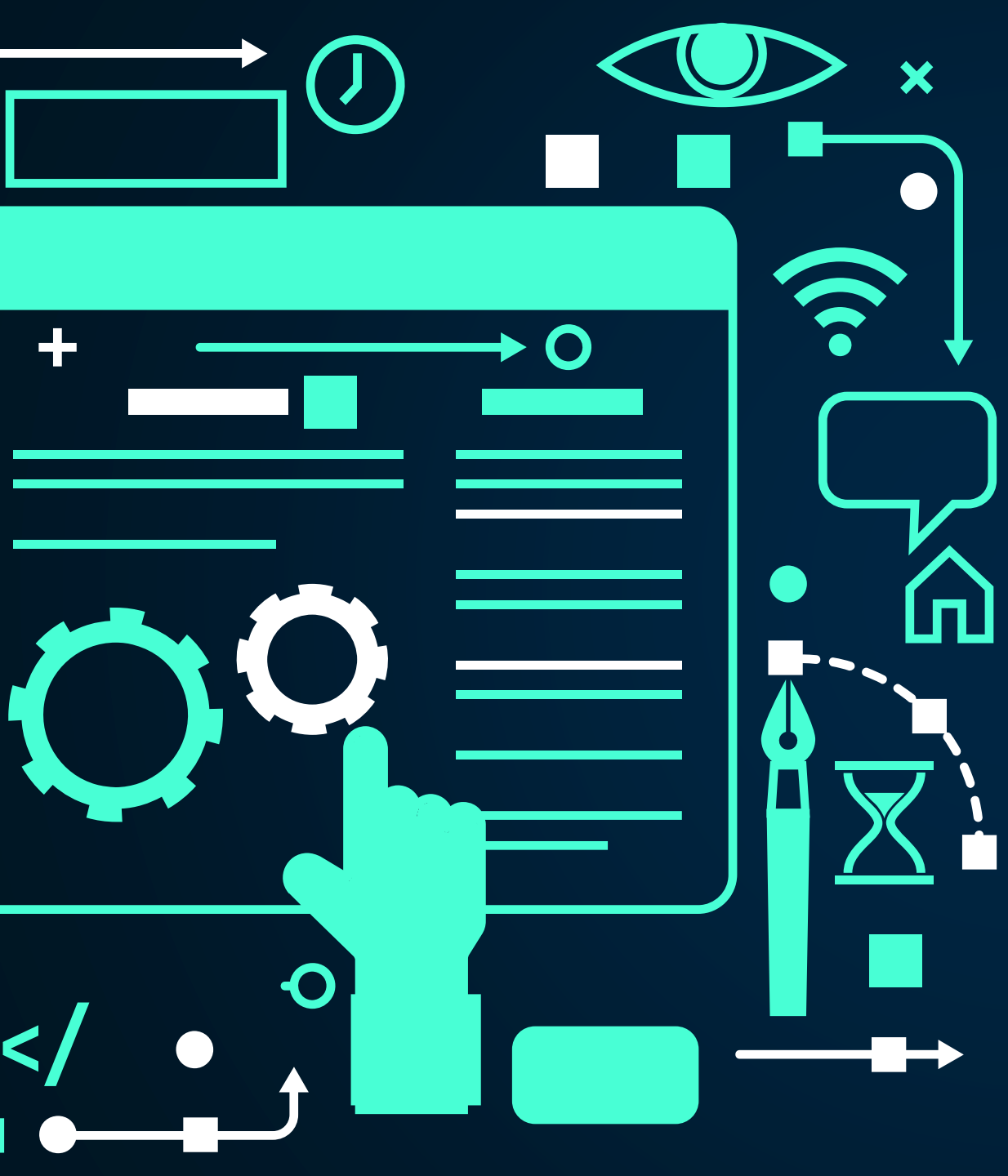


The Grid

A Decentralized Timeline

Large Scale Software Development

THE TEAM

Ana Rita Ramada
José Frederico Rodrigues
José Pedro Ferreira
Lucas Calvet

TABLE OF CONTENTS

Technologies

Python

Used for **implementing the protocol** (manages the DHT, handles signatures, transfers posts between users) and a simple API for the frontend

React

Used for **the frontend** to make it easy to interact with the system and exchange public keys through QR codes

Kademlia - DHT

“A distributed hash table for decentralized peer-to-peer computer networks.”

Kademlia is being used to keep **information about users** in a **decentralized manner**, allowing for the efficient storage and retrieval of user data without relying on a single central server.

Each user in the network is assigned a unique identifier, and their data is stored in the DHT using this **identifier as a key**. The data is then retrieved by other users by using the same identifier.

Public-key Signature System - Ed25519

The Ed25519 signature scheme is being used to sign posts with the user's private key and verify the signature of a post using the author's public key.

User is created and a private and public key are generated

User publishes a new post, that's **signed** with its private key

This user's followers verify the signature of the post using the public key provided by the user

Architecture - User

For each User it was defined:

- **private key,** which is used to sign their posts
- list of **subscriptions**
- list of **subscribers**

Users can create posts and they can also query the network for posts from their subscriptions.

DHT - Bootstrapping

A **bootstrapping** method id being used with out Kademlia DHT.

This is done by creating a User object with the necessary information and adding it to the network as the seed node.

This initial node will then be used to establish the network and **allow other nodes to join.**

Architecture - Receiver

The **Receiver** class, that subclasses Thread, was created with the objective of listening for incoming messages.

When a **message** is received, a handler determines the operation specified by the message and calls the appropriate method.

The four possible operations are:

- Subscribe
- Unsubscribe
- Request posts
- Send posts

These methods perform various tasks such as adding or removing subscribers and sending or receiving posts.

Subscribing

1. User A wants to subscribe User B.

2. First, A searches for B’s public key and returns an error if it does not exist. If it does exist, A sends a message to B letting them know they want to follow them.

3. If User B is online, B responds with all their posts (that they authored) and both users add each other to the correct entries (subscriptions and subscribers).

4. If B is offline, A will add themselves to B’s DHT table entry and request their posts from the first user that both subscribes B and is currently online.

Unsubscribing

1. User A wants to unsubscribe User B.
2. A will send a message to B letting them know they want to unfollow them.
3. If B is online, both users will remove each other from the correct entries.
4. If B is offline, A will remove themselves from B’s DHT table entry and remove B from their own table entry.

Request / Send Posts

To retrieve a user's posts, we need to specify the public keys of two users: the first is the user whose posts we want to retrieve, and the second is the user from whom we want to retrieve them.

We ask for the posts starting from a certain id, and will receive all the posts from that id onwards.

After receiving a request for posts, a user will look for posts from the target requested starting from the given id, and if they exist they will send them.

Frontend Implementation

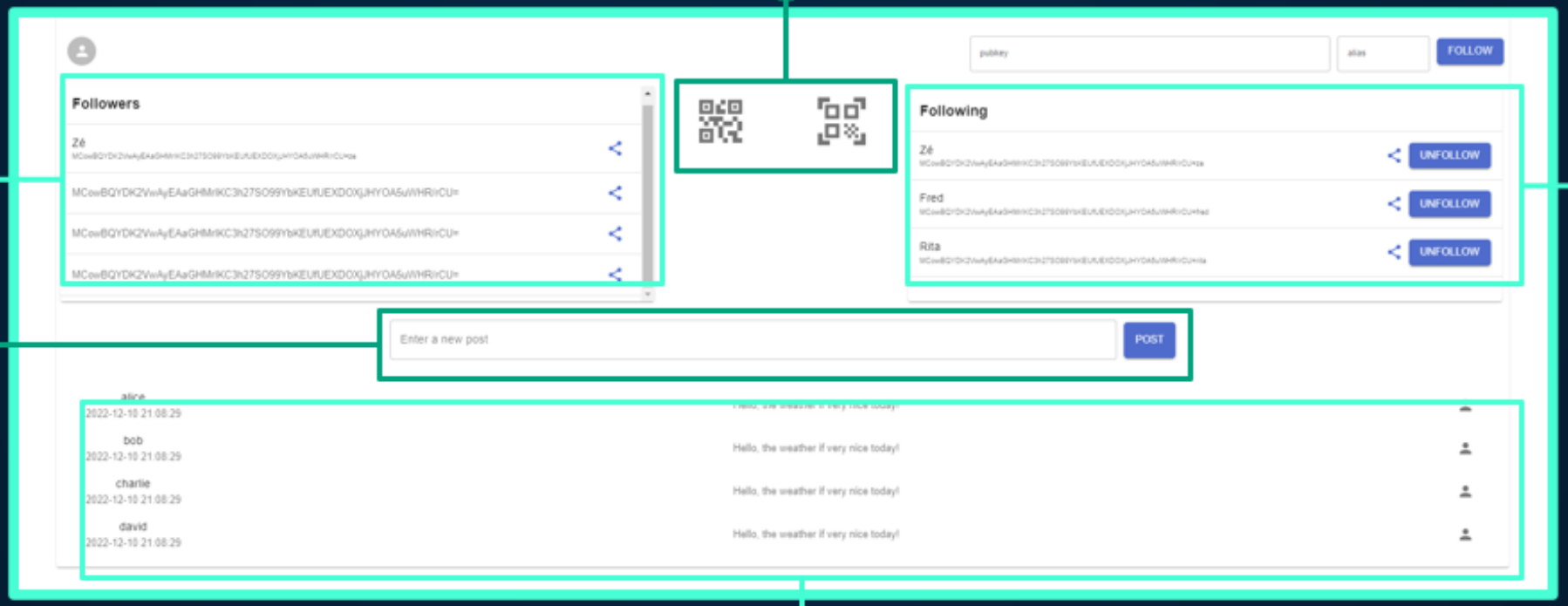
As it was said, our frontend was implemented using **React.**
For its implementation it was necessary to create an **API** with the necessary endpoints.

QR codes to share my key and get other user's public key

All users that are subscribed to me

All users I'm subscribed to

Input field to write new posts



Timeline with all the posts from users that I'm subscribed to

Conclusions

The use of a decentralized system brings multiple advantages by increasing the security and privacy for users.
Additionally, because decentralized systems are often built on distributed networks, they can be more resilient and less vulnerable to attacks or disruptions.

Overall, it's a complicated task to engineer and implement such a system

Future Work

- Explore authentication mechanisms
- Posts with other media content, not just text

Demo