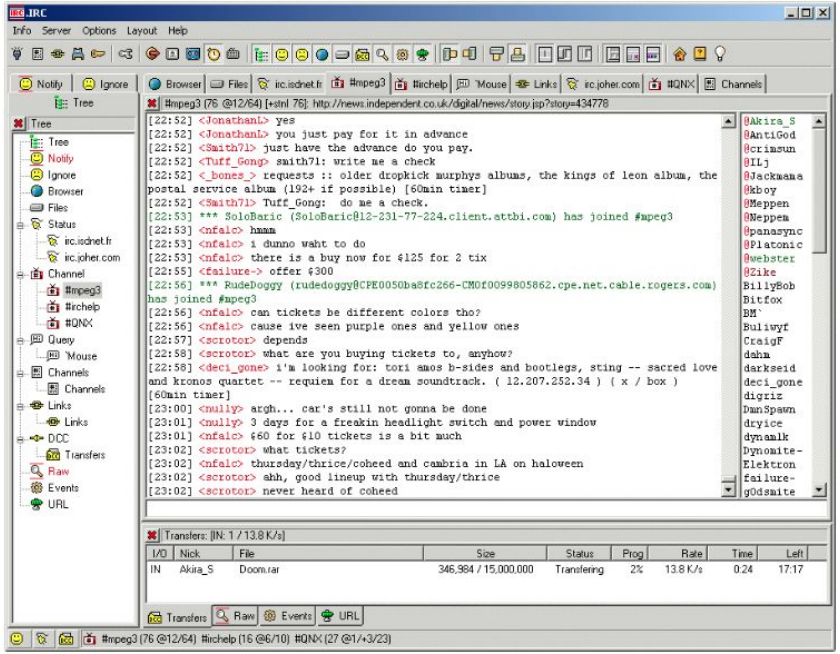
OEDC

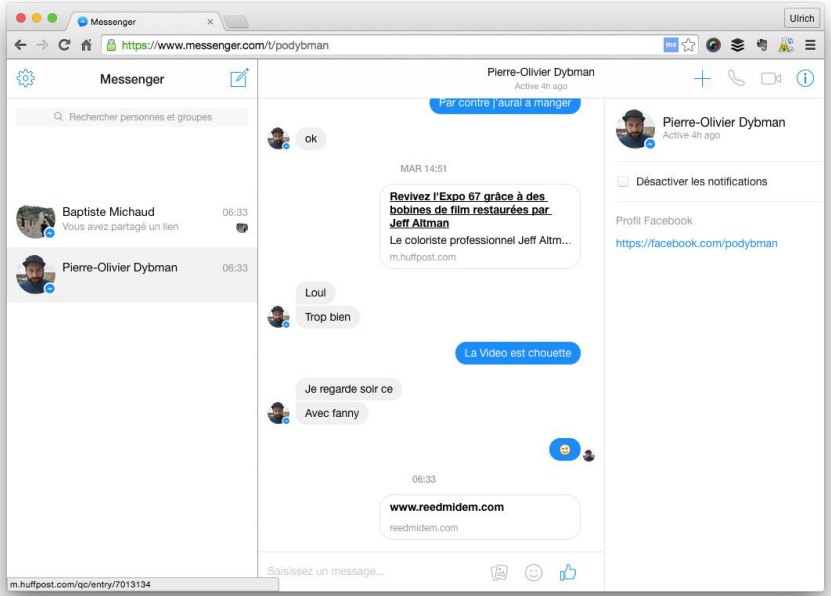
Online Ephemeral Decentralized Chat

Summary

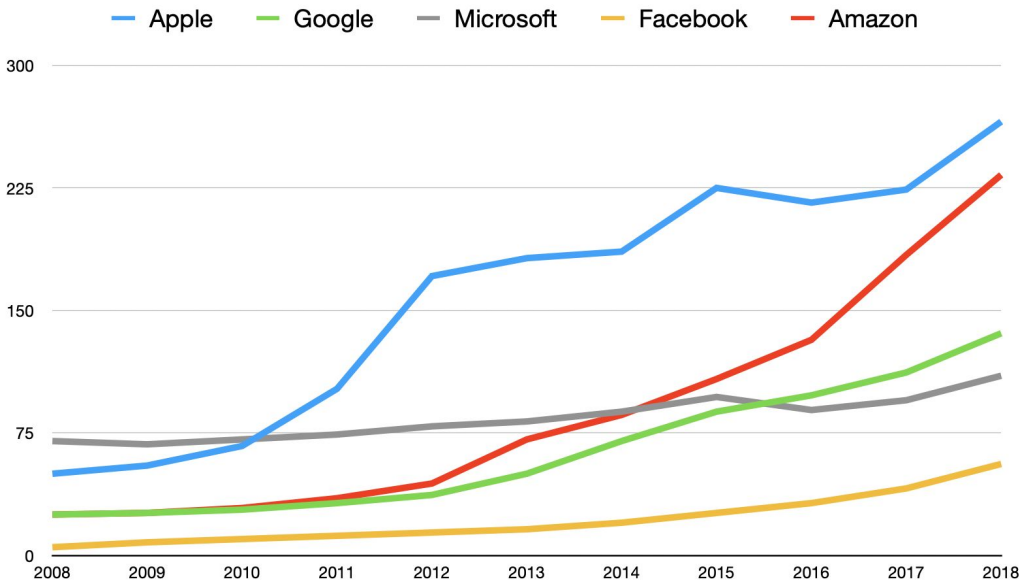
Introduction

The evolution of online chat



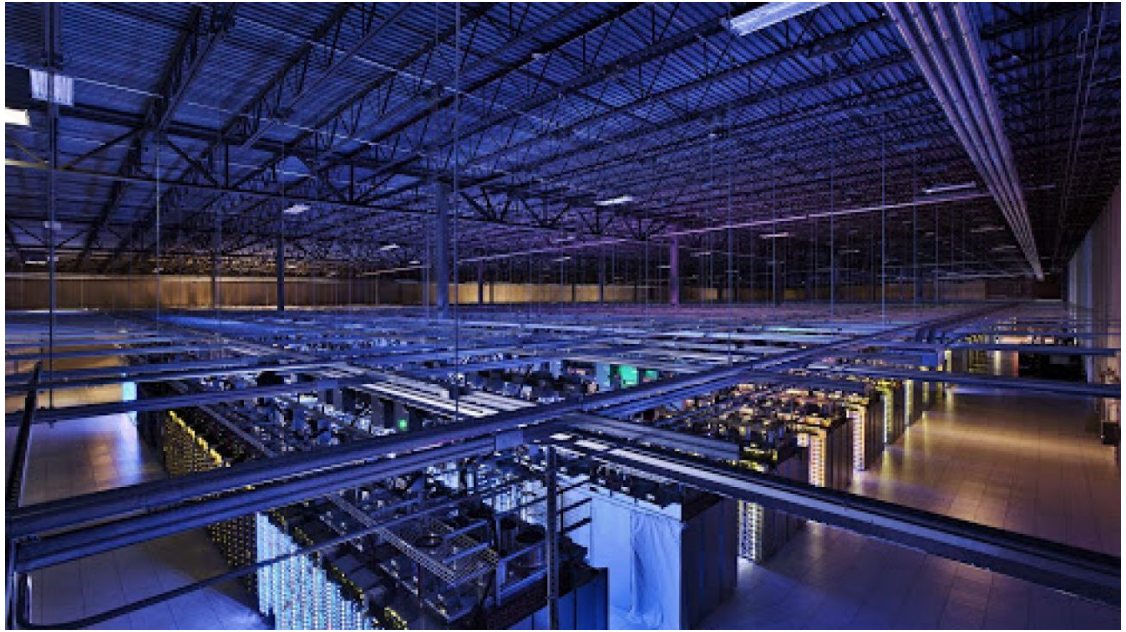


The rise of GAFAM



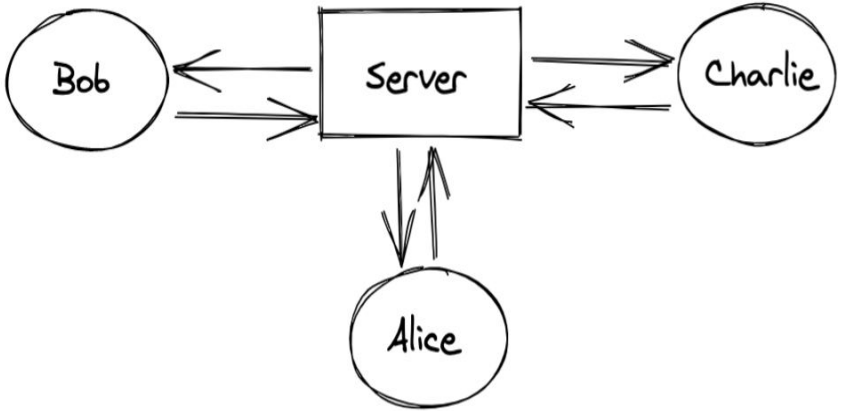
Market capitalization in billion of Dollars

Problems

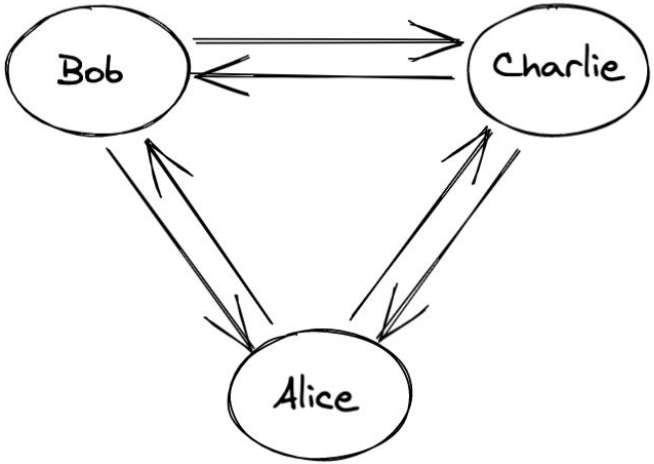



Donald J. Trump
@realDonaldTrump
51 Following 88,7M Followers
Account suspended
Twitter suspends accounts that violate the Twitter Rules.

Back to decentralized models



Centralized model



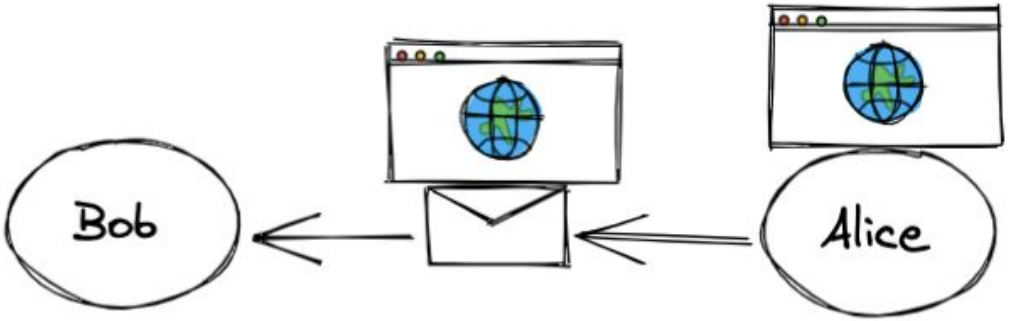
Decentralized model

Pear-chat.com

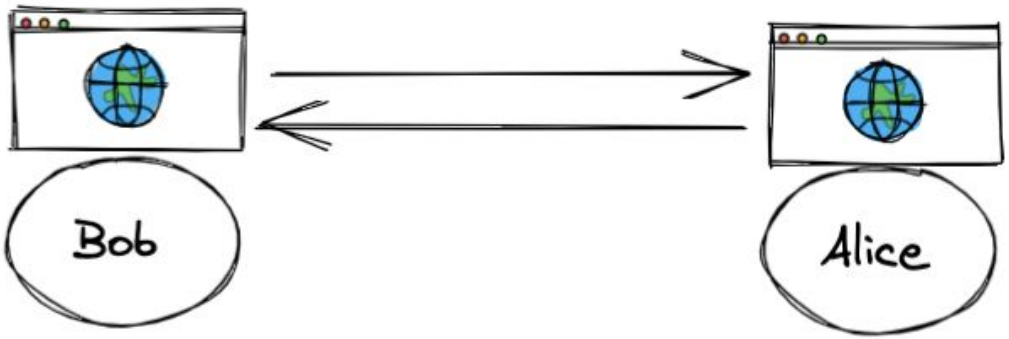
Fully decentralized?

→ How to distribute the chat application?

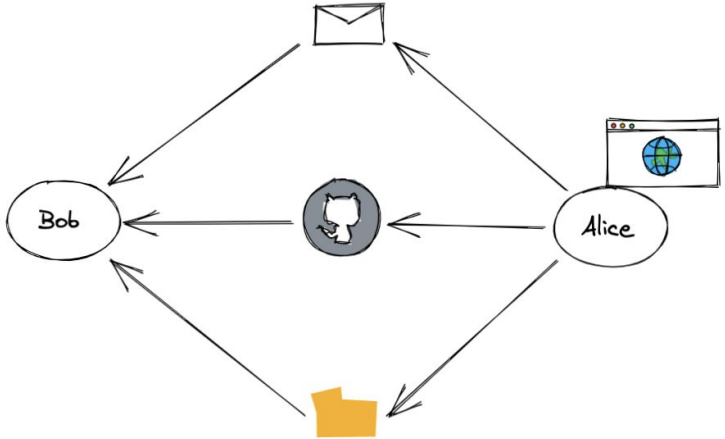
Fully decentralized?



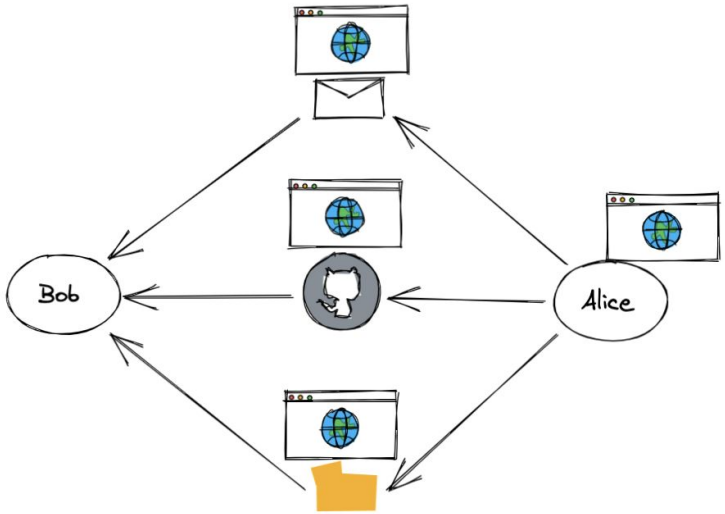
Fully decentralized?



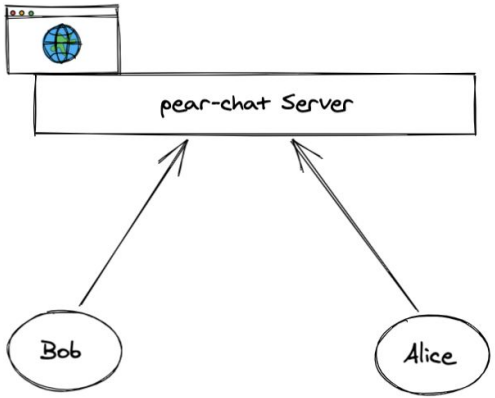
Fully decentralized?



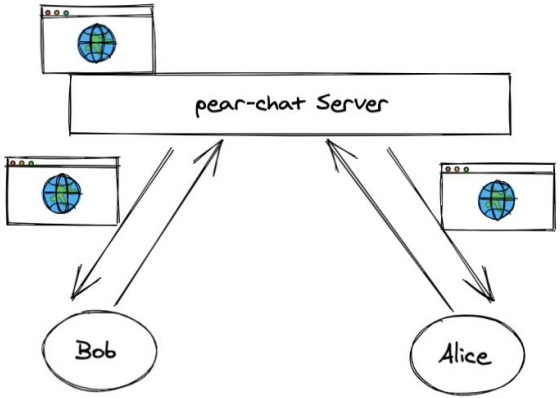
Fully decentralized?



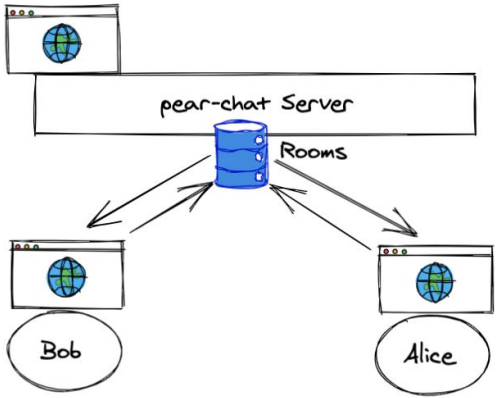
Fully decentralized?



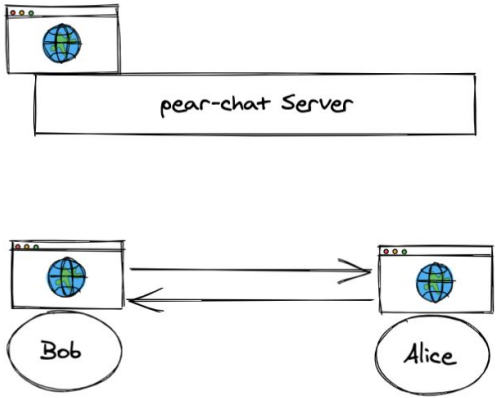
Fully decentralized?



Fully decentralized?



Fully decentralized?

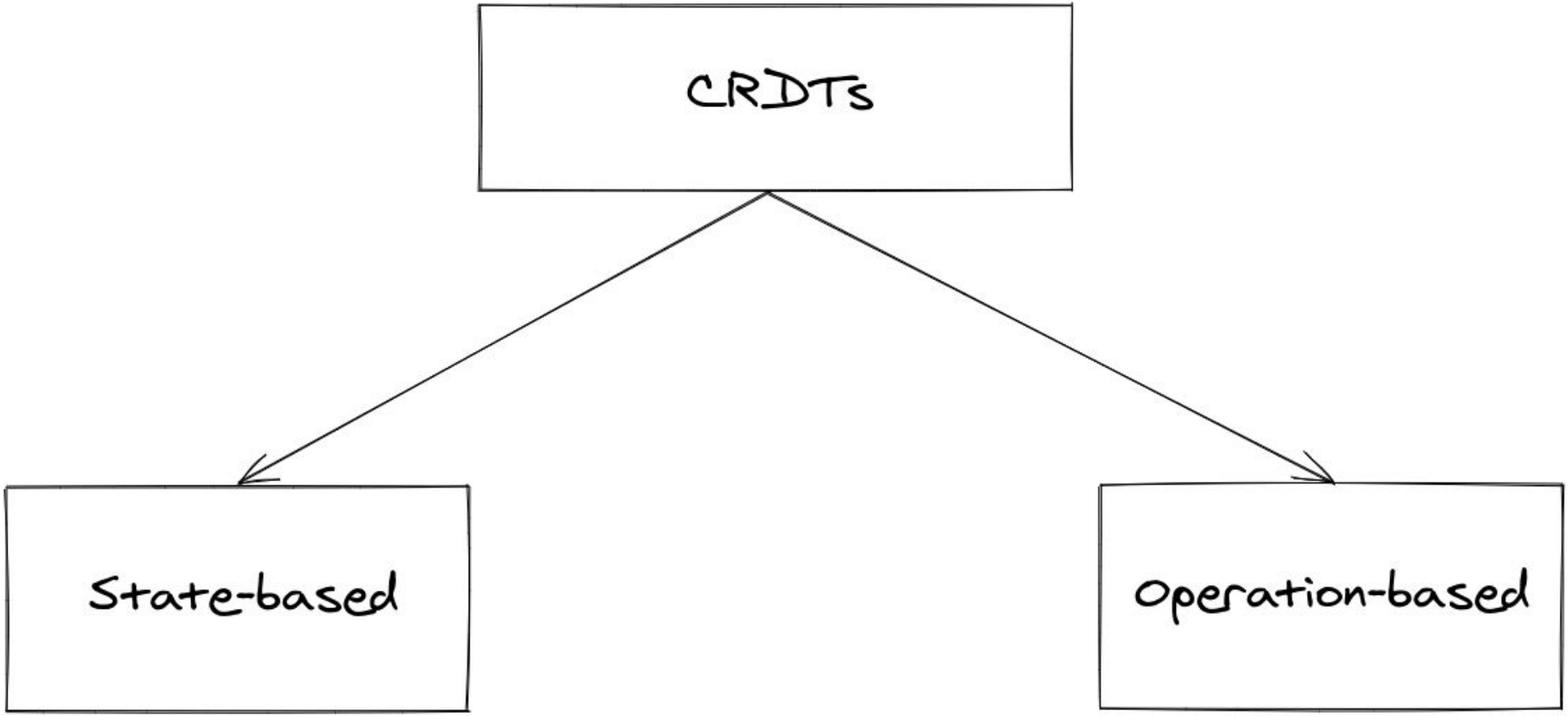


CRDTs

What Are CRDTs?

Example of a PN-counter

Time	Instance A	Instance B
t1	INCRBY key1 10	INCRBY key1 50
t2	-- Sync --	
t3	GET key1 => 60	GET key1 => 60
t4	DECRBY key1 60	INCRBY key1 60
t5	-- Sync --	
t6	GET key1 => 60	GET key1 => 60


CRDTs
State-based
Operation-based

State-based objects

$(S, s0, q, u, m)$

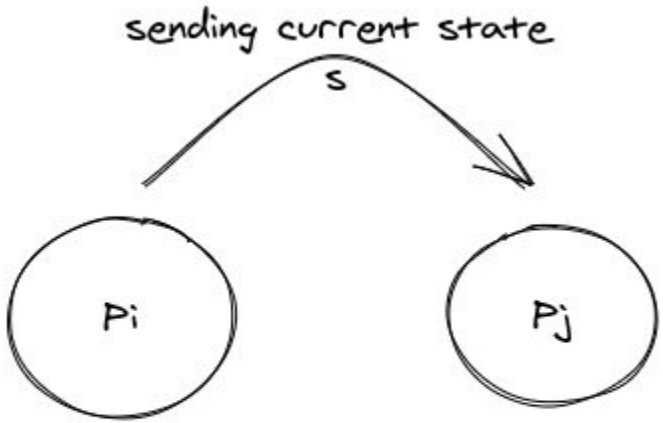
the replica at process pi has state si ∈ S, called its payload;

the initial state

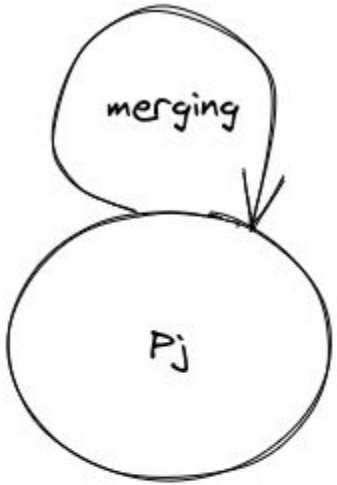
query method to read the state of the object

update method to modify the object

method to merge the state from a remote replica



1



2

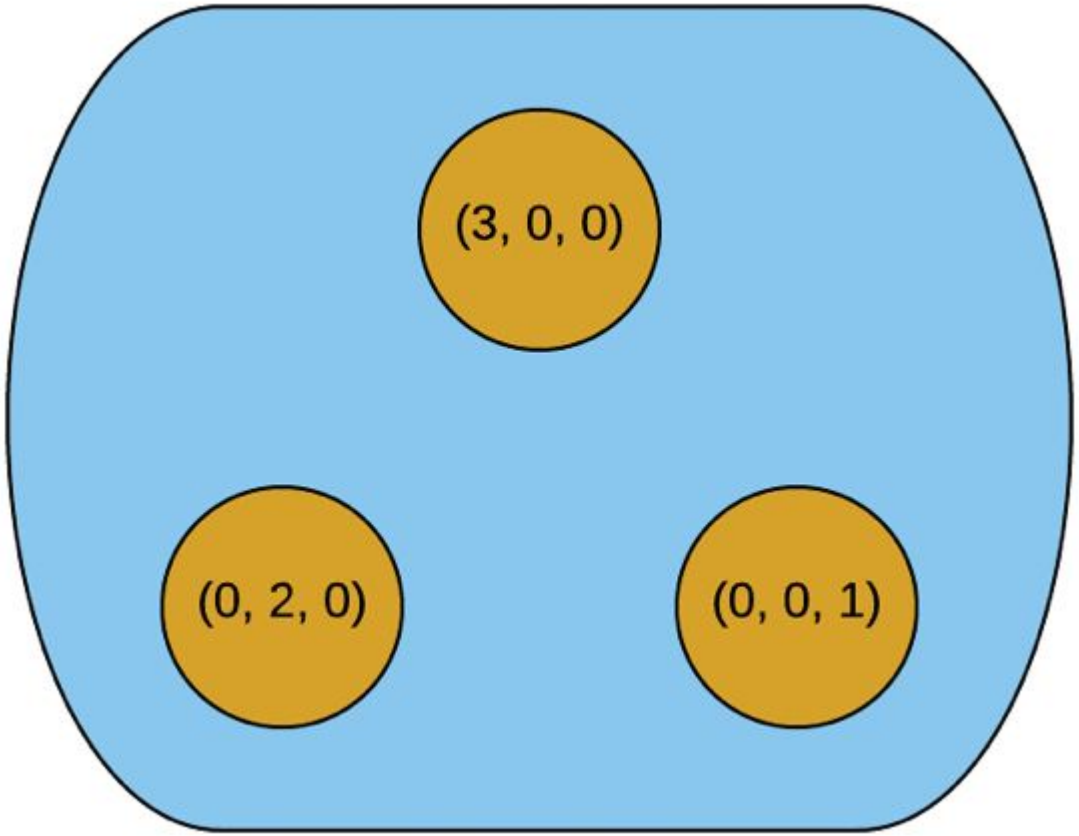
We assume that an enabled method executes as soon as it is invoked.

Final state > order

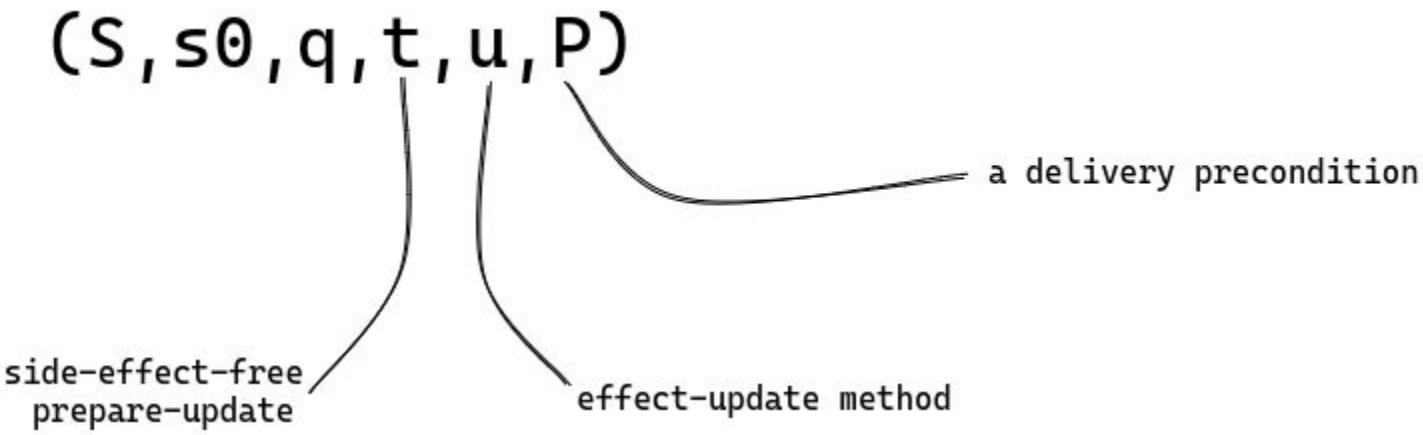
X: (3, 0, 0)

Y: (0, 2, 0)

Z: (0, 0, 1)



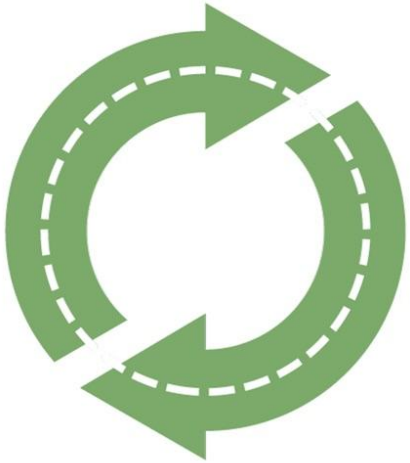
Operation-based objects



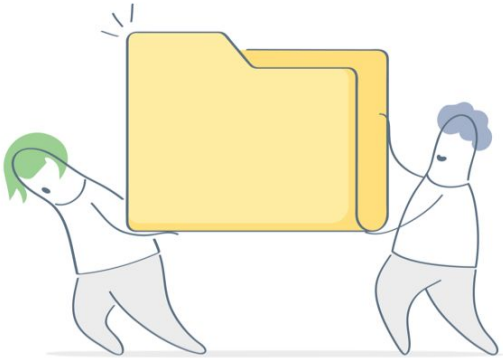
The effect-update method executes at all replicas (said downstream).

All concurrent operations commute = convergence of op-based CRDTs

Commutative Replicated Data Type (CmRDT)



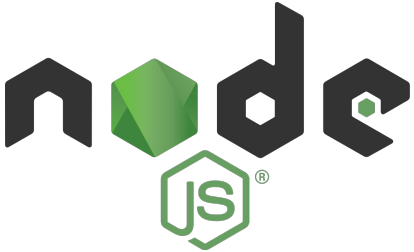
Why CRDTs?

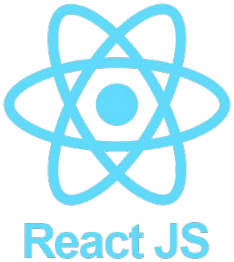


Global project architecture

Different technologies

- Project is online-oriented
  - Dedicated language -> JavaScript
  - Dedicated libraries
- Two sides: back-end and front-end





Comparison between PHP and JS

PHP

- Server-oriented
- Dynamic updating through requests to a server
- Requires a powerful server
- Capable of almost anything
- Code is hidden

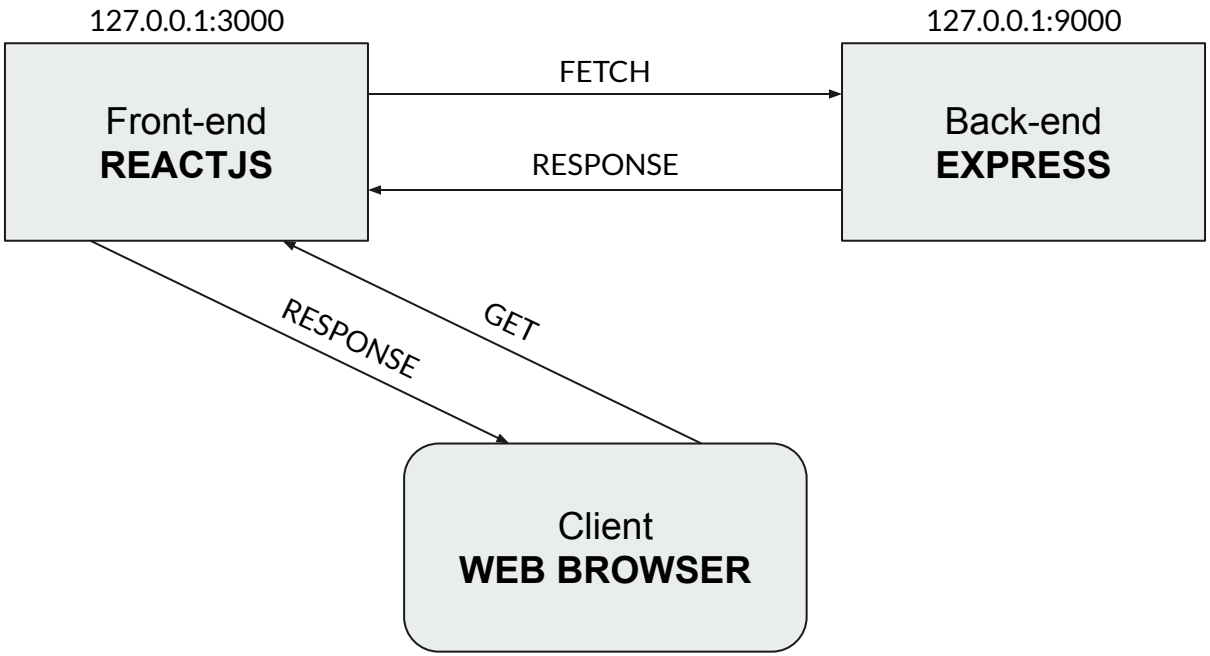
Not suitable for decentralized architecture

JavaScript

- Client-oriented
- Dynamic updating on client side without server interactions
- No server required
- Extremely versatile
- Code is visible to everyone

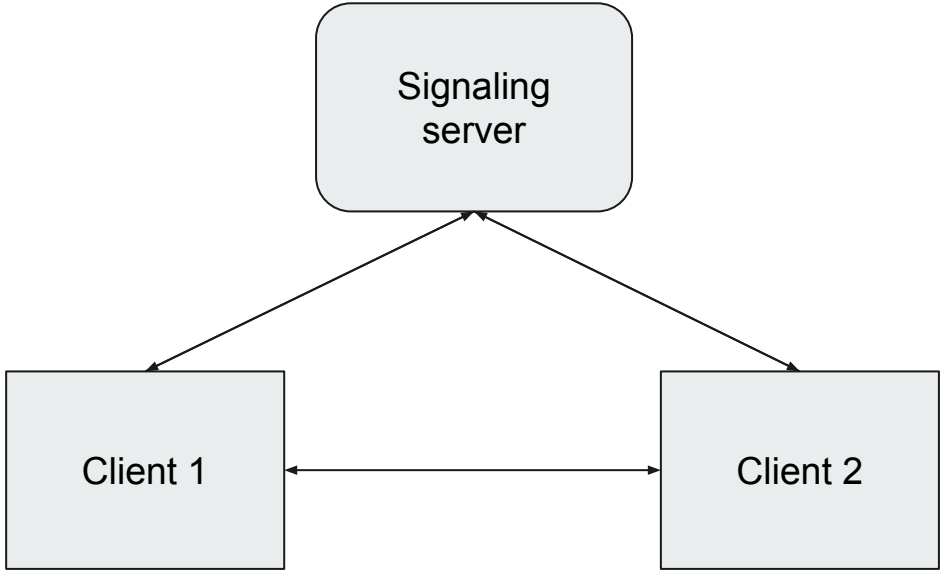
Highly suitable for decentralized architecture

Application architecture



Main issue with WebRTC

- Open source protocol developed in 2011
- Highly suitable for decentralized chats
- Not truly decentralized
- Peer-to-peer communications
- Some issues on some network



Back-end functionalities

Back-end presentation

- Working with two important modules
  - webrtc-swarm: applying WebRTC in NodeJS
  - signalhub: signaling server for WebRTC
- Express is used as an HTTP server
- MongoDB is used to store chat rooms information

Express JS

mongoDB

Express

- Express is a fast, unopinionated, minimalist web framework for NodeJS
- Creates a HTTP server
  - Express routes make an “access point” to back-end functions
  - Public repository to store JS scripts to inject in HTML

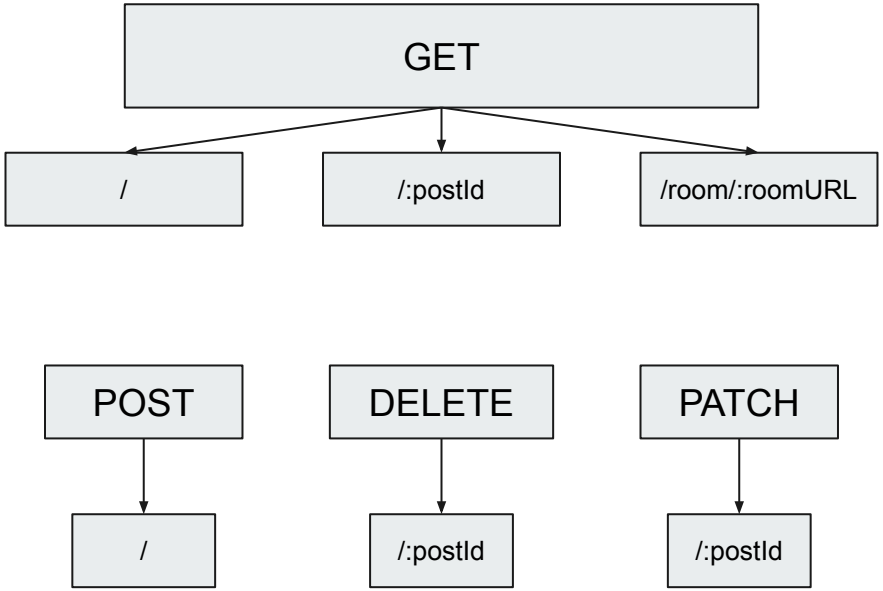
Express JS

MongoDB

- MongoDB (NoSQL) => document-oriented
- RESTful API with Express
- Creating methods (GET, POST...)
- Creating a predefined document model
- Mongoose for integration in NodeJS

How it works

- Three GET methods
  - Get document list
  - Get a document with its ID
  - Get a document with its URL
- A single POST method
  - Push document in database
- A single DELETE method
  - Remove a document with its ID
- A single PATCH method
  - Updating a document with its ID



How it works

- Defining a document model
- Creating a router for Express
- Methods declared in the router
- Defining the URL to route data

Mongoose makes MongoDB's integration easier

```
const BrowserSchema = mongoose.Schema({
    title: {
        type: String,
        required: true
    },
    privacy: {
        type: Number,
        required: true
    },
    visibility: {
        type: Number,
        required: true
    },
    maxusers: {
        type: Number,
        required: true
    },
    connusers: {
        type: Number,
        required: true
    },
    url: {
        type: String,
        required: true
    },
    userList: {
        type: [],
        required: true
    },
    password: String
})
```

Routes usage

```
// Fetch the list of available rooms
fetch('http://localhost:9000/browser')
    .then((response) => {
        return response.json()  // Convert the response object to JSON data
    })
    .then((res) => {            // Get the result of JSON response data
        this.setState({ rooms: res })
    })
```

- FETCH is a function allowing data recovery through a single URL
- Data conversion from String to JSON
  - Easier to process

- Defining response body in function options

```
const requestOptions = {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({
        title: this.state.name,
        visibility: 0,
        maxusers: 8,
        connusers: 1,
        url: md5(this.state.name)
    })
}
```

Response example

localhost:9000/browser

localhost:9000/browser

JSON Raw Data Headers

Save Copy Pretty Print


```
[{"_id":"60754681e4e3ce19b68558d3","title":"Enseirb","visibility":0,"maxusers":8,"connusers":2,"url":"81f0f0a0208eabce0ae5284b23c7478f","__v":0,"password":""},
{"_id":"60758d05e4e3ce19b68558d4","title":"Eirb","visibility":1,"maxusers":4,"connusers":3,"url":"f508dd886c130b94d8d16abbe10d0ea5","password":"enseirb","__v":0},
{"_id":"60758d1be4e3ce19b68558d5","title":"NewRoom","visibility":0,"maxusers":6,"connusers":4,"url":"4fe617d1d3ea2a8c96ed6eccb12243dc","__v":0,"password":"blopblop"},
{"_id":"60758d37e4e3ce19b68558d6","title":"ChatRoom","visibility":1,"maxusers":8,"connusers":2,"url":"e44b71ad408feebba591b8512fb5841f","__v":0,"password":""}]
```


localhost:9000/browser/607

localhost:9000/browser/60758d37e4e3ce19b68558d6

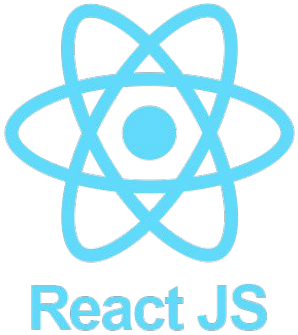
JSON Raw Data Headers

Save Copy Pretty Print

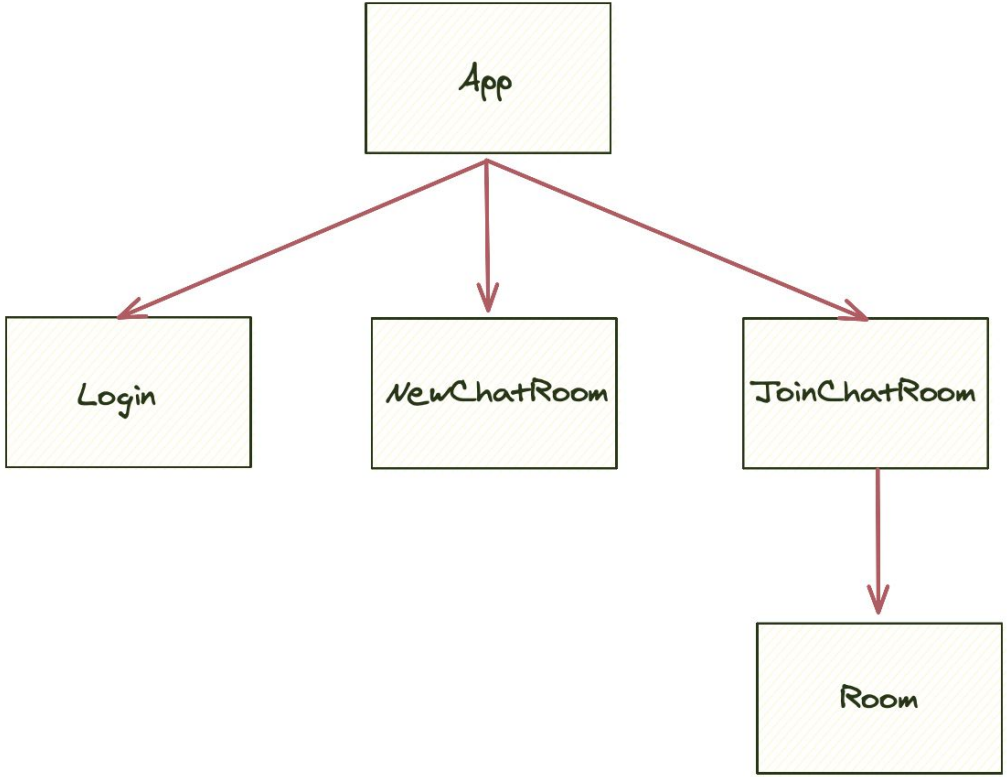

```
{"_id":"60758d37e4e3ce19b68558d6","title":"ChatRoom","visibility":1,"maxusers":8,"connusers":2,"url":"e44b71ad408feebba591b8512fb5841f","__v":0,"password":""}
```

Front-end functionalities

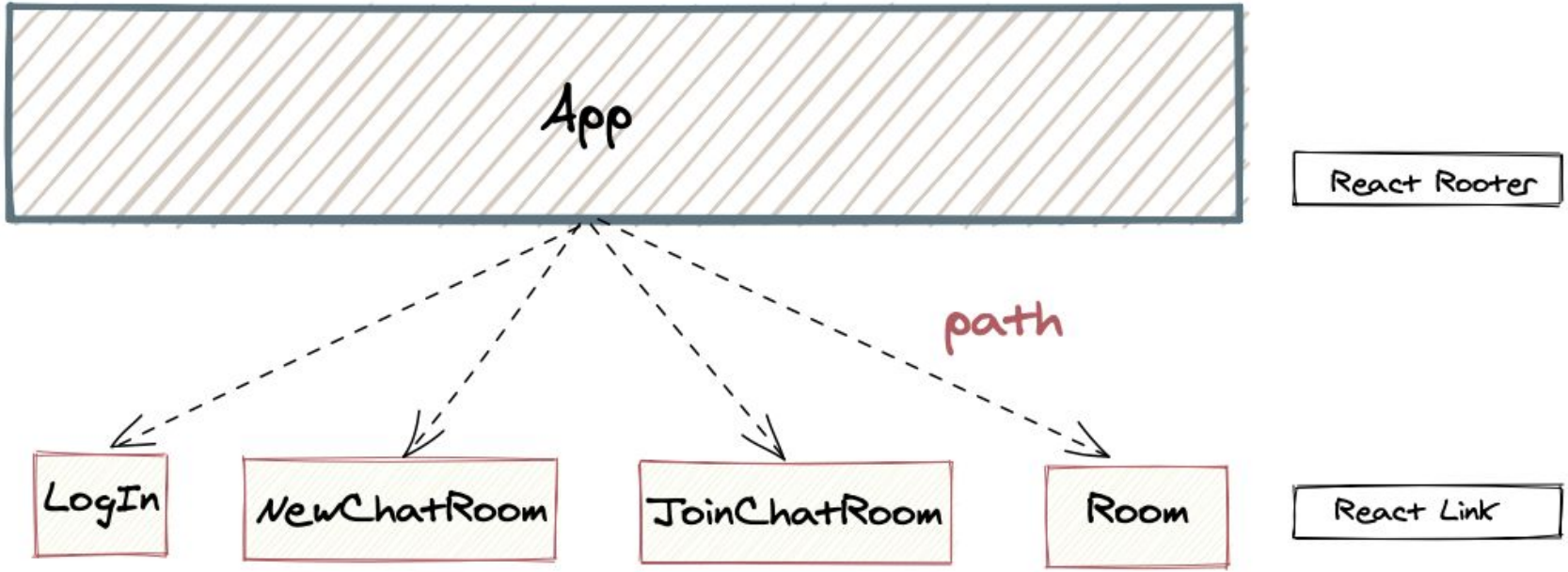
Front-end Language and framework



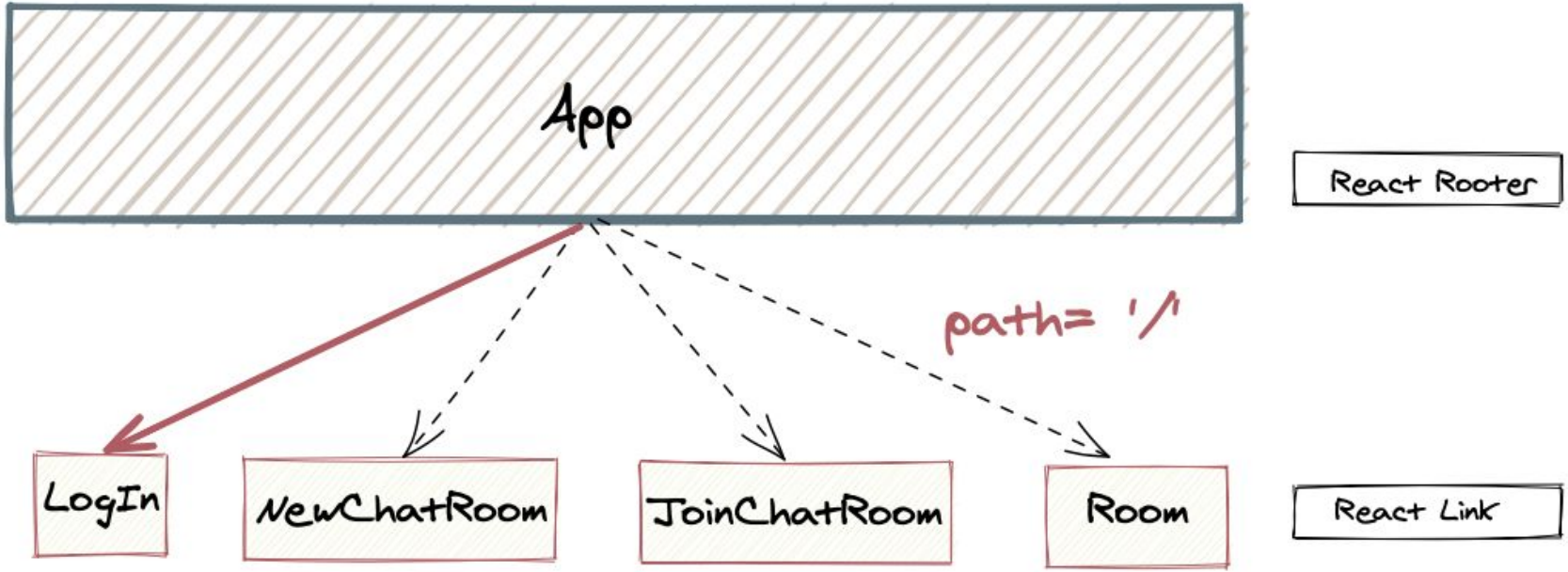
Front-end components



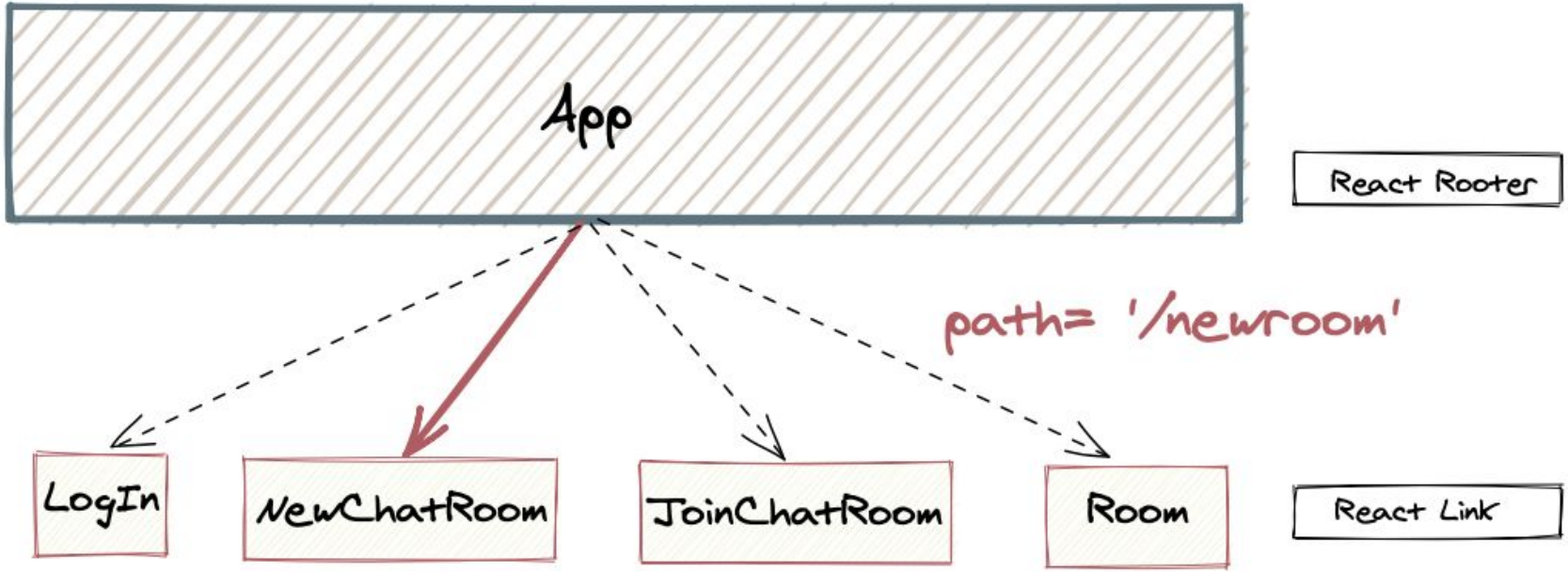
Components routing

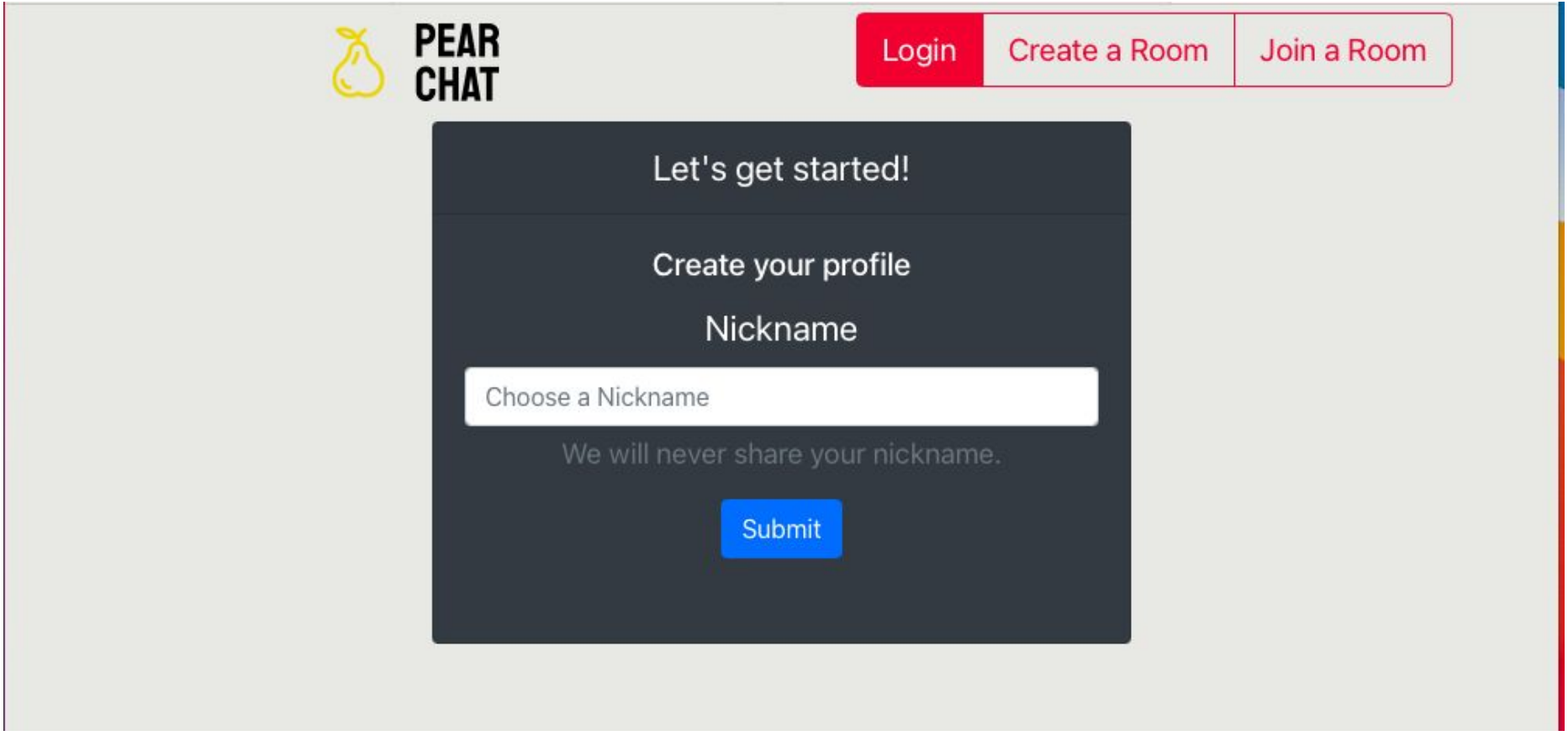


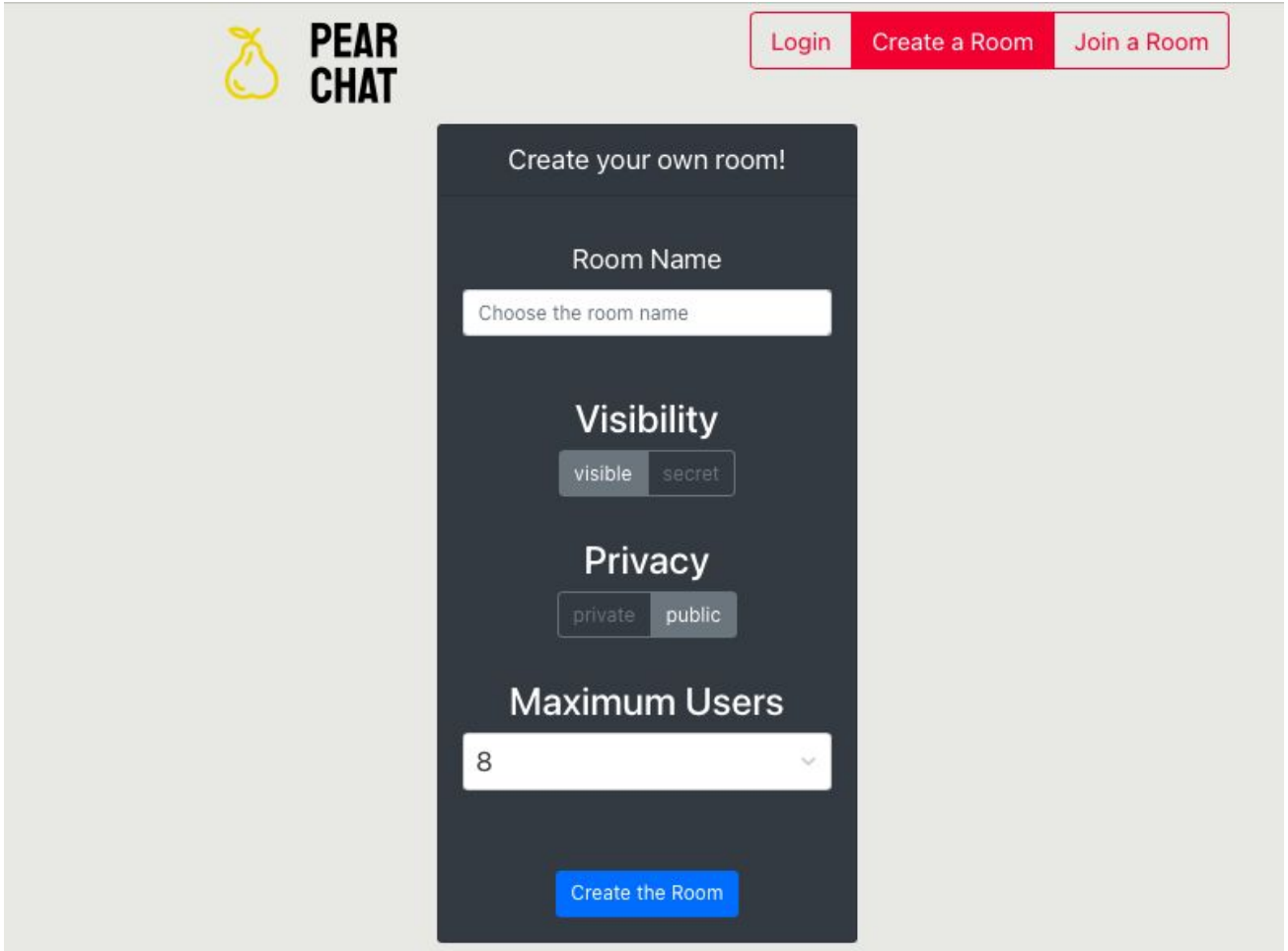
Components routing



Components routing

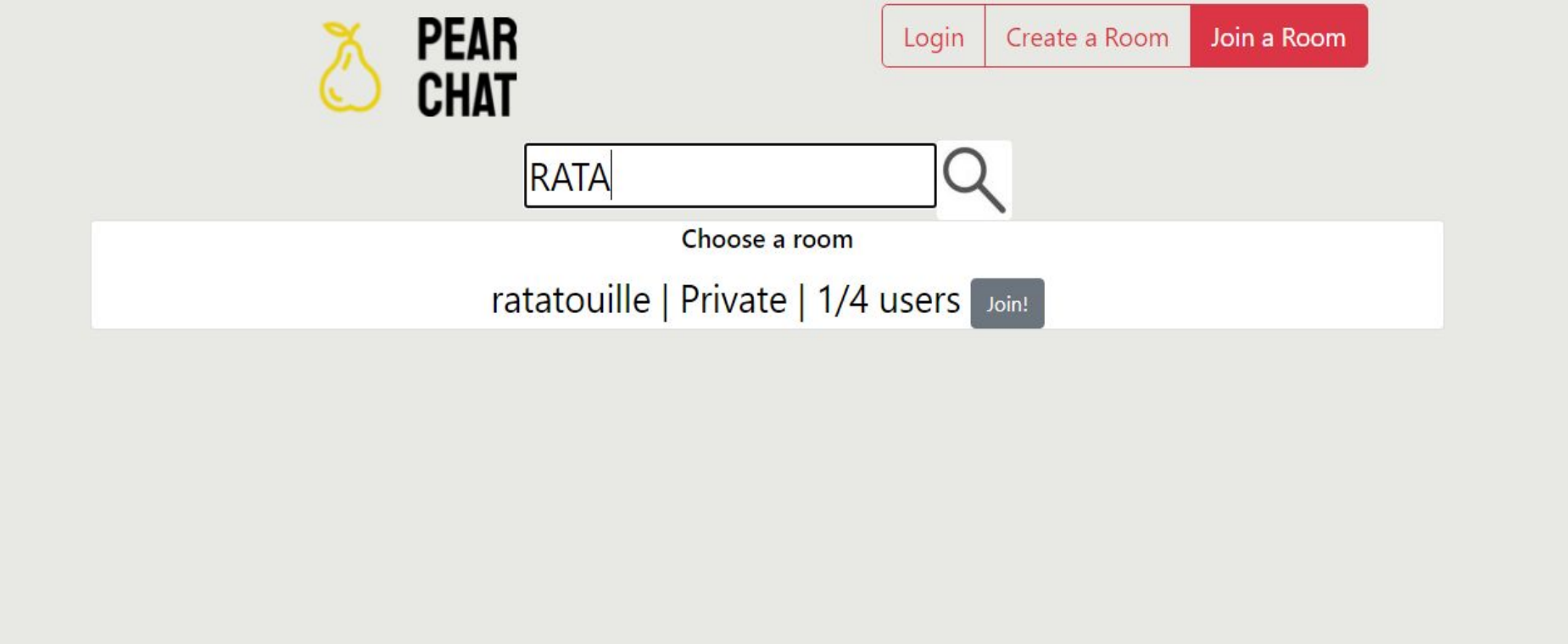



PEAR
CHAT
Login
Create a Room
Join a Room
Let's get started!
Create your profile
Nickname
Choose a Nickname
We will never share your nickname.
Submit

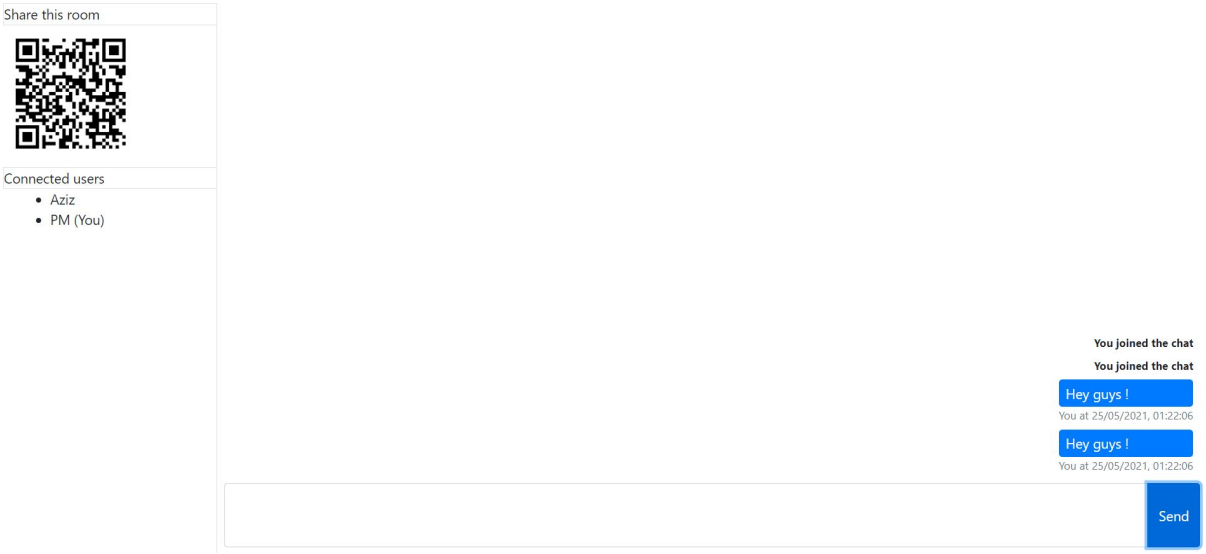

PEAR
CHAT
Login
Create a Room
Join a Room
Create your own room!
Room Name
Choose the room name
Visibility
visible
secret
Privacy
private
public
Maximum Users
8
Create the Room

Join a room


PEAR
CHAT
Login
Create a Room
Join a Room
Search a room...
Choose a room
Chat | Public | 4/8 users
Join!
ratatouille | Private | 1/4 users
Join!


PEAR
CHAT
Login
Create a Room
Join a Room
RATA
Choose a room
ratatouille | Private | 1/4 users
Join!

Chat with your friends



Project Management

- Agile Project Management Method : Client Involvement - User story Based - Incremental Repeated Process.

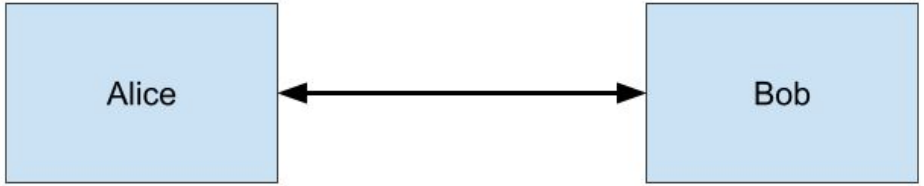
The Project Group has been divided into three main programming teams :

- Back-end Team : Achraf - Hamza - Guillaume - Romain
- Front-end Team : Anas - Pierre Malo - Timothée
- Merge Team : Guillaume - Romain

User Story Based Agile Method

- Client Requirements are translated into User Stories.
- User Stories are divided into tasks.
- Tasks are scheduled according to priority and organized chronologically via Gantt Diagram.

User Story 1 : Connexion entre deux utilisateurs



Hypothèse : les adresses IP et les ports sont connus par chaque utilisateur.

Scénario :

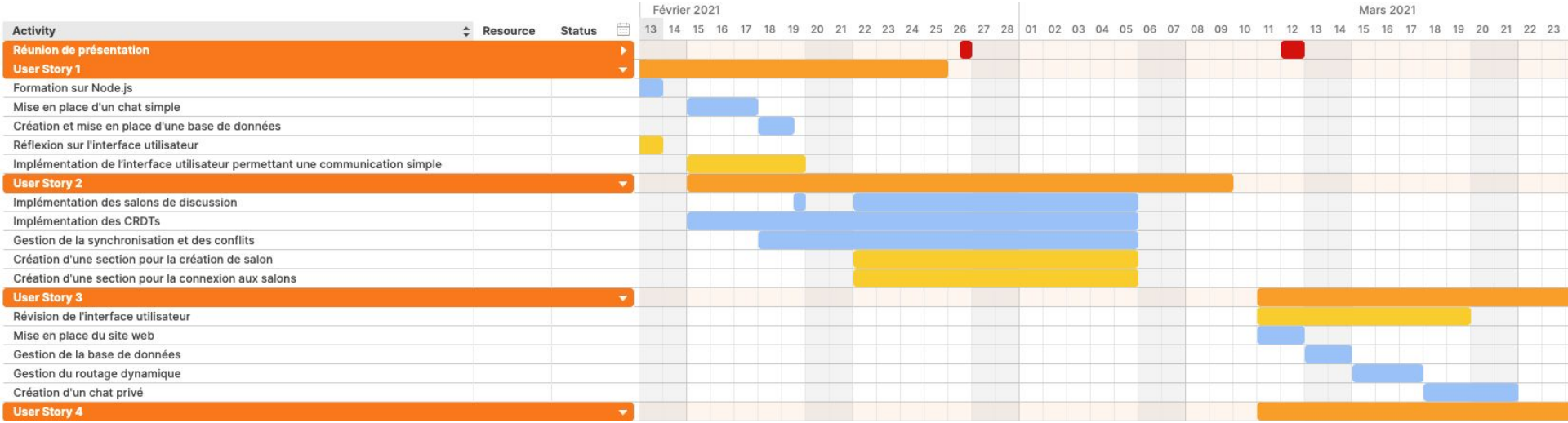
1. Alice se connecte à Bob
2. Bob accepte la connexion
3. Alice envoie un message à Bob
4. Bob répond au message

Solutions techniques :

- Création d'un fichier ou d'une base de données contenant les identifiants des utilisateurs
- Système de socket d'écoute
- Implémentation de l'interface utilisateur permettant une communication simple

GANTT Diagram

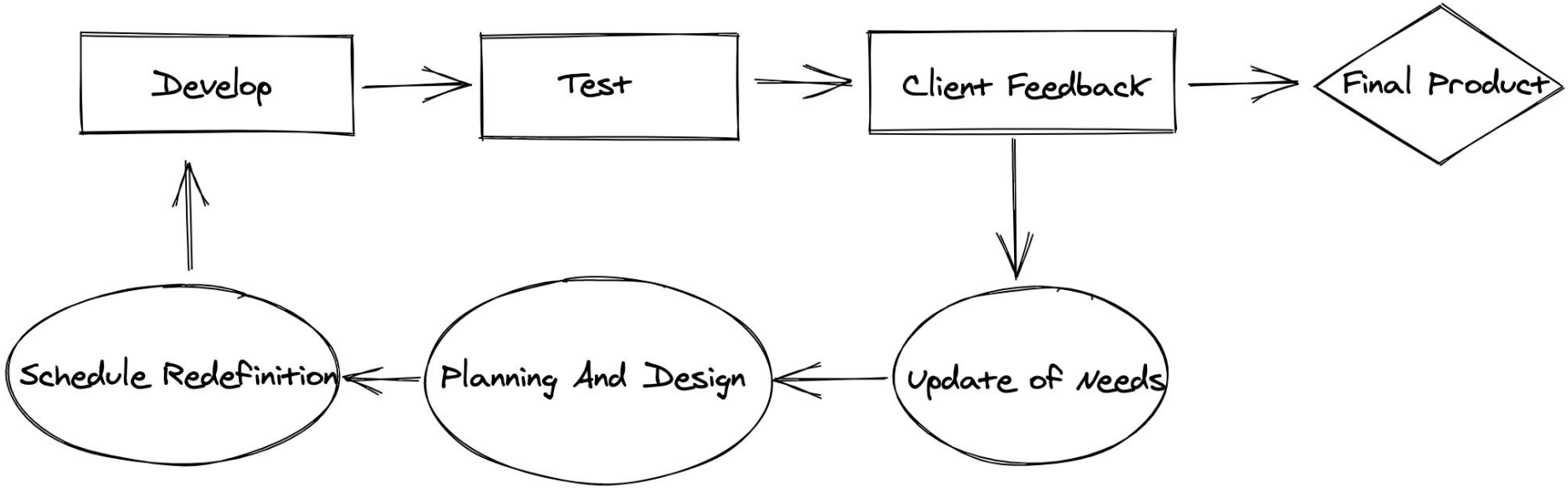
- Project Scheduling is illustrated in a GANTT Diagram :



Communication With The Client

- Programmed Visual conferences after every major advancement.
- Feedback is used to update the project’s timeline and task management.

Management Framework

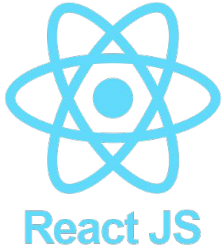


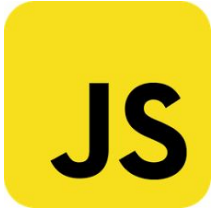
Conclusion

Let’s test our app !

www.pear-chat.com

What did this project teach us ?





Difficulty encountered

-The technologies that fit the best with our project

-Having a complete decentralized application

Other positive points of the project

As a group :

-Group dynamic

-Project management

-Team spirit

-Communication

Individually :

-Curiosity

-Autonomy

-Creativity


THANK YOU
FOR YOUR ATTENTION