deti universidade de aveiro
departamento de eletrónica, telecomunicações e informática

CrowdWire

<!-- MASSIVE ONLINE MEETINGS -->

<>

OUTSIDE AT HOME

</>

LICENCIATURA EM ENGENHARIA INFORMÁTICA

PROJETO DE INFORMÁTICA

GRUPO 09

CONTEXT AND STATE OF THE ART

Gather.town

Gather is the main inspiration for our project.

Discord

Discord has introduced a recent feature called Stages, similar to conferences.

DogeHouse

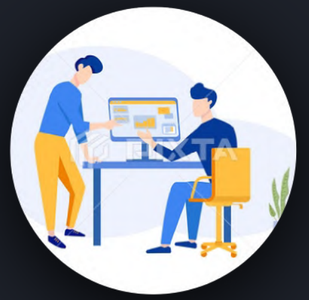
DogeHouse has the purpose to host large voice meetings in a very scalable manner.

REQUIREMENTS GATHERING

- WebRTC P2P vs MCU vs SFU
- Need of Horizontal Scaling of the Media Service: Use of a Container Orchestration Tool (Kubernetes)
- Investigation of Gather.town Functionalities.
- Brainstorming which ideas we could realistically implement and how.
- Research current technologies for each Service (Browser Game Engine, STUN/TURN Server for WebRTC, ...)
- Ideas Discussion with Advisor.

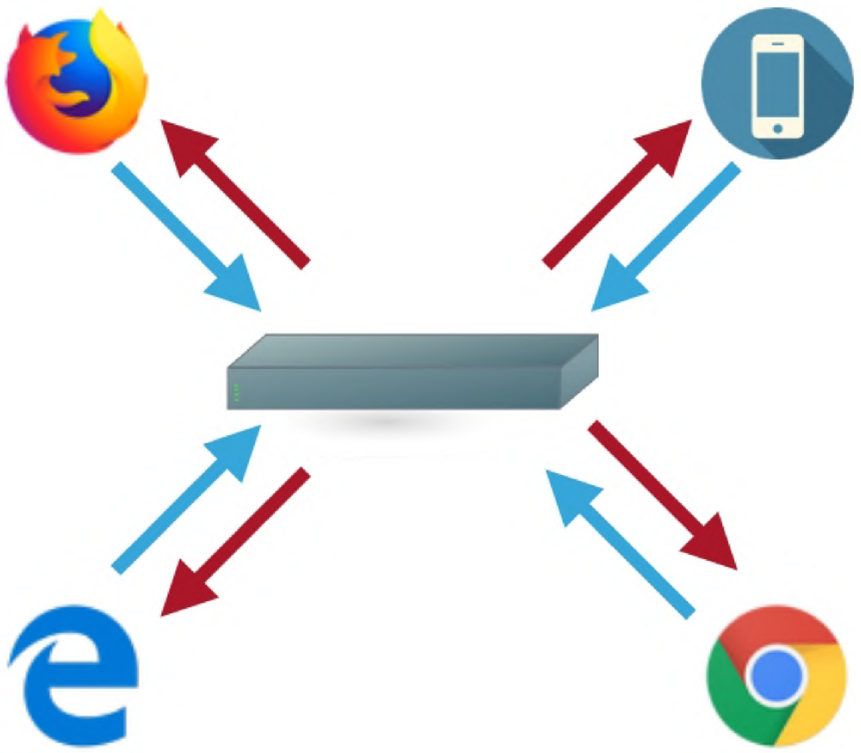
BRAINSTORMING

STATE OF THE ART RESEARCH



IDEAS DISCUSSION WITH ADVISOR

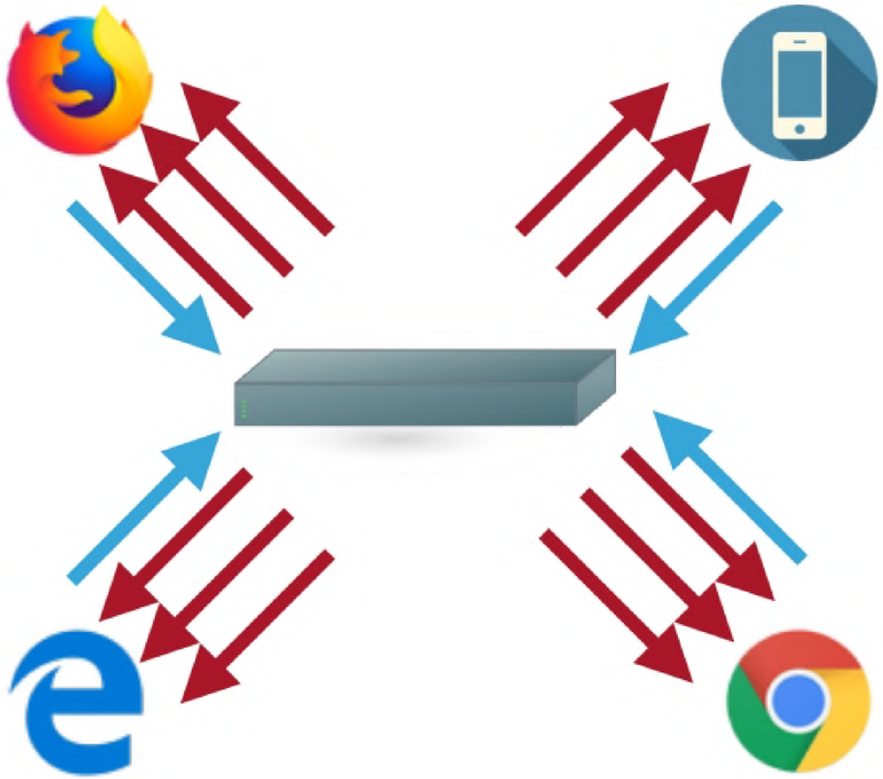
MULTIPOINT CONTROL UNIT (MCU)



- Participants send their media to the server
- Participants receive others media in a single stream, mixed by the server
- ✓ Clients need to handle a single remote stream
- ✓ Server performs transcoding
- ✓ Low download link required
- ◉ CPU intensive in server side
- ◉ High latency
- ◉ Fixed remote participants representation
- ◉ Low flexibility in client side

Source taken from: Presentation from Open Sips Summit 2018 Amsterdam (link here)

SELECTIVE FORWARDING UNIT (SFU)



- Participants send their media to the server
- Participants receive others media in separate streams, one each
- ✓ Server simply routes. High throughput, low latency
- ✓ Low CPU usage in server side
- ✓ Client can decide what streams to receive
- ✓ Client/Server can choose quality for each stream
- ◉ Higher download link required
- ◉ No transcoding

Source taken from: Presentation from Open Sips Summit 2018 Amsterdam (link here)

ACTORS

Guest

Does not have account.
Basic functionalities (voice and text chat, movement, ...)

Registered User

Has account.
History of Actions.
Extra Role Permissions.

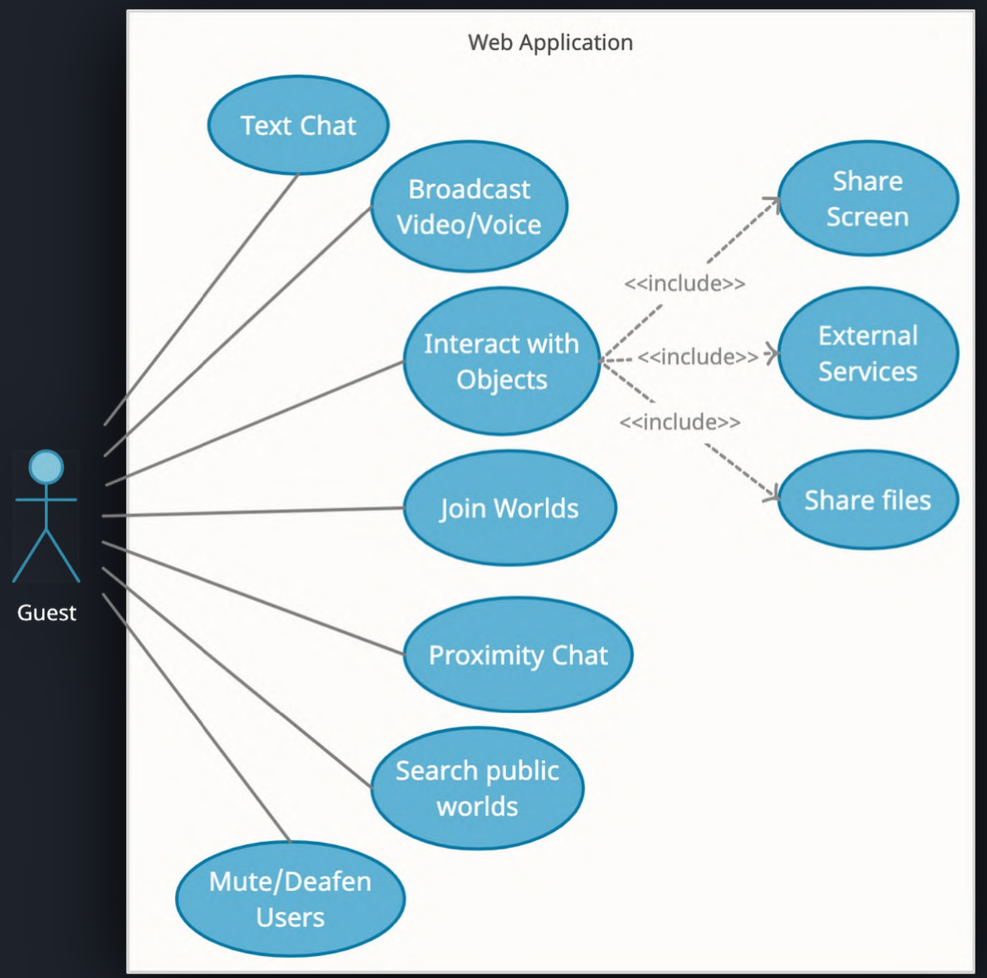
World Creator

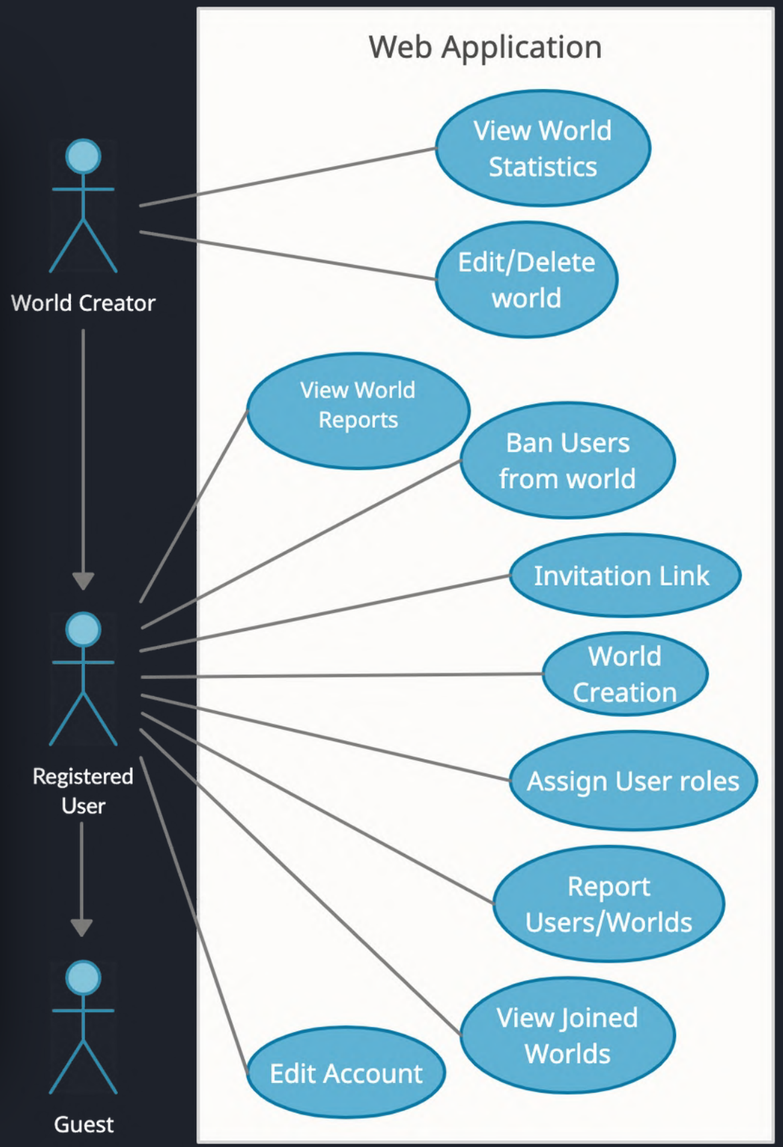
Edit and manage worlds.
View world statistics.

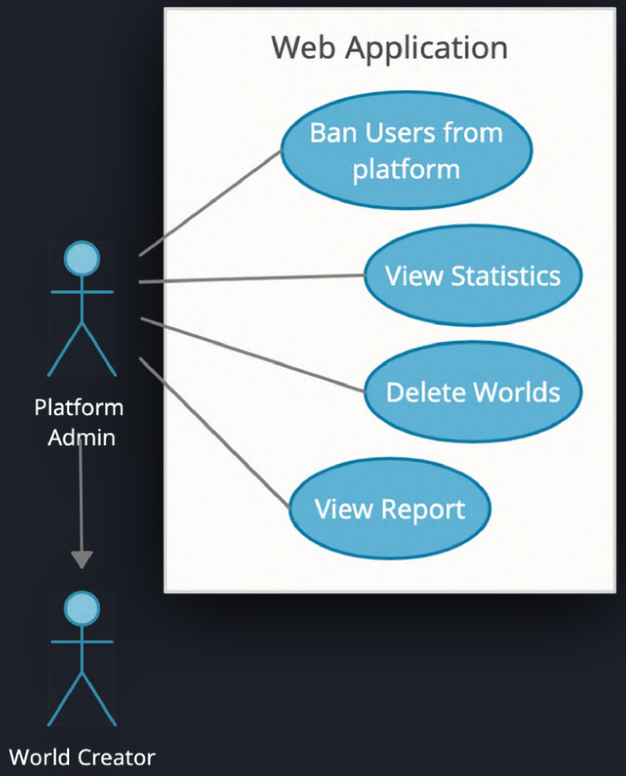
Platform Admin

Monitor everything.
Visualize global statistics.

USE CASES


Web Application
Text Chat
Broadcast Video/Voice
Interact with Objects
Join Worlds
Proximity Chat
Search public worlds
Mute/Deafen Users
Share Screen
External Services
Share files
<<include>>
<<include>>
<<include>>
Guest


Web Application
World Creator
Registered User
Guest
View World Statistics
Edit/Delete world
View World Reports
Ban Users from world
Invitation Link
World Creation
Assign User roles
Report Users/Worlds
View Joined Worlds
Edit Account


Web Application
Platform Admin
World Creator
Ban Users from platform
View Statistics
Delete Worlds
View Report

Functional Requirements - Business Rules

1. Separation Between Platform Admin Actor and others (Registered Users, Guests and World Creator)
2. Roles with Permissions for each World for each User. Default Ones:

- **Member**
- **Speaker**
- **Moderator**

Functional Requirements - Authentication and Authorization Level

1. Login and Guest Joining
2. External Authentication Services (GoogleAuth)
3. Limited Functionalities to Guests
4. Worlds Privacy (Can be Public or Private)

Functional Requirements - Administrative Functions

1. Access to a Platform Admin Page
2. Visualize Statistics
3. Platform Management

Functional Requirements - Reporting

1. Issue and Bug Reporting through GitHub or Discord.

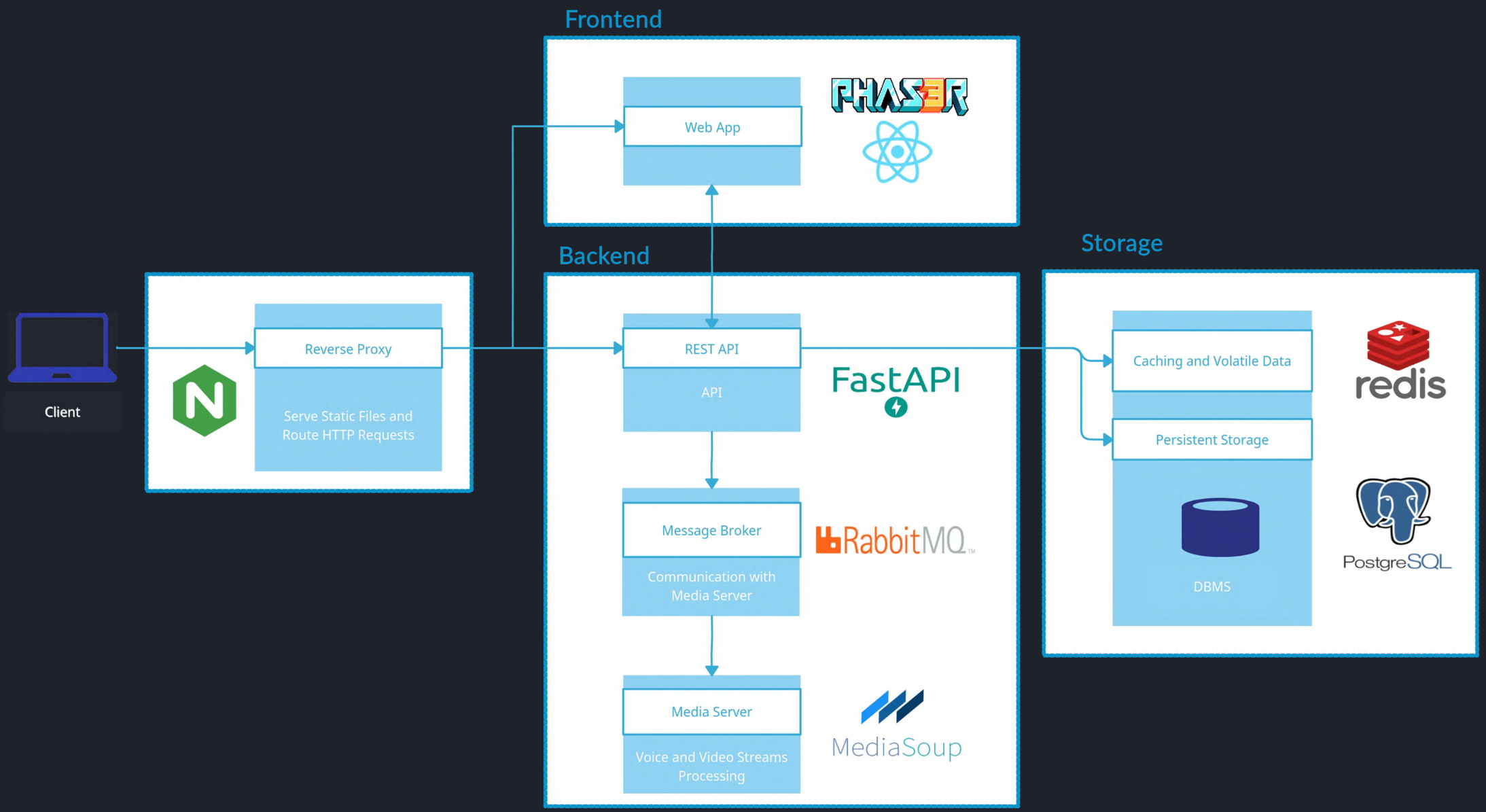
Functional Requirements - Historical Data

1. User logs in/out
2. Interaction with objects
3. Entering or leaving rooms

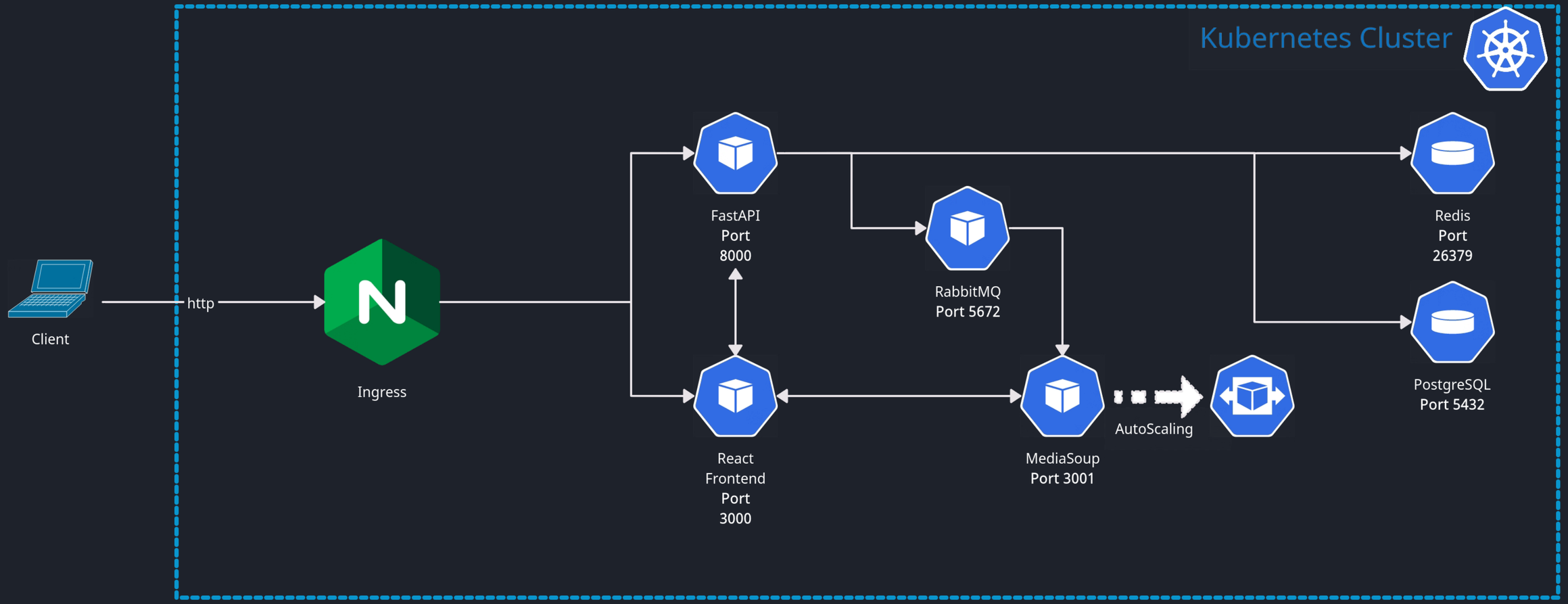
Non-Functional Requirements

- Usability
- Performance
- Availability
- Scalability
- Recoverability
- Compatibility

SYSTEM ARCHITECTURE



DEPLOYMENT DIAGRAM



PROTOTYPE PRESENTATION

Prototype Link