Project Proposal: Rendezvous

Group Components

Valerio Vinciarelli

Matricola: 1193280

Email: vinciarelli.1193280@studenti.uniroma1.it

Introduction

Rendezvous is a cloud service that allows registered users to contact other users to chat or make video calls.

The service allows users to register and a registered user can create video rooms, chat rooms or both by inviting other users via invitation links.

The service does not store chats and videos, but stores registered users and active rooms.

Automatically, a room closed by the creator is deleted from the system.

Topic [R1]

This project has as its topic the development of a simple service implemented through microservices.

The microservices currently identified are the following:

- Rendezvous UI: web page to access the service, a kind of welcome page through which users can register, log in and open rooms

- Rendezvous Users MS: microservice for user management

- Rendezvous RoomRouter MS: microservice for room management

- Rendezvous RoomServer MS: server microservice to manage users connected to a room (a RoomServer is a room)

- Rendezvous RoomClient MS: client microservice for connecting a user to a room (RoomServer)

Cloud environment [R2]

The microservices will be implemented in python and HTML / JavaScript.

The routing, login and service access services will implement RESTful APIs.

Subsequently a Docker container will be created for each component and eventually pushed to https://hub.docker.com/.

The cloud environment will then be implemented through Kubernetes, creating the appropriate resources to manage the microservices and the database will be implemented through a StatefulSet.

Ingress will be created for access to the web pages (welcome, login, rooms).

The service deployed on kubernetes implements autoscaling policies where necessary.

A first implementation will be locally via Minikube or Kind (with bootstrap kubeadm).

A second implementation can be uploaded to Google Cloud or AWS.

Tests [R3]

This phase of validation and testing of the service will be designed on these actions:

- Load Test: to carry out checks on the correct functioning of the autoscaling, generating fake clients that connect to the service

- Performance Monitoring: using monitoring tools (e.g. Grafana, Prometheus) the state of the cloud environment will be checked