Relatório Projeto ASINT

Grupo 8 (João Amado 79185, Manuel Monteiro 79138)

Arquitetura do Sistema

O sistema implementado é uma aplicação Web acessível através de um browser. Esta aplicação faz o gerenciamento do acesso dos utilizadores do IST ás salas disponíveis na instituição. O sistema é operado por dois diferentes tipos de utilizadores: administradores e utilizadores. Os administradores podem: observar quem esta em cada sala no momento, ver a história de todos os check-ins/check-outs desde o seu início do e enviar uma mensagem de texto aos utilizadores de uma determinada sala. Os utilizadores conseguem procurar as salas do IST disponíveis e fazer o check-in/check-out associado à entrada ou saída delas. Dentro de uma sala o utilizador consegue ver os outros utilizadores que também estão a utilizá-la no momento e ver as mensagens que os administradores enviam para a sala.

Através da Figura 1 é possível ter uma ideia da arquitetura geral do sistema.

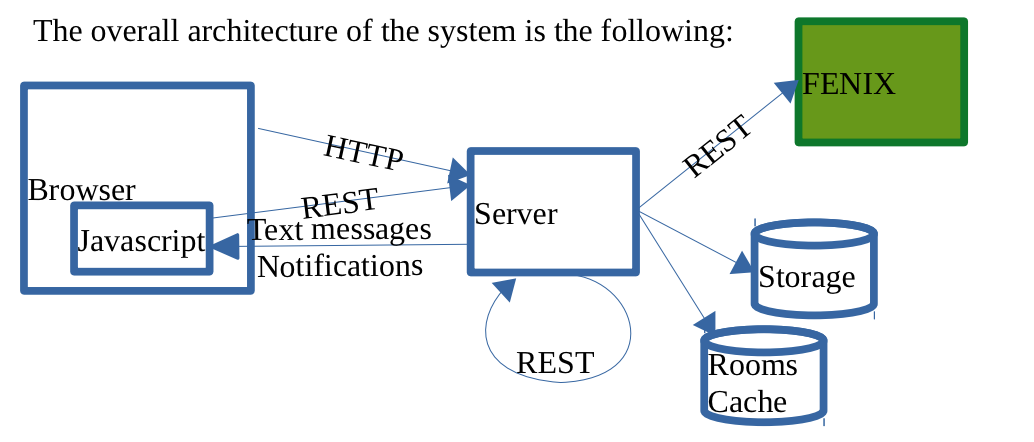


Figura 1

Os elementos principais do sistema são: servidor, cliente e base de dados. O servidor foi escrito em Node.js, de forma a ser acedido pelos clientes foi implementada uma REST API que é descrita na secção seguinte. O sistema comunica com o FENIX IST pela sua REST API e através desta é feita a autenticação dos utilizadores e a procura de salas disponíveis. O cliente foi desenvolvido em Vue.js que comunica com o servidor através da sua REST API. Na base de dados, implementada em MongoDB, é armazenada a informação persistente do sistema como os log de check-ins/check-outs dos utilizadores, as mensagens enviadas pelos administradores, dados do administrador, etc.

REST API implementada

Desseguida fazemos uma breve descrição das várias rotas da REST API implementada no servidor:

Método HTTP, rota:

GET / - Rota “home” que retorna o ficheiro index.html da pagina da aplicação.

GET /login/admin - Rota que recebe o username e password da tentativa de login do admin em base64 através do header HTTP authorization, da seguinte forma:

authorization: Basic base64(username : password)

Caso a autenticação seja bem-sucedida é retornada uma mensagem que faz set de uma cookie de nome 'fwa-authorization-admin' com o campo ll que contém a data em que o login foi executado na base de dados. Caso contrário é retornada uma mensagem com o status code 401.

GET /login/user - Rota que recebe o access code do FENIX IST através do URL no campo code:, com este código, a informação de autêntição da nossa aplicação o servidor vai fazer um pedido POST para a REST API do FENIX a pedir o authorization code associado ao utilizador que se pretender registar. Se este pedido for respondido correctamente o servidor vai receber uma mensagem com o accesss e o refresh token associdado ao utilizador que vai armazenar numa cookie chamada 'fwa-authorization' que com a resposta da rota “home” para qual o utilizador é redirecionado é enviada para o utilizador. Caso acontecer algum erro o utilizador é redirecionado para a rota /login/error.

GET /login/error - Rota para casos de erro que retorna um JSON com o campo error set com a mensagem de erro.

GET /logout - Rota que retorna uma mensagem que limpa as cookies geradas por esta aplicação 'fwa-authorization' e 'fwa-authorization-admin'.

GET/state - Rota que retorna um JSON com as informações do utilizador que se autenticou quer seja este admin ou utilizador de salas. Como primeira etapa deste processo o servidor verifica o campo signedCookies das mensagens para confirmar que está a lidar com um utilizador autenticado, este passo é feito em todas as rotas seguintes descritas.

GET /checkio/history- Rota que retorna um JSON com todos os logs (in e out) registados nas salas. Esta funcionalidade é especifica para o administrador.

POST /checkio/in - Rota que recebe no corpo da mensagem um JSON que tem os campos: roomId, que representa a sala que o utilizador pretende entrar; roomName, que é o nome da sala que o utilizador pretende entrar; username, que é o numero de identificação do IST do utilizador que pretende entrar na sala: displayName, que é o nome do utilizador que pretende entrar na sala. É retornado uma mensagem JSON com todos os utilizadores que no momento do registo estão dentro dessa mesma sala. Caso um utilizador esteja no momento do registo registado noutra sala é feito o check-out dessa sala e o check-in na pretendida.

POST /checkio/out - Rota que recebe no corpo da mensagem um JSON que tem os campos: roomId, que representa a sala que o utilizador pretende sair de; username, que é o identificador do IST do utilizador que pretende sair da sala. Caso tenha acontecido algum erro é retornado esse erro no campo da mensagem de resposta.

GET /rooms/find - Rota que recebe no campo query.search dos headers da mensagem o nome da sala a procurar. Primeiro é verificado se esta procura está em cache na base de dados, caso afirmativo esta é diretamente retornada ao utilizador num JSON na mensagem de resposta, caso contrário a procura é feita através de uma troca de mensagens entre o servidor e a rota GET /space/{id} da API do FENIX e no final desta é retornado um JSON ao cliente as salas obtidas e processadas num JSON.

GET /rooms/users – Rota que retorna um JSON com todos os utilizadores que no momento estão registados em alguma sala da aplicação. Funcionalidade especifica dos admins.

POST /rooms/{id}/messages - Rota que retorna um JSON com todas as mensagens e utilizadores loggados no momento na sala definida pelo parâmetro id presente no URL. Esta é uma funcionalidade especifica dos utilizadores, e apenas os utilizadores checked in numa determinada sala podem pedir as mensagens dessa sala, caso contrário uma mensagem com o status 401 é retornado.

PUT /rooms/{id}/messages – Rota que introduz na base de dados uma mensagem passada no body da mensagem num documento JSON no campo msg: que o admin envia para a sala identificada pelo parâmetro id no URL.

Tecnologias/bibliotecas utilizadas

No servidor:

A framework base que foi utilizada para desenvolver o servidor foi Express 4.16.1, esta é uma Framework para aplicações web mínima e flexível. Para a comunicação com a base de dados por parte do servidor foi utilizada a framework mongoose que através dos seus modelos de esquemas definidos os objetos que utilizamos na base de dados. As bibliotecas body-parser e cookie-parser como middleware para trabalhar com as mensagens de request´s e as cookies dos broswers respetivamente. A biblioteca axios funciona com um cliente HTTP baseado em promessas para nodejs que foi utilizada para fazer pedidos HTTP no servidor.

No cliente:

Para desenvolver a interface do cliente utilizámos a Framework vue.js, esta baseia-se em componentes cada uma dedicada a uma certa funcionalidade com a possibilidade de comunicarem entre si. Este código é pre compilado no servidor e depois é enviado para o cliente.

Base de dados:

Para a base de dados utilizámos o sistema NoSQL MongoDB. Fizemos esta escolha devido a neste programa não existir grandes relações entre os dados persistentes que pretendemos armazenar e também devido à fácil integração com a forma de manipulação dos dados JSON que utilizámos no projeto. Outra opção poderia ser um sistema SQL como mySQL com o qual também conseguiríamos o mesmo objetivo.

Interface do sistema para o utilizador

Em relação à interface do utilizador pretendemos desenhá-la o mais simples possível. Na página inicial o utilizador é encaminhado para o Login e aí tem a opção de escolher se pretende fazer o login como utilizador ou admin. Caso a autenticação de um administrador tenha sido efetuada com sucesso este deparasse com uma página onde através de uma checkbox tem a opção de ver ou esconder os checks in/out feitos e por debaixo disto aparece uma lista de salas em que existe utilizadores registados nelas no momento e em cada uma existe um formulário onde é possível enviar uma mensagem para essa sala. No final da página existe um botão de logout que permite o admin terminar a sua sessão no sistema. Caso a autenticação seja feita por um utilizador este primeiro é reencaminhado para uma página do FENIX que lhe vai pedir o respectivo username e password caso as informações estejam correctas, o utilizador passa para uma página onde consegue ver a sua informação do FENIX, a sala onde está registado no momento com os outros utilizadores que também estão nessa sala e as mensagens que a sala recebeu. Existe também um formulário de pesquisa de salas por nome, onde perante a procura, os utilizadores vão ser mostrados os resultados obtidos como o nome se assemelha de alguma forma à sua procura. Quando o resultado da procura estiver disponível as salas obtidas são mostradas em forma de lista onde para cada sala existe a opção de entrar se não estiver já nessa sala ou de fazer logout se estiver dentro dessa sala. No final existe o botão de logout.

Implementação do sistema na cloud da Google

A ultima etapa do projeto foi implementar o sistema desenvolvida na cloud da Google. Para isso iniciámos no nosso espaço da cloud 4 instancias de virtual machines, 3 para a base de dados MongoDB (1 servidor principal, outro de backup e outro arbiter) e outro para o correr o servidor do programa. Nesta ultima foram instaladas todas as tecnologias necessárias para correr o servidor como o npm e as bibliotecas necessárias à sua utilização, nesta é o servidor está a correr.

Integração com o sistema FENIX

Este programa foi integrado com o sistema FENIX em dois aspectos: na autenticação dos utilizadores e na procura de sala no IST.

Manipulação dos dados

Como é referido anteriormente no projeto grande parte das manipulações dos dados é feita através de JSON. Tanto na troca de mensagens entre cliente/servidor como o armazenamento dos dados na base de dados.