Aplicações Distribuídas sobre a Internet

Reporte do Projeto - Access control system (Parte 2)

1^{nd} **Semester 2021/2022**

José Rocha nº 194304

Index

1 Init

A classe *init.py* constrói a nossa aplicação Flask e cria a nossa BD via SQLAlchemy. É criado um endpoint para a classe *User.py* e outro para a classe *Gate.py*.

Esta classe é extendida pelas restantes classes da nossa aplicação Flask e como tal foi definido também um dicionário de erros para lidar com os resultados dos requests aos serviços e BD.

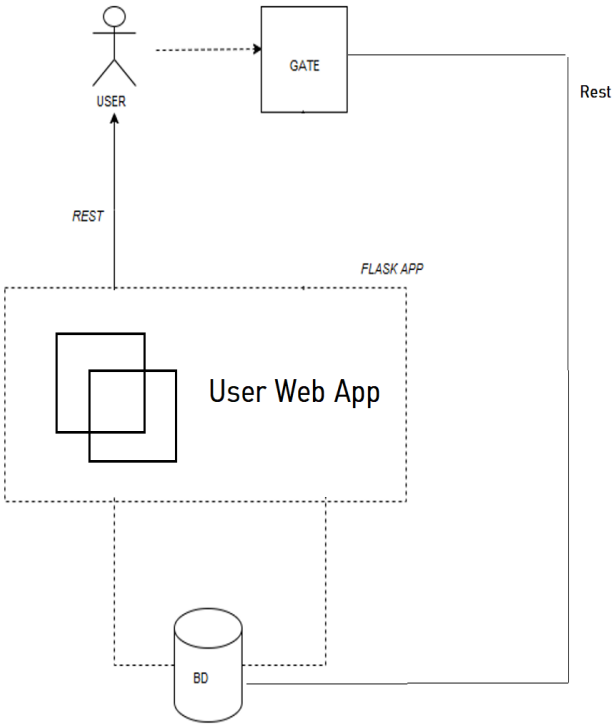

Figure 1: Modelo do sistema de Gates

2 User.py

A classe *user.py* é ponsável por toda a gestão de interações do utilizador com o sistema. Nesta aplicação web, após uma autenticação bem sucedida, será possível ao utilizador consultar as suas estatísticas de entradas e gerar um QRcode para entrar nas Gates do Fenix.

Esta aplicação tem 3 páginas HTML: index e qrcode. No index estão as funcionalidades de Autenticação e as de geração de um QRcode e de consulta de estatisticas. Esta duas só ficam disponíveis depois de um utilizador se autenticar com sucesso.

Abaixo está a descrição dos serviços fornecidos por esta aplicação.

2.1 Homepage

@app.route('/', methods=['GET'])

Input: *nothing*
Output: *user_ index.html*

Funcionalidade: Mostra ao utilizador a homepage da aplicação

2.2 QR Code

@app.route('/qrcode', methods=['GET'])

Input: *nothing*
Output: *qrcode.html*

Funcionalidade: Mostra ao utilizador a página para geração do QRcode.

2.3 Gestão das Entradas

@app.route('/user-entry-history', methods=['GET'])

Input: *string(FenixID), Booleano(Entrada válida ou inválida)*
Output: *JSONIFY da entrada criada*

Funcionalidade: Adiciona uma entrada (ou tentativa de) no sistema.

2.4 Estatísticas

@app.route('/user-stats', methods=['GET'])

Input: *string (FenixID)*
Output: *String (lista das Entradas)*

Funcionalidade: Lista para um utilizador todas as suas entradas.

2.5 Login

@app.route('/login', methods=['GET'])

Input: *nothing*
Output: *Redirect*

Funcionalidade: Inicia o processo de Login no Fenix

2.6 Callback

@app.route('/callback', methods=['GET'])

Input: *nothing*
Output: *Redirect to profile*

Funcionalidade: Recebe a resposta do Fenix ao Login e guarda os dados necessários para validar o utilizador a entrar na aplicação

2.7 Profile

@app.route('/profile', methods=['GET'])

Input: *nothing*
Output: *Redirect to indez.html*

Funcionalidade: Valida o utilizador na BD e redireciona para a homepage.

3 Gate_app

A classe *gate.py* representa uma Gate no sistema e permite ao utilizador entrar fisicamente no fenix. A Gate lê um QRcode e valida se o usercode enviado corresponde a um utilizador no sistema. Caso seja, a gate é aberta mostrando na aplicação Web um semáforo verde, caso contrário a entrada não é válida e o semáforo continua vermelho.

Abaixo está uma descrição dos serviços implementados pela Gate:

3.1 Homepage

@app.route('/', methods=['GET'])

Input: *nothing*
Output: *gate_index.html*

Funcionalidade: Mostra ao utilizador a homepage da aplicação

3.2 Gates

@app.route('/gates', methods=['GET'])

Input: *nothing*
Output: *JSON list com as gates do sistema*

Funcionalidade: Mostra uma lista com as gates

3.3 QRCode

@app.route('/qrcode', methods=['POST'])

Input: *JSON String (Usercode)*
Output: *Resultado válido ou inválido*

Funcionalidade: Verifica se o QRCode lido pela gate existe no sistema. Se existir, envia uma mensagem de validez, caso contrário envia uma mensagem de invalidez.

3.4 Nova entrada

@app.route('/qrcode', methods=['POST'])

Input: *String (FenixID), Booleano (iValido*
Output: *JSON string*

Funcionalidade: Quando um QRcode é validado, é adicionado no sistema o histórico dessa entrada.

4 Gate_data

A classe *gate_ data.py* representa o modelo da Gate na nossa base de dados. Esta classe não sofreu alterações face ao projeto anterior.

5 User_data

A classe *user_ data.py* tem duas tabelas, a tabela User que guarda o registo dos utilizadores no sistema; e a tabela UserEntryHisotry que guarda o histórico dos acessos do utilizador ao sistema. Para um utilizador queremos guardar o seu FenixID, o usercode que lhe permite aceder ao sistema, o seu token de autenticação e uma flag que nos diz se ele é administrador do sistema ou não. Se for, terá acesso a mais operações de consulta na aplicação.

Quanto ao histórico de entradas, queremos guardar o FenixID do utilizador em questão, guardamos o registo da Gate por onde ele entrou, guardamos um indicativo a dizer se a entrada foi válida ou inválida (quando o QRcode não é validado com sucesso) e guardamos o Timestamp da ocorrência.

6 Conclusão

Em jeito de balanço, gostaria de referir que há vários aspectos do trabalho feito que gostaria de ver melhorados. Por exemplo, na autenticação do utilizador não consegui colocar o redirect a funcionar para o serviço de "/login" quando o utilizador carrega no botão de "Autenticar". O processo funciona, mas inicialmente tenho de ser eu a fazer a chamada (colocar no browser) o endereço "localhost:5000/login". Este detalhe impediu que um dos pontos mais importantes desta parte do projeto ficasse completamente correto, mas ainda asism acho que a autenticação é possível e como tal o resultado é positivo.

Uma outra situação prende-se com o facto de não guardar os dados do FenixID do utilizador para criar o QRCode, ou seja, utilizo apenas o UserCode que a BD gera. Tenho a percepção que esta situação não está correta, mas também aqui fico satisfeito com a implementação de validação do QRcode via web estar a funcionar.

Por fim, gostaria de ter investido mais tempo "normalização" da API pois não estou satisfeito com a assinatura de alguns dos serviços em questão.

De qualquer forma, acho que as principais funcionalidades foram conseguidas, este projeto permitiu-me dar-me um melhor enquadramento dos conceitos de Front-End, Back-End, server-side, client-side e também de algumas técnicas de programação web, nomeadamente ao nível das dinâmicas das web-pages com a integração de Ajax e javascript no código.