Universidade
Europeia
LAUREATE INTERNATIONAL UNIVERSITIES

Relatório de Projeto Wifi Mapper

Universidade Europeia
LAUREATE INTERNATIONAL UNIVERSITIES

Título: Wifi Mapper

Participante:

Número	Nome
50037065	João Calapez

Introdução

Este trabalho visa como objetivo a avaliação de competências às unidades curriculares de programação web e sistemas de informação geográficos, para tal foi criado um projeto denominado de wifi mapper, que demonstra pontos de acesso wifi que são introduzidos por um upload de ficheiro, ou manualmente por um utilizador.
Este projeto serve para explorar as capacidades do leaflet (biblioteca utilizada para gestão do mapa e objetos contidos no mesmo), com grandes volumes de dados.

Universidade Europeia
LAUREATE INTERNATIONAL UNIVERSITIES

Indice

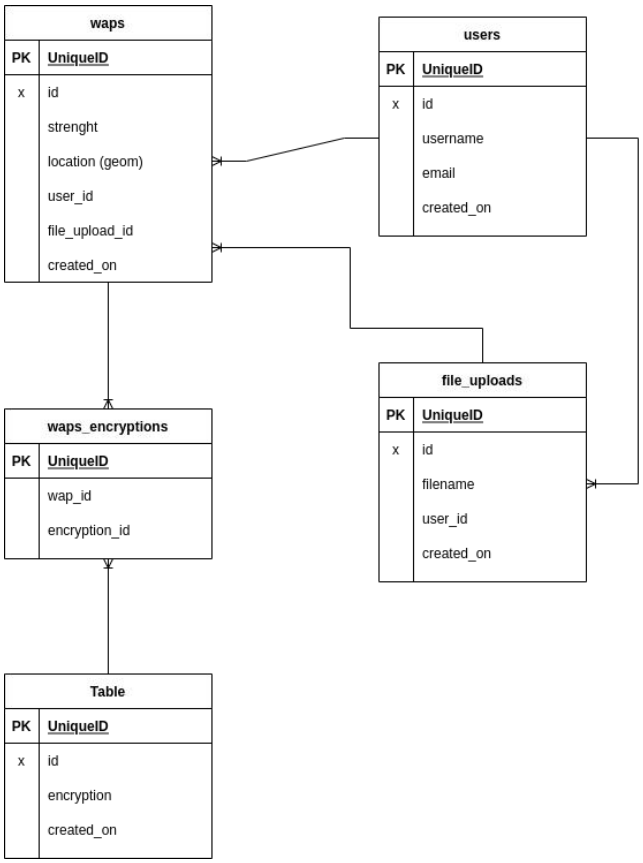
Introdução à arquitetura escolhida base de dados

Para este projeto, as primeiras key features a serem implementadas foram o upload de pontos geográficos na base de dados e a disponibilização desses pontos à API.

Por razões de logística e de tempo, a base de dados teve incrementos ao longo do desenvolvimento.

A tabela principal deste sistema é a tabela waps onde se encontram as localizações dos pontos conjuntamente com o seu bssid (nome de rede), tem também um user id parta rastrear o utilizador responsável pela criação ou o file_upload_id para identificar em que ficheiro o ponto foi carregado, estes dois ultimos pontos não foram implementados, pois não tinham alta prioridade no desenvolvimento.

Na figura seguinte podemos encontrar o desenho de como seria a base de dados final, mas não chegou a ser implementada a tabela file_uploads.



Com este modelo já é possível fazer algumas querys simples que nos dão alguma informação sobre a localização espacial não descuidando a performance do lado do cliente, sendo que neste ponto é de notar que as pesquisas feitas do lado do cliente tem um limite de 20 resultados, e os pontos obtidos dentro da bounding box tem um limite de 4000 resultados para poupar a memória do lado do browser do cliente.

A introdução do postgis na base de dados postgres permite fazer operações com cálculos geométricos com excelente performance e adiciona ao postgres funções tais como: ST_Buffer, ST_POINT e ST_Distance funções estas que foram fundamentais para o bom desenvolvimento deste projeto e para a performance do mesmo.

Para dados de teste foram carregados 100.000 pontos, e resposta da base de dados com cálculos relativos a esta quantidade de pontos foi bastante positiva.

Algumas das querys utilizadas no projeto:

```
`Select id, ST_X(location), ST_Y(location), bssid from waps
        where waps."location" &&
        ST_SetSRID(ST_MakeBox2D(ST_POINT(${st1}),ST_POINT(${st2})),4326) limit 4000;`
```

Esta query recebe duas localizações (st1, st2) que por sua vez é dividida em 2 pontos geometrics distintos ficando então: `ST_POINT(${st1}),ST_POINT(${st2})`

Estes dois argumentos são envolvidos na função ST_MakeBox2D que recebe dois argumentos, SW e NE que traduzindo para a nossa aplicação equivalem ao canto inferior esquerdo e canto superior direito, e gera um poligno com a dimensão do ecrã do cliente, este poligono é atribuído o srid 4326, envolvendo o poligno na função ST_SetSRID, que leva como argumentos um poligon separado de um srid.

Como podemos observar esta query pode devolver mais resultados que aqueles que o cliente pode aguentar com o seu hardware, por isso é feito um limite de 4000 registos para manter a fluidez do frontend ao processar esta informação.

```
SELECT buffer.id, buffer.bssid, distances.dist
FROM
  (
  SELECT id, bssid,
          ST_X(LOCATION),
          ST_Y(LOCATION)
   FROM waps
   WHERE waps."location" && ST_SetSRID(ST_Buffer(ST_POINT(${lat}, ${lng}), 0.01), 4326)
   ) buffer
INNER JOIN
  (
  SELECT id, ST_Distance("location", 'POINT(${lat} ${lng})'::GEOGRAPHY) AS dist
   FROM waps w
   ) distances ON buffer.id = distances.id
   order by dist asc;
```

Esta ultima query surgiu na necessidade de ordenar os pontos à volta de um determinado ponto selecionado. Numa primeira abordagem a query retornava todos os pontos contidos dentro de um buffer, mas o buffer é gerado tendo em conta um ponto e a ordenação dos resultados não é relativa ao ponto do buffer. De forma a garantir a ordenação correta dos resultados é criado um buffer de raio 0.01 e comparado com uma tabela de distâncias gerada a partir do ponto selecionado e comparada com todos os pontos dentro do buffer, desta forma o ponto selecionado tem sempre a distância 0 e a ordenação retorna sempre resultados ascentes.

Implementação Leaflet

Foram encontradas algumas limitações para este projeto utilizando o leaflet, nomeadamente a impossibilidade de ser atribuído um id a um marcador costumizavel, sendo que isto foi ultrapassado utilizando um label dentro do marcador na sua inicialização registando o id da base de dados para poupar memoria do lado do cliente com informação, não acedida de momento.

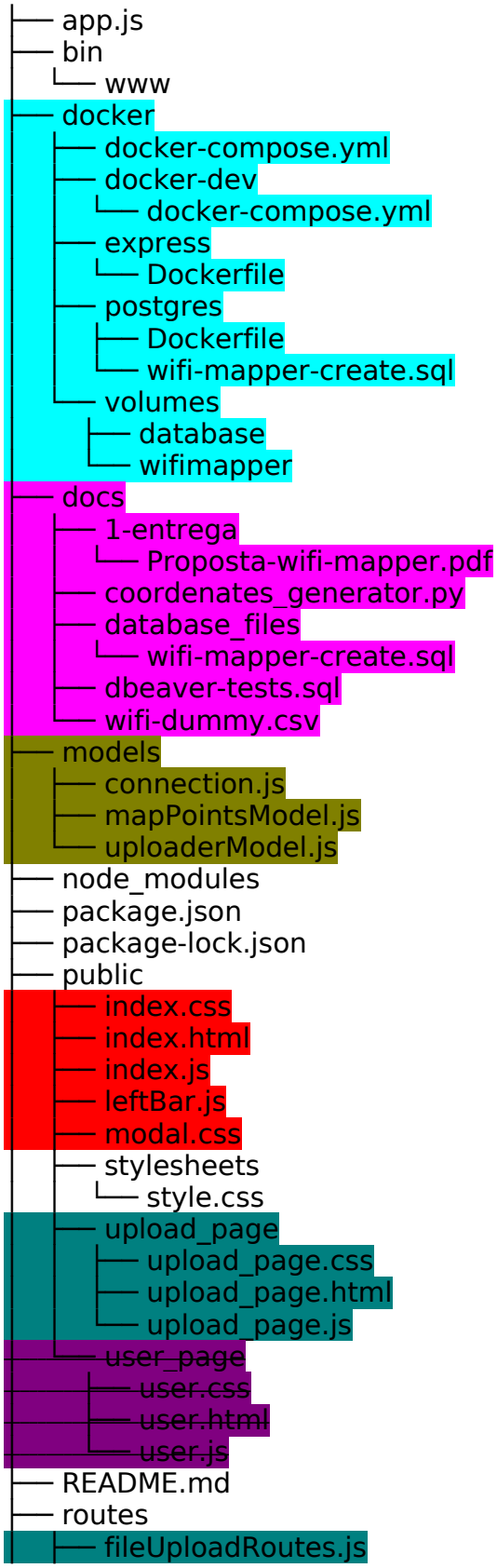
Estão implementadas duas listas em memória que são geradas sempre que há alterações no ambiente do cliente.

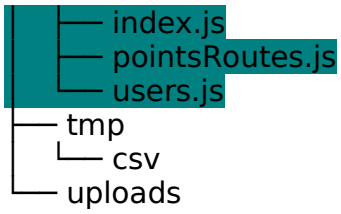
markersActiveHash - Contem todos os objetos marker que são gerados no mapa.

activeSideBarPointsHash - Contem todos os objetos que são apresentados na barra esquerda quando esta é chamada.

Para a determinação e preenchimento da bounding box foi criada a função getInBoundingBox() que não recebe argumentos, mas recolhe os pontos necessários para a criação de uma bounding box e chama a função getPointsWithinBoundingBox(point1, point2) que faz um pedido à api que determina a bounding box e devolve todos os pontos contidos na mesma.

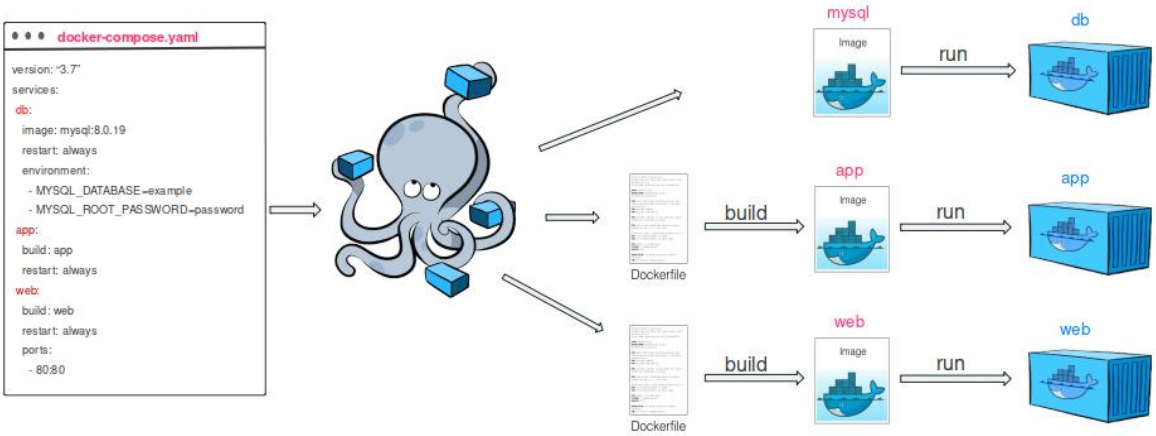
Descrição da organização de código do projeto




index.js
pointsRoutes.js
users.js
tmp
csv
uploads

Utilização de Docker

O projeto foi preparado para ser corrido num servidor através de docker, o docker cria contentores, de certa forma separados do os do servidor, e desta forma qualquer teste corrido num computador local em docker, há a certeza que irá correr no servidor, pondo assim preocupações de compatibilidades e versões de parte.
A pasta docker contem um docker-compose.yml que contem as instruções de construção do sistema, e que por sua vez aponta para os ficheiros Dockerfile, que contem as instruções de como os contentores são montados.



No docker-compose.yml estão também parametrizadas as variáveis de ambiente, por ex: “PGUSER, PGHOST” que fazem referência às variáveis de ambiente, utilizadas pelo postgres do lado do backend, e a construção da network do ambiente docker do projeto, sendo que existem 2 redes, uma interna que faz a ligação da base de dados ao backend, não permitindo ligações ao exterior, e uma externa que faz ligação apenas com o backend e frontend, desta forma isolamos a base de dados do mundo exterior.

Existe também uma pasta com o nome docker-dev que foi utilizada para o desenvolvimento, tendo esta contidas instruções para o lançamento de contentor docker com postgres com uma porta aberta localmente para acesso facilitado.

A pasta de volumes serve para mapear caminhos específicos dentro dos contentores para garantir persistência de dados especialmente da base de dados postgres.

Bibliotecas Javascript Utilizadas

Nome	Versão	Descrição
express-fileupload	1.2.1	Adiciona o parametro *files* *ex: req.files*
fast-csv	4.3.6	Permite a leitura de ficheiros csv
leaflet	1.7.1	Biblioteca que permite a utilização do leaflet
multer	1.4.3	Permite guardar ficheiros no sistema
pg	8.7.1	Coneção a Base de dados Postgres

Especificações API

Ponto de Upload

Titulo	Upload de ficheiro
	Aceita um ficheiro em formato csv no frontend e envia para processo.
URL	/api/file_upload
Tipo de Pedido	Post
URL Params	Obrigatórios:
	“data”:”file”
	“mimeType”: “multipart/form-data”
	Opcionais
	Nenhum
Mensagem sucesso	{status: true, message:”File is uploaded”, status:”200”}
Erro	{status:500}
Notas	

Localização de pontos

Titulo	Load de todos os pontos dentro de bounding box
	Recebe 2 pontos e calcula bounding box e pontos contidos dentro da bounding box
URL	/api/points/bb
Tipo de Pedido	Get
URL Params	Obrigatórios:
	p1: (lat, long)
	p2: (lat, long)
	Opcionais:
	Nenhum
Mensagem sucesso	{ status: 200, result: wap_points }
Erro	{status:500}
Notas	

Titulo	Pontos bssid nas proximidades
	Recebe um ponto (lat, long) e devolve todos os pontos contidos dentro de um buffer ordenados por distancia do ponto inserido.
URL	/api/points/nearby
Tipo de Pedido	Get
URL Params	Obrigatórios:
	lat
	long
	Opcionais:
	Nenhum
Mensagem sucesso	{ status: 200, result: wap_points }
Erro	{status: 500}
Notas	

Filtros

Titulo	Bssid filter
	Recebe um identificador bssid e retorna resultados da base de dados aproximados ou iguais ao identificador
URL	/api/points/filter
Tipo de Pedido	Get
URL Params	Obrigatórios:
	name: (bssid)
	Opcionais:
	Nenhum
Mensagem sucesso	{ status: 200, result: wap_points }
Erro	{status: 500}
Notas	

Titulo	Informação de Ponto por id
	Recebe um id e devolve informação sobre o ponto.
URL	/api/points/
Tipo de Pedido	Get
URL Params	Obrigatórios:
	id
	Opcionais:
	Nenhum
Mensagem sucesso	{ status: 200, result: wap_points }
Erro	{status: 500}
Notas	

Insert Update e Delete

Titulo	Insert de um ponto
	Recebe um objecto e cria registo na tabela de waps e na tabela wpas_encryptions, se encryption não existir, cria tambem a encryption.
URL	/api/points/insert
Tipo de Pedido	Post
URL Params	Obrigatórios:
	{name: newName, encyption: newEncryption, strength: newStrength, lat:newLat, long:newLong}
	Opcionais:
	Nenhum
Mensagem sucesso	{ status: 200, result: wap_points }
Erro	{status: 500}
Notas	

Titulo	Update ponto por id
	Recebe um objecto com um objecto e novos campos dentro compara campos alterados e faz update aos campos alterados
URL	/api/points/update
Tipo de Pedido	Post
URL Params	Obrigatórios:
	{object:oldObject, field1:newParam, ...,}
	Opcionais:
	Nenhum
Mensagem sucesso	{ status: 200, result: wap_points }
Erro	{status: 500}
Notas	Logica de processo foi delegada ao backend para não deixar esforço de processo do lado do cliente e para não revelar camadas de negócio.

Titulo	Remoção de um ponto
	Recebe um id e remove o id da tabela waps e da tabela waps_encryptions
URL	/api/points/delete
Tipo de Pedido	Post
URL Params	Obrigatórios:
	id
	Opcionais:
	Nenhum
Mensagem sucesso	{ status: 200}
Erro	{status: 500}
Notas	

KPI’s

Titulo	Contador de pontos totais em base de dados
	Devolve o numero total de pontios existentes em base de dados.
URL	/api/points/total
Tipo de Pedido	Get
URL Params	Obrigatórios:
	Nenhum
	Opcionais:
	Nenhum
Mensagem sucesso	{ status: 200, result: numberOfPoin}
Erro	{status: 500}
Notas	

Casos de Utilização

Nome: Fazer upload de ficheiro csv (Implementado)

Descrição: Como utilizador quero fazer upload de um ficheiro contendo pontos de acesso wifi e a sua localização agregada.

Pré-condições: O utilizador tem de efetuar login para o upload ficar registado no seu nome

Passo a passo:

1. O utilizador abre o ecrã upload
2. O utilizador faz login no ecrã de upload
3. O utilizador seleciona o o ficheiro a transferir e faz o upload do mesmo

Pós-condições: O utilizador recebe uma notificação pop up a informar que o ficheiro está a ser carregado

Nome: Aplicar filtros ao mapa para ver hetmap gerado pelos pontos registados (Não Implementado)

Descrição: Como utilizador quero aplicar filtros ao mapa para poder observar umheatmap gerado pelos pontos de acesso registados na base de dados.

Pré-condições: Não existem pré condições de utilizador

Passo a passo:

1. O utilizador clica num botão que ativa o heatmap.

Pós-condições: O mapa é alterado e passa a ser visualizado um heatmap em vez de pontos de acessos registados.

Nome: Consultar ponto de acesso (Implementado)

Descrição: Como utilizador quero a possibilidade de escolher um ponto de acesso registado e consultar o seu BSSID, força do sinal e coordenadas gps em texto.

Pré-condições: Não existem pré condições de utilizador

Passo a passo:

1. O utilizador escolhe um ponto de acesso no mapa e clica no mesmo
2. A informação é apresentada

Pós-condições: A janela de informação sobre o ponto de acesso ficará aberta até o utilizador efetuar um clic noutro lado do ecrã ou noutro ponto de acesso.

Nome: Criar Ponto de acesso (Implementado) Descrição: Como administrador quero poder criar um ponto de acesso diretamente no mapa sem a necessidade de ser criado por upload de ficheiro. Pré-condições: Utilizador tem de ter permissões de administrador Passo a passo: 1. O administrador seleciona uma coordenada no mapa. 2. São introduzidos os detalhes do ponto de acesso na janela pop up. 3. É selecionada a opção de gravar na janela pop up. Pós-condições: O ponto de acesso é guardado em base de dados.
Nome: Editar Ponto de acesso (Implementado) Descrição: Como administrador quero poder editar um ponto de acesso diretamente no mapa. Pré-condições: Utilizador tem de ter permissões de administrador Passo a passo: 1. O utilizador seleciona um ponto de acesso existe 2. O utilizador edita o ponto selecionado via janela pop up. Pós-condições: O ponto de acesso é modificado.

Notas finais

O projeto foi implementado com funcionalidades em falta, mas com conteúdo suficiente para produzir um produto minimamente viável, a autenticação que controlaria quem pode ou não fazer alterações aos dados ficou em falta, tendo sido prevista a utilização de oauth2 que permitia agregar a cada pedido um token de sessão que seria verificada a sua validade pela api, e a utilização do valor de força do sinal de cada ponto gerado no mapa para a construção de uma layer heat map para observar as forças de sinal dominantes na região observada.
A formatação da página de upload ficou em falta, embora esta se encontre funcional.

Como opinião pessoal, foi muito gratificante e divertido trabalhar neste projeto, pois foi a minha primeira vez a trabalhar numa solução full stack, é muito interessante o caminho da informação entre base de dados, backend e o envio de dados pela api e a gestão desses mesmos dados do lado do frontend. Embora o projeto tenha ficado funcional, mas incompleto em alguns sentidos este projeto será algo que poderei vir a trabalhar no meu tempo livre para aumentar as minhas capacidades.

Entre alterações que gostava de fazer à api estão, a possibilidade de gravar no modelo de dados o objeto json marker do leaflet, utilizando colunas json, para evitar gerar o objeto sempre que chamado, desta forma tinha melhor controlo sobre o objeto do lado do backend, a introdução dos utilizadores como referido acima, e a criação de mecanismos de gestão de uploads como descrito no 1º capitulo deste relatório.