Universidade do Minho

Mestrado Integrado em Engenharia de Telecomunicações e Informática

Projeto de Telecomunicações e Informática I

Relatório Fase Final

Grupo 2

Guimarães

30/01/2018

Universidade do Minho

Índice

Universidade do Minho

Índice de figuras

Universidade do Minho

Índice de tabelas

Lista de abreviaturas

AP – *Access Point*

RP – *Reference Point*

RSSI – *Received signal strength indication*

MAC – *Machine Access Control*

KNN – *K-Nearest Neighbours*

1. Introdução

O presente relatório encontra-se dividido em duas fases. Inicialmente iremos falar de algumas mudanças que foram necessárias fazer, tendo em conta o que foi dito no relatório inicial.

Na fase de desenvolvimento pretendemos explicar as principais funcionalidades das três partes que constituem este projeto, ou seja, Servidor, Android e Web. Ainda neste tópico, iremos abordar as principais dificuldades pelas quais passamos ao longo da execução deste projeto, nomeadamente a falta conhecimentos no que toca ao desenvolvimento web e servidor.

No terceiro tópicos, iremos apresentar um teste de localização, utilizando o algoritmo por nós implementado, realizando uma comparação da posição obtida com a real. Pretendemos também explicar aqui quais as falhas existentes no algoritmo.

Na conclusão, fazemos uma reflexão acerca do trabalho feito ao longo do semestre.

2. Alterações realizadas

2.1. Software utilizado

No software utilizado, optámos por não fazer a base de dados em MySQL. Em deterioramento deste, preferimos utilizar MongoDB. A razão para optarmos por este foi o facto de as informações serem guardadas sob a forma de documentos semelhantes a JSON. A nossa base de dados MongoDB encontra-se armazenada na cloud, utilizando MongoDB Atlas.

2.2. Modelo de dados

O modelo de dados por nós pensado na fase inicial deste projeto sofreu algumas alterações, com o objetivo de facilitar a realização deste trabalho. Inicialmente, o modelo apresentado estava demasiado complexo, contendo informações que não foram necessárias armazenar na nossa base de dados.

O modelo relacional de dados encontra-se a seguir apresentado.

Espaco (nome_espaco,url_imagem,nome_Dono)

User (name,email,password,tipo)

FingerPrint (MAC,RSSI,point_x,point_y,nome_espaco,estado)

Na tabela 1, encontram-se explicadas as colunas de cada uma das tabelas.

Tabela 1 - Elementos do modelo de dados

Nome da tabela	Coluna	Objetivo
Espaco	nome_espaco	Representa o nome do espaco
	url_imagem	Diretoria da imagem no espaço, ou seja, onde ela se encontra guardada no servidor
	Nome_Dono	Permite associar um dono a um espaço por ele criado
User	name	Nome do utilizador
	email	Email do utilizador
	password	Password de acesso ao sistema
	tipo	Tipo de utilizador
FingerPrint	MAC	MAC do AP (Access Point)
	RSSI	Valor de RSSI da medição
	point_x	Coordenada do ponto x na fase offline
	point_y	Coordena do ponto y na fase offline
	nome_espaco	Associar a medição a um único espaço
	estado	Distinguir as medições da fase online e offline, true na fase online, false na fase online

3. Comunicação Aplicação Android – Servidor

Para a realização deste projeto, optamos por colocar o nosso servidor na Amazon.

Para que a comunicação ocorra é necessário associar o servidor a uma porta. Do lado do Android, apenas é necessário declarar o URL *(Uniform Resource Locator),* de onde faz parte a porta a utilizar, sobre o qual pretendemos realizar uma tarefa (exemplos: *get* e *post*). Para que tal seja possível, do lado do servidor é necessário definir as rotas a utilizar, tendo em conta o URL. É aqui que entra a utilização da base de dados, pois definimos a qual tabela pretendemos aceder, realizando uma das operações referidas anteriormente neste parágrafo.

4. Desenvolvimento

Ao longo deste tópico iremos abordar as principais funcionalidades implementadas, mostrando os layouts e/ou excertos de código que entendemos como importantes.

4.1. Android

No login, os utilizadores são diferenciados através do tipo correspondente. Cada tipo de utilizador possui funcionalidades diferentes, sendo que todos os tipos podem encontrar num qualquer espaço existente, podendo assim obter a sua localização. No menu de login é ainda possível escolher a opção para criar uma conta.

Na figura 1 encontra-se apresentado o *layout* correspondente ao login.

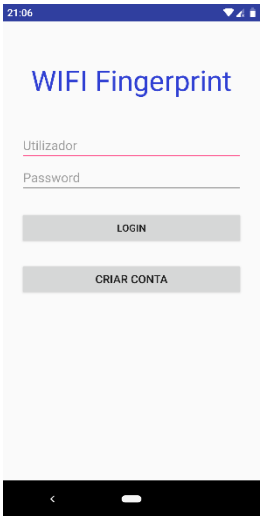

Figura 1 - Login

Após o login, utilizadores do tipo “Normal” ou “Premium” têm acesso às mesmas opções.

Na situação de um utilizador “Dono”, para além de poder fazer tudo o que os restantes utilizadores fazem, este pode também realizar calibração de um espaço e ainda consultar os utilizadores existentes. Existe ainda a opção “Gerir Conta”, que se encontra disponível para todos os tipos de utilizador, onde é possível alterar a *password.* Isto pode ser visto na figura 2. Este apenas pode calibrar os seus espaços, adicionando novos RP’s *(Reference Point)* que apenas ficam associados ao espaço que escolheu calibrar (figura 3 – escolha do espaço a calibrar). Qualquer utilizador pode entrar em qualquer um dos espaços existentes, de forma a obter a sua localização (figura 4 – entrar num espaço).

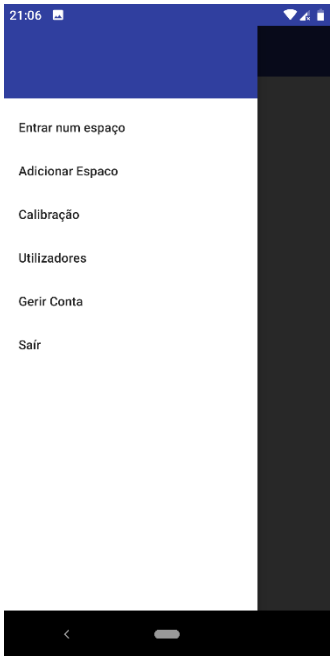

Figura 2 - Opções Dono

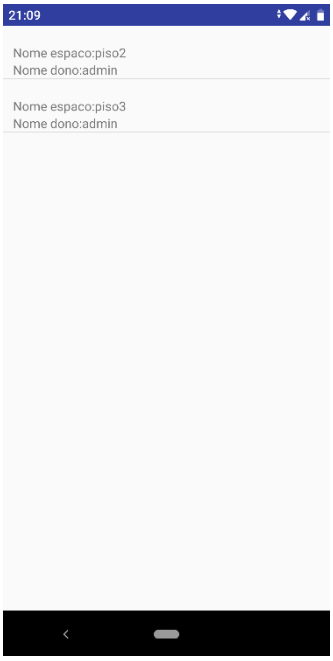

Figura 3 - Escolha do espaço a calibrar

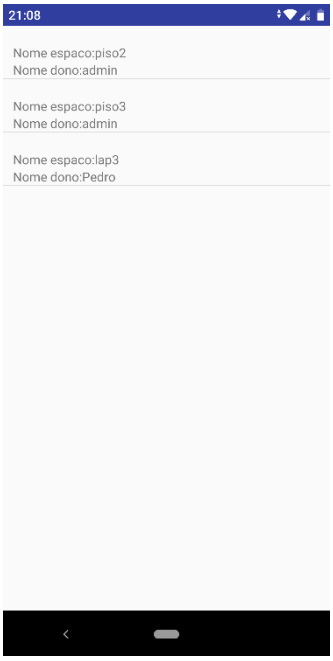

Figura 4 - Entrar num espaço

Ao realizar a calibração de um espaço, adicionando um novo RP, são enviados para a base de dados as coordenadas as coordenadas x e y, valor de RSSI *(Received signal strength indication),* o endereço MAC *(Machine Access Control)* do AP, nome do espaço associado e ainda o estado. Por cada novo RP adicionado, são calculadas 10 amostras, ou seja, uma para cada AP. Se se encontrarem disponíveis menos de 10 AP's no momento da leitura, o número de amostras em falta são enviadas a zero (x, y e RSSI a zero). Isto permite que não ocorram falhas nos cálculos realizados pelo algoritmo no servidor, que mais à frente será explicado.

4.2. Algoritmo de localização

Inicialmente o grupo decidiu adotar um algoritmo probabilístico que fosse o mais preciso na tentativa de obter resultados próximos dos valores reais. Porém, devido à má gestão do tempo, o grupo teve de optar por utilizar um método mais simples de implementar, contudo igualmente eficaz, o KNN (*k-nearest neighbours)*.

Em *wifi fingerprinting* o algoritmo KNN seleciona os k exemplos da fase offline mais próximos no espaço desejado.

A fase offline e online são compostas por vários vetores recolhidos num espaço. Cada um destes representa o *RSSI* medido para cada *AP* detetado. Cada vetor acima mencionado corresponde a uma posição no espaço, obtida através da latitude e longitude.

Na tabela 2, abaixo representada, podemos observar uma amostra da fase offline, com valores obtidos através da base de dados. Cada linha da tabela corresponde a um RP. E cada valor negativo corresponde ao valor do *RSSI* obtido num determinado RP para cada AP.

Tal como sugere a tabela 2, independentemente do número de AP's encontrados, o grupo decidiu manter um valor fixo de 10, como já no ponto anterior foi dito, para uma melhor implementação do algoritmo.

Tabela 2 - exemplo da fase de calibração, onde são acrescentados os RP's ao mapa

RP	AP1	AP2	AP3	AP4	AP5	AP6	AP7	AP8	AP9	AP10	X	Y
1	-86	-76	0	-72	-77	-48	-64	-90	-86	-87	33	40
2	-76	-72	-77	-48	-86	0	-90	-64	-86	-87	281	37
3	-87	-90	-76	-64	-72	-77	-48	-86	-86	0	31	382
4	-64	-48	-76	-86	-72	-86	-90	0	-77	-87	275	402
5	-49	-87	-87	-87	-70	0	0	-65	-77	0	143	38
6	0	0	-49	-65	-70	-77	-87	-87	-87	0	137	23
7	-47	0	0	0	0	0	0	0	-45	-76	855	354

Na tabela 3 encontra-se um exemplo da fase online.

Tabela 3 - exemplo da fase online

AP1	AP2	AP3	AP4	AP5	AP6	AP7	AP8	AP9	AP10	X	Y
-85	-80	-69	-60	-44	0	-87	0	0	-55	138	24

4.2.1. Explicação do algoritmo

Inicialmente, é medida a distância euclidiana entre cada ponto da fase offline com o da fase online.

É feita a raiz quadrada da soma das diferenças ao quadrado entre cada valor de *RSSI* da fase offline e online, para os respetivos AP's.

```
                offline.forEach(element => {
                    for(var i = 0; i < element.rssi.length; i++) {
                        //console.log(JSON.stringify(element.rssi));
                        //console.log(JSON.stringify(online.rssi));
                        if(element.rssi[i] != 0 || online.rssi[i] !=
0){
                            soma += Math.pow((element.rssi[i] -
online.rssi[i]), 2);  //online[0].rssi[i] ????
                        }
                        //console.log(soma);
                    }
                    //console.log(soma);
                    //console.log(Math.sqrt(soma));
                    var aux={x: element.x , euclid: Math.sqrt(soma)};
                    //console.log(aux);
                    somas[j]=(aux);

                    j++;
                    soma = 0;
                });
```

Figura 5 - Método que permite calcular a distância euclidiana entre cada RP da fase offline com o local onde se quer saber a posição, na fase online

Posteriormente é escolhido um valor para k. Após várias tentativas para a escolha do k ideal, o grupo chegou à conclusão que o número três era o que melhor.

É realizada a ordenação dos valores obtidos em cada distância euclidiana medida. Como o k escolhido foi o 3, o que equivale a dizer que queremos os 3 vizinhos mais próximos do ponto no qual queremos saber a localização, só os 3 primeiros valores serão escolhidos.

```
                somas.sort(comparator); // Array com cada distância
euclidiana ordenado por ordem crescente

                var valorK = 3; // Aqui escolhemos o valor de k
```

Figura 6 - método que ordena o array das distâncias euclidianas por ordem crescente

Os vizinhos mais próximos devem contribuir mais para a posição estimada do utilizador, por isso devem ter um peso maior. Calculamos o peso de cada vizinho como sendo 1 a dividir pela distância euclidiana de cada um dos vizinhos. Seguidamente, esse resultado é dividido pela soma de todos os k pesos, dando assim o peso final de cada um dos k vizinhos.

```
            for(var i = 0; i < valorK; i++){ // K  = 3 -> Os 3
vizinhos mais próximos

                peso[i] = 1/somas[i].euclid;
            }

            var somaPeso = 0;

            for(var i = 0; i < valorK; i++) {
                somaPeso += peso[i]; // Soma de todos os pesos
            }

            console.log("Soma peso: " +somaPeso);

            var pesoTotal = [];

            for(var i = 0; i < valorK; i++) {
                pesoTotal[i] = peso[i] / somaPeso; // Peso total
de cada vizinho
            }

            console.log("Peso total: " +pesoTotal);
```

Figura 7 - parte do código que nos permite atribuir pesos distintos a cada um dos vizinhos mais próximos, conforme a proximidade

Por fim, a localização estimada da posição onde se encontra o utilizador pode ser calculada como a média ponderada da localização dos vizinhos mais próximos, ou seja, é o somatório de cada coordenada, x e y, de cada um dos vizinhos a multiplicar pelo seu respetivo peso, calculado anteriormente.

```
                for(var i = 0; i < valorK; i++) {
                    longEstimada += (offline[somas[i].x - 1].long *
  pesoTotal[i]);
                    latEstimada  += (offline[somas[i].x - 1].lat *
  pesoTotal[i]);
                }

                console.log("Long: " +longEstimada);
                console.log("Lat: " +latEstimada);
```

Figura 8 - parte do código onde são calculados, ambos x e y

4.2.2. Teste ao cálculo da localização

Uma vez finalizado o algoritmo, seguiram-se uma série de testes para avaliar a precisão do mesmo. Os resultados obtidos pelo grupo foram razoáveis, mas só na primeira recolha da fase de calibração, uma vez que, como os professores fizeram notar, nas vezes seguintes à primeira recolha, não era feita a correta associação entre os APs para a medição das distâncias euclidianas, nunca podendo dar, por isso, um valor satisfatório.

Nas figuras seguintes, 9 e 10, apresentamos os resultados de um dos destes realizados. Aqui pode ser verificada a posição exata e a posição determinada pelo algoritmo.

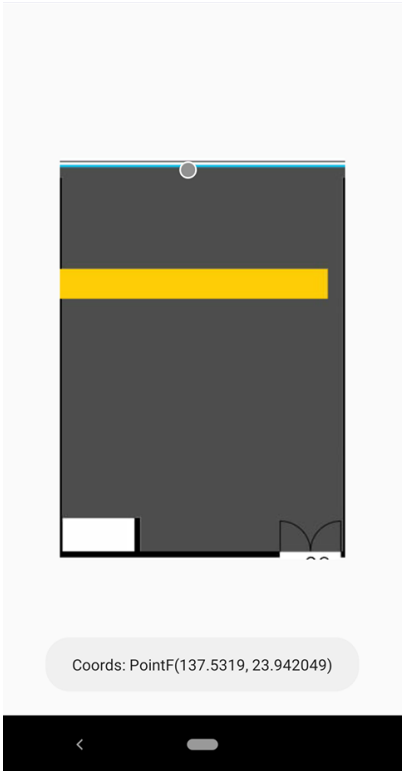

Figura 9 - posição exata

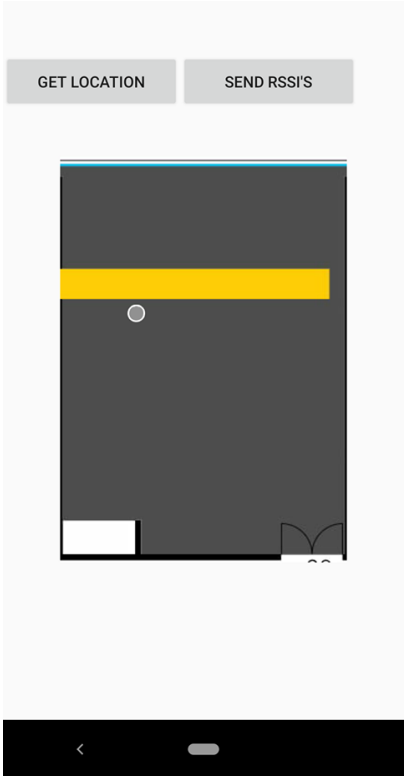

Figura 10 - posição calculada pelo algoritmo

Verificando os cálculos executados no servidor, pelo algoritmo, o valor de longitude obtido é de 81,3064 e o valor da latitude é 178,0275.

Considerando as dimensões da sala, onde foi executado este teste, como sendo 7x12 metros (x=7 e y=12 metros). Através deste mapa concluímos que os 7 metros correspondem a 300,5418 pixels na imagem.

Assim sendo podemos determinar a distância entre os dois pontos, posição exata e posição calculada:

X_{exato}=137,5319

Y_{exato}=23,9420

$X_{calculado}$=81,3064

$Y_{calculado}$=178,0275

$$d=\sqrt{(137{,}5319-81{,}3064)^2+(23{,}9420-178{,}0275)^2}=164{,}02$$

Convertendo este resultado para metros:

$$d=\frac{7*164{,}02}{300{,}5418}=3{,}82 \text{ metros}$$

Como podemos constatar, através do resultado obtido, que o nosso algoritmo apresenta uma eficácia bastante satisfatória. Isto porque o erro de imprecisão detetado foi de 3,82 metros, o que nos parece ser um valor satisfatório.

5. Conclusão

Ao longo desta fase de projeto, o grupo começou por fazer algumas correções para o poder elaborar com sucesso.

Ao longo deste projeto não nos foi possível realizar as principais funcionalidades que seriam de esperar ao nível da aplicação gestora. O grupo apenas se limitou na realização dos layouts da aplicação, daí não termos dado atenção a esta ao longo deste relatório.

Ao nível da aplicação de localização, ou seja, aplicação Android, pensamos que os objetos exigidos foram alcançados. A comunicação com o servidor foi estabelecida com sucesso, permitindo assim a realização do cálculo da posição do utilizador. Nem sempre o cálculo desta é determinada com êxito, como anteriormente referido neste relatório, existindo algumas limitações no algoritmo implementado.

Em suma, a realização deste projeto permitiu ao grupo a obtenção de novas competências, nomeadamente na programação em JavaScript, cujos conhecimentos eram praticamente nulos.