

Sensorização e Ambiente (1<sup>o</sup> ano do MEI)

**Monitorização de Acesso a Edifícios**

Relatório de Desenvolvimento

João Pereira  
PG47325

Luís Vieira  
PG47430

Pedro Barbosa  
PG47577

16 de maio de 2022

## **Resumo**

O presente trabalho prático, realizado no âmbito da unidade curricular de Sensorização e Ambiente inserida no perfil de Sistemas Inteligentes, visa o desenvolvimento de um Monitorizador de Acesso a Edifícios.

Este irá começar com uma introdução e contextualização sobre o tema. De seguida, o mesmo irá ser aprofundado através da concretização dos objetivos associados desde o início do projeto, especificação dos sensores utilizados, descrição, exploração e visualização dos dados recolhidos durante todo o projeto e será apresentado o sistema desenvolvido em função do tema e dos nossos objetivos. Finalmente serão apresentados os resultados finais de todo o projeto bem como discutido aquilo que foi o nosso trabalho ao longo do mesmo, retirando algumas conclusões e idealizando alterações e/ou implementações futuras para o mesmo.

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>2</b>
<b>2</b>	<b>Estrutura do Relatório</b>	<b>3</b>
<b>3</b>	<b>Objetivos</b>	<b>4</b>
<b>4</b>	<b>Sensores Utilizados</b>	<b>5</b>
<b>5</b>	<b>Dados Recolhidos</b>	<b>6</b>
5.1	Exploração . . . . .	6
5.2	Tratamento de Dados . . . . .	6
5.3	Visualização dos Dados Transformados . . . . .	8
5.4	Modelos de Machine Learning Utilizados . . . . .	10
5.4.1	DecisionTree Regressor . . . . .	10
5.4.2	Rede Neuronal Artificial . . . . .	11
5.4.3	Modelo final escolhido e resultados . . . . .	12
<b>6</b>	<b>Sistema Desenvolvido</b>	<b>13</b>
6.1	Arquitetura . . . . .	13
6.2	Funcionamento . . . . .	14
6.2.1	Start Activity . . . . .	14
6.2.2	Login . . . . .	15
6.2.3	Registo . . . . .	16
6.2.4	Main Activity . . . . .	17
6.2.5	Action Bar . . . . .	18
6.2.6	Chat . . . . .	18
6.2.7	Perfil . . . . .	19
6.2.8	Editar Perfil . . . . .	20
<b>7</b>	<b>Conclusão e Trabalho Futuro</b>	<b>21</b>

# Capítulo 1

## Introdução

O desenvolvimento do tema "Monitorização de Acesso a Edifícios" insere-se na procura de conceptualizar e implementar sistemas de monitorização de ambientes tirando partido do uso de sensores físicos e/ou virtuais com um maior foco na Internet of Things e/ou Smart Cities.

Este projeto consistiu em, fazendo uso de sensores físicos e/ou virtuais, procurar e solucionar problemas relevantes relacionados com o mundo atual que nos rodeia, de forma eficiente e eficaz. Problemas como a gestão populacional, o controlo do tráfego rodoviário, o controlo dos parâmetros ambientais, entre outros, eram alguns dos temas passíveis de serem explorados.

A nossa equipa decidiu, com base num dos temas sugeridos no enunciado, a "Monitorização de Acesso a Edifícios", realizar o controlo populacional do estabelecimento de eleição da mesma, de forma a podermos perceber quando o mesmo se encontrava completamente cheio ou não. Esta ideia surgiu já no nosso segundo ano e desde então que procurámos os meios para desenvolver a mesma. Com as ferramentas de Sensorização e Ambiente pudemos finalmente concretizar o projeto e apresentá-lo de forma concreta com o desenvolvimento de uma aplicação completa.

Desta forma, o projeto desenvolvido apresenta as seguintes funcionalidades: contabilização do número de pessoas no interior do estabelecimento, sistema de registo e login, chat para interação entre os diferentes utilizadores, perfil editável.

Assim sendo, e tendo em consideração todos os tópicos acima abordados, este relatório visa ajudar a compreender e explicar todos os raciocínios e tomadas de decisão feitas de forma a conseguir realizar todos os objetivos para este projeto e, assim, atingir o resultado final desejado.

# Capítulo 2

## Estrutura do Relatório

Este relatório inicia-se no capítulo 1 onde é feita uma contextualização e enquadramento do tema em estudo e uma explicação do problema a ser tratado e os pressupostos a si inerentes.

Segue-se do capítulo 3 onde é feita uma análise dos objetivos estabelecidos pela equipa para o desenvolvimento do projeto apresentado.

Posteriormente, no capítulo 4 será explicada quais os sensores utilizados para o efeito, bem como o porquê da sua utilização para o presente projeto.

Em seguida, no capítulo 5 serão apresentados e explorados os dados recolhidos desde o início do momento da recolha e serão ainda feitas algumas conclusões sobre os mesmos, bem como apresentados alguns pormenores e detalhes que consideramos relevantes.

Depois, no capítulo 6 será apresentado toda a aplicação desenvolvida, desde a sua arquitetura até ao seu funcionamento, e ainda serão debatidas algumas ideias que a equipa considera que poderia alterar e/ou implementar no mesmo.

Finalmente o relatório termina com o capítulo 7 onde é feita uma síntese do documento, uma análise crítica generalizada do trabalho e dos resultados obtidos e uma reflexão sobre o trabalho futuro.

# Capítulo 3

## Objetivos

Aquando da apresentação do enunciado do projeto e posterior explicação, a equipa imediatamente lembrou a ideia que havia surgido entre si no segundo ano por mera diversão. No entanto fizemos por procurar mais alguns temas que pudessem ser interessantes de trabalhar desde a conexão entre os movimentos do rato e velocidade de escrita no teclado com os sentimentos da pessoa, aplicações que permitissem a promoção de negócios através de geofences, entre outros.

Após alguma discussão com o docente chegámos à conclusão de que algumas das ideias que haviam surgido seriam já realizadas por outros grupos e que deveríamos fazer algo diferente, existindo também outras que a própria equipa descartou por não ver ser realizada com sucesso, e sem ser comprometida, a recolha dos dados necessários. Assim ficou concordado entre a equipa e o docente que, através da utilização de sensores físicos iríamos monitorizar um estabelecimento a nível populacional, permitindo a qualquer pessoa obter essa informação através de uma aplicação.

Para além disto ficou estabelecido que, através do uso de modelos de Machine Learning e com base nos dados que iriam ser recolhidos, deveríamos ser capazes de prever o número de pessoas no estabelecimento num determinado dia e determinada hora.

# Capítulo 4

## Sensores Utilizados

Antes de proceder à explicação do sensor utilizado, a equipa considera importante refletir sobre os diferentes tipos de sensores estudados durante as aulas da unidade curricular e perceber o porquê de estes não serem os mais adequados à concretização do projeto planeado e objetivos a serem cumpridos.

Foram estudados dois tipos de sensores: soft (ou virtual) e físico. O sensor virtual, ou soft sensor, é uma combinação de vários processos de recolha de informação e modelos que utilizam essa mesma informação é um software onde várias métricas são processadas de forma conjunta. Já os sensores físicos são dispositivos que medem métricas físicas, como a temperatura, e a convertem num sinal a ser lido por um observador ou um objeto.

No caso do nosso projeto nós apenas precisamos de saber a quantidade de pessoas que se encontram dentro do estabelecimento, uma medida física, logo não é adequado usar sensores virtuais/soft. Nesta medida, para desenvolver o nosso projeto fizemos uso de uma placa Arduino ESP-8266 genérica.

Para concretizar o projeto seria necessário programar a placa para o efeito pretendido. No entanto, fizemos uso, uma vez que se adequa ao nosso trabalho, de um programa já desenvolvido de *CrowdSensing* que foi estudado durante as aulas. Assim, tivemos apenas que alterar as variáveis de acesso à rede à qual a placa estaria ligada e as credenciais de acesso à base de dados no Firebase.

De forma sucinta, aquilo que a placa Arduino se encontra programada para realizar é capturar os endereços MAC de todos os dispositivos que enviem um sinal de probing ao ponto de acesso que se cria na placa Arduino. Posteriormente, agregado com a data, timestamp que recebeu o sinal e distância à placa, envia, periodicamente ou quando o buffer se encontra cheio, a informação que recolheu para a base de dados.

No entanto, durante o projeto deparamo-nos com alguns problemas que não conseguimos resolver nem perceber o porquê de estar a acontecer. A partir do momento em que o número de pessoas simultâneas dentro do estabelecimento capturas pela placa ultrapassava as 21, numa estimativa, deixava de capturar e enviar informação para a base de dados, sendo necessário forçar o reinício físico da mesma, desligando e voltando a ligar à fonte de energia. Isto reflete-se nos resultados que serão apresentados posteriormente, bem como no correto funcionamento da aplicação desenvolvida.

# Capítulo 5

## Dados Recolhidos

Todos os dados recolhidos pela placa Arduino foram enviados para uma *Realtime Database*, no Firebase, sendo armazenado em cada envio: os probes (dispositivos detetados), a placa da qual veio o envio, o timestamp do envio e, para cada dispositivo detetado, o endereço MAC, a data e hora em que foi detetado e a distância a que se encontrava da placa.

Além destes foram recolhidos outros dados aquando do registo dos utilizadores no sistema, para poderem usufruir da aplicação, sendo eles: o nome completo do utilizador, o nome que será posteriormente apresentado no chat desenvolvido (*display name*), o email e o curso de cada um.

### 5.1 Exploração

Para o projeto desenvolvido pela nossa equipa, a nível da aplicação apenas trabalhamos com o número total de dispositivos detetados pela placa no último envio de informação, com o *display* e com o nome completo do utilizador.

Já a nível do modelo de machine learning trabalhamos com a informação armazenada relativa a cada dispositivo, de cada entrada na base de dados, enviada pela placa Arduino.

### 5.2 Tratamento de Dados

Posteriormente à recolha de dados feita através da placa esp8266 e do seu armazenamento, necessitámos de tratar os mesmos de forma a ter informação útil para no final fornecer ao nosso modelo de Machine Learning. Em seguida enunciam-se os passos necessários para que a equipa conseguisse extrair os dados da *Realtime Database* e os transformasse em informação de qualidade para os modelos:

- **Transferir o ficheiro json** com todos os dados recolhidos, que contém várias entradas do tipo:

```
1  "-N-X88N1JPTCqD9xy6_R": {  
2    "deviceId": "GRUP03_ID1",  
3    "probes": [  
4      {  
5        "date": "Wed Apr 13 09:57:19 2022\n",  
6        "mac": "6c:a6:04:8c:a7:16",
```



```

7     "previousMillisDetected": 11454,
8     "rssi": "-56"
9   },
10  {
11    "date": "Wed Apr 13 09:57:19 2022\n",
12    "mac": "dc:9b:d6:4e:c8:e0",
13    "previousMillisDetected": 28992,
14    "rssi": "-79"
15  }
16 ],
17 "timestamp": 1649840264753,
18 "type": "WIFI"
19 },
20 ...
21

```

- **Desenvolver um script** em python com o objetivo de transformar o ficheiro json num ficheiro csv em que cada linha contém informação sobre um único probe. As colunas representam atributos como o dia, a hora, o mac address e a data no formato aa-mm-dd;
- **Importar o dataset** para um notebook, com o objetivo de aplicar algumas técnicas de Machine Learning:

	mac_address	date	day	hour
0	6c:a6:04:8c:a7:16	2022-4-13	Wed	9
1	dc:9b:d6:4e:c8:e0	2022-4-13	Wed	9
2	6c:a6:04:8c:a7:16	2022-4-13	Wed	10
3	6c:a6:04:8c:a7:16	2022-4-13	Wed	10
4	22:a5:89:b3:9e:fb	2022-4-13	Wed	10
...	...	...	...	...
143577	a0:51:0b:4d:3e:ab	2022-5-3	Tue	10
143578	fe:c9:62:c2:d5:ad	2022-5-3	Tue	10
143579	4e:ac:e3:4b:70:5b	2022-5-3	Tue	10
143580	34:de:1a:98:af:8a	2022-5-3	Tue	10
143581	4e:ad:a6:7d:87:b4	2022-5-3	Tue	10

143582 rows × 4 columns

Figura 5.1: Dataset inicial

- **Eliminar** os casos em que o mesmo mac address surge mais do que uma vez na mesma hora, durante o mesmo dia;
- **Aplicar técnicas de *Feature Engineering*** de modo a adicionar mais informação relevante para o problema. Em primeiro lugar adicionamos a informação de quantos mac addresses diferentes surgiam em cada hora do dia, e decidimos que essa coluna seria aquilo que o nosso

modelo deveria prever. Acabamos por substituir essa coluna pela coluna do mac address. Em segundo lugar, decidimos adicionar mais informação sobre o tempo, separando a data em número do dia, dia da semana, mês, parte do dia e hora. Por fim adicionamos uma coluna com informação binária, indicando se o dia é feriado ou não (incluímos os seguintes feriados: sexta feria santa, páscoa, 25 de abril e dia do trabalhador):

	mac_count	day	hour	month	day_number	day_part	is_holiday_eve
0	2.0	6	9	4	13	2	0
2	28.0	6	10	4	13	2	0
30	51.0	6	11	4	13	2	0
81	64.0	6	12	4	13	0	0
145	233.0	6	13	4	13	0	0
...	...	...	...	...	...	...	...
114982	4.0	5	6	5	3	1	0
114986	5.0	5	7	5	3	1	0
114991	22.0	5	8	5	3	1	0
115013	134.0	5	9	5	3	2	0
115147	119.0	5	10	5	3	2	0

463 rows × 7 columns

Figura 5.2: Dataset final

## 5.3 Visualização dos Dados Transformados

Antes de avançar para a elaboração dos modelos de Machine Learning, decidimos visualizar uma descrição do número de mac addresses por cada hora do dia, ao longo dos diferentes dias. Adicionalmente comparámos a variação desses valores de acordo com a hora do dia, no geral, com a parte do dia e com o mês. Em seguida são apresentados vários resultados obtidos nesta fase de visualização:

```
dataset['mac_count'].describe()
```

count	463.000000
mean	248.954651
std	292.710541
min	1.000000
25%	8.000000
50%	83.000000
75%	460.000000
max	1107.000000
Name: mac_count, dtype: float64	

```
dataset.loc[dataset['mac_count'] == 1107]
```

mac_count	day	hour	month	day_number	day_part	is_holiday_eve
94314	1107.0	Fri	16	4	29 afternoon	0

Figura 5.3: Descrição da coluna mac\_count

Como se pode verificar na figura 5.3, o dia com mais mac addresses diferentes registados no café

Gota D'Água foi numa sexta feira, no dia 29 de abril, entre as 16h e as 17h. Também se pode concluir que a média de pessoas / mac addresses diferentes que passam no café, numa hora, é de aproximadamente 248.

```
dataset.groupby(['hour'])['mac_count'].mean()
hour
0    482.352936
1    464.200012
2     41.549999
3    10.700000
4     8.550000
5     5.200000
6     6.750000
7     8.700000
8    22.150000
9    56.285713
10   120.190475
11   230.300003
12   191.600006
13   253.750000
14   328.421051
15   348.421051
16   405.222229
17   469.277771
18   489.277771
19   485.166656
20   403.555542
21   396.333344
22   543.947388
23   400.894745
Name: mac_count, dtype: float32
```

Figura 5.4: Número médio de mac addresses por hora, durante o período de recolha de dados

Na figura 5.4 conseguimos inferir que ao longo de todos os dias em que houve recolha de dados, as horas em que o café era mais frequentado correspondiam principalmente às horas da noite.

```
dataset.groupby(['day'])['mac_count'].mean()
day
Fri    334.625000
Mon    237.263885
Sat    162.968750
Sun     65.492958
Thu    285.347839
Tue    288.867920
Wed    387.274200
Name: mac_count, dtype: float32

dataset.groupby(['day_part'])['mac_count'].mean()
day_part
afternoon    366.986664
dawn         110.485878
morning      134.064514
night        437.162170
Name: mac_count, dtype: float32
```

Figura 5.5: Média de mac addresses captados nos diferentes dias da semana, e nas diferentes partes do dia

Como era de esperar, em média existem mais registos de mac addresses na quarta feira, à noite, como se pode ver na figura 5.5. Estes valores podem justificar-se devido à noite académica. Para além destas informações podem ser encontrados alguns *plots* utilizados no *notebook*, que seguirá em anexo no ficheiro compactado relativo ao projeto.

## 5.4 Modelos de Machine Learning Utilizados

Com o conjunto de dados final, resultante das transformações enunciadas anteriormente, tratámos este problema como um **problema de regressão**, sendo que o nosso conjunto de atributos são todas as *features* que não sejam o `mac_count`, e o nosso *target* é o nosso `mac_count`. Desta forma, **pretendemos oferecer um modelo capaz de prever o número de `mac_addresses` que poderão ser captados num determinado dia, durante uma dada hora, no café Gota D'Água.**

De modo a solucionar este problema decidimos utilizar dois tipos de modelo: **DecisionTree Regressor** e **Rede Neuronal Artificial**.

### 5.4.1 DecisionTree Regressor

Para este modelo, dividimos o conjunto total de dados (463 casos), em 80% para treino (370), e 20% para teste (93).

- **Na tentativa 1** foi criada uma instância deste modelo, com `random_state=2022`, e como função de custo o *absolute error*.

```
1 regressor = DecisionTreeRegressor(random_state=2022, criterion='
  absolute_error')
2 regressor_fit = regressor.fit(X_train,y_train)
3
```

- **Na tentativa 2** utilizámos o mesmo modelo mas com *cross validation* com 5 *folds*:

```
1 scores = cross_val_score(regressor_fit, X_train, y_train, cv = 5)
2 print("mean cross validation score: {}".format(np.mean(scores)))
3 print("score without cv: {}".format(regressor_fit.score(X_train, y_train
4 )))
```

Score obtido sem cross-validation: 1.0;

Score médio obtido com cross-validation: 0.34822.

- **Na tentativa 3** foi utilizada a técnica de GridSearchCV, utilizando o modelo DecisionTree Regressor. Foram utilizados 5 folds, e o hiperparâmetro a otimizar foi o *min\_samples\_split*, a variar entre 2 e 10, que corresponde ao número mínimo de amostras que cada nó interno da árvore deve ter para ser dividido. Como *scoring* utilizamos o `r2_score` e no final escolhemos o melhor resultado.

```
1 scoring = make_scorer(r2_score)
2 g_cv = GridSearchCV(DecisionTreeRegressor(random_state=2022),
3
4 param_grid={'min_samples_split': range(2, 10)},
5 scoring=scoring, cv=5, refit=True)
6
7
8 g_cv.fit(X_train, y_train)
9 g_cv.best_params_
```

```

10
11
12     result = g_cv.cv_results_
13     # print(result)
14     r2_score(y_test, g_cv.best_estimator_.predict(X_test))
15

```

Score obtido: 0.466

## 5.4.2 Rede Neuronal Artificial

Depois de algumas tentativas percebemos que este modelo não funcionou nada bem com este conjunto de dados, e não foi selecionado como o modelo final. No entanto apresentámos as técnicas utilizadas:

- Conversão de todos os atributos para float.
- Standardização dos dados.
- Divisão dos dados em treino e teste (20% para teste).
- Divisão dos dados de treino resultantes em treino e validação (10% para validação).
- Modelo utilizado:

```

1     inputs = tf.keras.Input(shape=X_train.shape[1:])
2
3     x = tf.keras.layers.Dense(32, activation='relu')(inputs)
4     x = tf.keras.layers.Dense(16, activation='relu')(x)
5     x = tf.keras.layers.Dense(8, activation='relu')(x)
6     outputs = tf.keras.layers.Dense(1, activation='linear')(x)
7
8     model = tf.keras.Model(inputs=inputs, outputs=outputs)
9

```

- Utilização do otimizador adam, métrica mae e função de custo mae. Utilização de cross\_validation com 3 folds, 50 épocas e batch\_size 16.
- MAE obtido: Cerca de 140 mas o modelo apresentava overfitting.

### 5.4.3 Modelo final escolhido e resultados

Tendo em conta a descrição dos modelos anteriores, escolhemos como modelo final o modelo da tentativa 3 do DecisionTree Regressor. Em seguida segue a comparação entre alguns casos relativos aos mac\_counts reais, e os previstos:

```
Previsto: 990.5 | Real: 265.0 Diferença entre o real e o previsto: 725.5
Previsto: 360.8 | Real: 872.0 Diferença entre o real e o previsto: 511.2
Previsto: 9.0 | Real: 12.0 Diferença entre o real e o previsto: 3.0
Previsto: 539.0 | Real: 608.0 Diferença entre o real e o previsto: 69.0
Previsto: 5.0 | Real: 2.0 Diferença entre o real e o previsto: 3.0
Previsto: 573.8 | Real: 319.0 Diferença entre o real e o previsto: 254.79999999999995
Previsto: 431.2 | Real: 117.0 Diferença entre o real e o previsto: 314.2
Previsto: 539.0 | Real: 925.0 Diferença entre o real e o previsto: 386.0
Previsto: 8.5 | Real: 20.0 Diferença entre o real e o previsto: 11.5
Previsto: 20.0 | Real: 24.0 Diferença entre o real e o previsto: 4.0
Previsto: 806.0 | Real: 509.0 Diferença entre o real e o previsto: 297.0
Previsto: 4.5 | Real: 17.0 Diferença entre o real e o previsto: 12.5
Previsto: 65.5 | Real: 233.0 Diferença entre o real e o previsto: 167.5
Previsto: 71.0 | Real: 1.0 Diferença entre o real e o previsto: 70.0
Previsto: 332.0 | Real: 256.0 Diferença entre o real e o previsto: 76.0
Previsto: 38.25 | Real: 13.0 Diferença entre o real e o previsto: 25.25
Previsto: 1.6 | Real: 6.0 Diferença entre o real e o previsto: 4.4
```

Figura 5.6: Comparação entre os valores previstos e os reais

# Capítulo 6

## Sistema Desenvolvido

Aquando da ideia inicial para o desenvolvimento deste projeto, o pretendido era uma aplicação funcional ao ponto de mostrar com a maior das precisões quantas pessoas se encontravam no estabelecimento e onde se estavam sentadas. Apesar de ainda ser uma ambição da equipa poder elevar a mesma a esse patamar, rapidamente nos apercebemos que tal não seria tão fácil de concretizar.

Assim estabelecemos que queríamos cumprir com os objetivos a que nos propusemos inicialmente e, a partir daí, desenvolver funcionalidades que considerássemos interessantes de integrar na nossa aplicação, com o intuito de que a mesma fosse, de forma geral, utilizada por todos os que frequentam o mesmo estabelecimento que nós.

### 6.1 Arquitetura

Toda a informação que pode ser visualizada na aplicação encontra-se, como mencionado previamente, armazenada no Firebase, desde o número de dispositivos, até às mensagens do chat implementado.

A aplicação foi desenvolvida por completa no *Android Studio* e, por isso, vamos analisar cada uma das componentes de forma isolada de forma a melhor perceber o que acontece em cada uma delas.

Numa estrutura simplificada, a aplicação funciona da seguinte maneira:

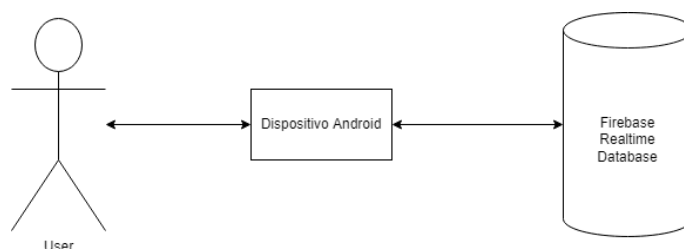


Figura 6.1: Arquitetura Simplificada da Aplicação

## 6.2 Funcionamento

### 6.2.1 Start Activity

Trata-se da primeira interface que aparece quando se inicializa a aplicação e conta com o título da aplicação e dois botões: um de registo e um de login.



Figura 6.2: Start Activity

Quando o utilizador interage com um dos botões é então redirecionado para a atividade associada e respetiva interface gráfica, podendo posteriormente continuar a interagir com a mesma.



### 6.2.2 Login

É na presente atividade que o utilizador, após se encontrar registado no sistema, tem acesso a todas as funcionalidades que a aplicação tem ao seu dispôr.

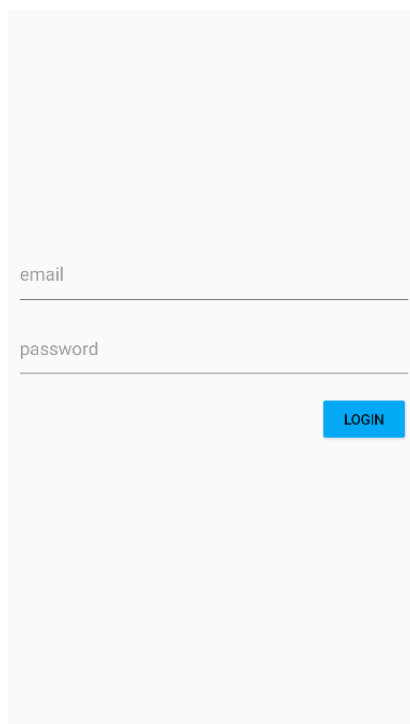
A login form with a light gray background. It features two input fields: the top one is labeled 'email' and the bottom one is labeled 'password'. Both labels are in a small, gray font and are positioned to the left of their respective input lines. To the right of the 'password' field is a blue rectangular button with the word 'LOGIN' in white, uppercase letters.

Figura 6.3: Login Activity

Após introduzir os respetivos valores nos campos associados, ao premir o botão "Login", o sistema vai procurar na autenticação do Firebase uma conta associada e confirmar os mesmos. Caso estes se encontrem válidos, o utilizador para a poder aceder às ferramentas existentes. Caso contrário recebe um aviso de credenciais inválidas e deve introduzir as mesmas novamente.

### 6.2.3 Registo

Já na atividade de registo, não muito diferente da de login, o utilizador insere as informações necessárias ao registo nos respetivos campos, nomeadamente: display name, nome completo, email, password e curso que frequenta.

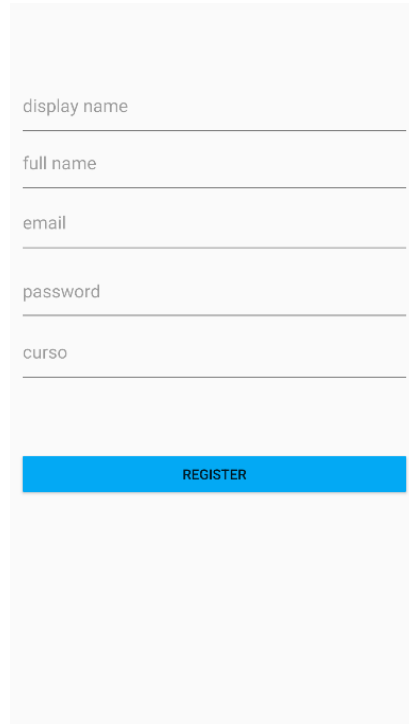
A screenshot of a mobile application's 'Register Activity' screen. It features a light gray background with a white card-like container. Inside the container, there are five text input fields stacked vertically, each with a placeholder label: 'display name', 'full name', 'email', 'password', and 'curso'. Below these fields is a solid blue rectangular button with the word 'REGISTER' in white, uppercase letters. The overall design is clean and modern.

Figura 6.4: Register Activity

As únicas verificações feitas durante o registo para estes campos é se não se encontram vazios e se a password contém mais de 6 carateres. Após isso, o serviço de autenticação do Firebase regista o email e password num serviço específico próprio e atribui um identificador ao utilizador.

Todas estas informações que o utilizador preenche são guardadas numa classe denominada *MyUser*, a qual é posteriormente guardada na base de dados para acesso posterior.

Cada utilizador é guardado no nodo "**users**" da base de dados com o respetivo identificador gerado automaticamente pelo Firebase.

#### Class MyUser

Nesta classe são guardados o email, nome completo, curso e nome de display aquando do registo, para posteriormente poder ser utilizado noutras atividades como o chat e o perfil do utilizador.

## 6.2.4 Main Activity

A Main Activity é a atividade principal e que serve de ponte para a maioria das outras atividades como o chat, perfil ou até mesmo para a Start Activity, ao fazer logout. Aqui está presente a contabilização do total de dispositivos presentes no estabelecimento aquando do último envio de informação, fazendo-se acompanhar de um botão "Update" que permite atualizar essa mesma contagem, caso a mesma não aconteça de forma automática.



Figura 6.5: Main Activity

Esta contagem resulta do acesso, na base de dados, ao último envio de informação por parte da placa Arduino e contabilização do número de filhos dentro do nodo "**probes**". Ao carregar no botão de update o processo acima mencionado é executado de forma a atualizar o valor.

Para além disso, faz-se acompanhar também de uma action bar, que contém a denominação do estabelecimento bem como um menu que permite aceder a outras atividades, e de uma mensagem de aviso sobre o valor apresentado ao utilizador.

## 6.2.5 Action Bar

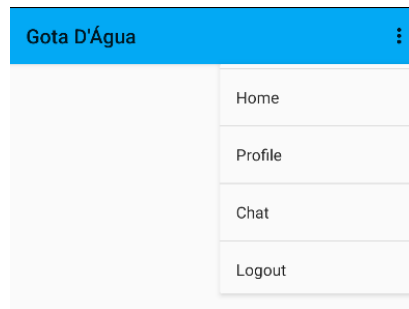


Figura 6.6: Action Bar

A action bar previamente mencionada permite a um utilizador navegar entre as diferentes funcionalidades disponíveis, contendo quatro botões: **Home**, que redireciona para a Main Activity, **Profile**, que redireciona para o perfil do utilizador autenticado, **Chat**, que redireciona para o chat implementado, e, por fim, **Logout**, que permite ao utilizador desautenticar-se e é redirecionado para a Start Activity.

## 6.2.6 Chat

A implementação de um chat foi uma ideia que surgiu à equipa uma vez que, junto do nosso grupo de amigos, é costume enviar mensagens a perguntar se alguém se encontra no estabelecimento. Assim, este serve para cumprir esse intuito, mas de forma a expandir a todos aqueles que frequentam o estabelecimento, deixando de ser restrito ao grupo habitual.

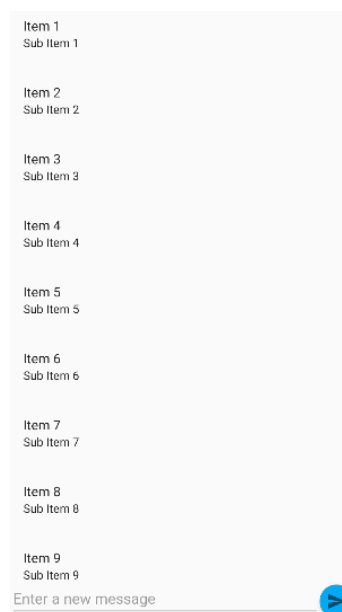


Figura 6.7: Chat

Nesta interface o utilizador pode visualizar mensagens antigas que estejam guardadas na base de dados, quem a enviou e quando, bem como introduzir texto que pode enviar ao carregar no botão cujo ícone é uma seta de envio.

Quando pressiona o botão de envio, é guardado num objeto *ChatMessage* as informações pertinentes para a posterior visualização, ou seja, o conteúdo da mensagem, quem a enviou, e o dia e hora a que enviou a mesma. Esta informação é posteriormente guardada no nodo "*messages*" na Firebase.

### Class ChatMessage

Esta classe guarda a informação de cada envio de mensagem, nomeadamente, o conteúdo da mensagem, quem a enviou e quando a enviou, sendo depois utilizada na leitura da informação da base de dados.

### 6.2.7 Perfil

Na atividade referente ao perfil, o utilizador pode visualizar o nome completo que introduziu bem como o nome que é apresentado aos outros utilizadores.



Figura 6.8: Profile Activity

Esta é acompanhada por um botão *Edit* que permite ao utilizador alterar este conteúdo. Ao interagir com o mesmo é redirecionado para uma atividade de edição de perfil.

### 6.2.8 Editar Perfil

Por último, a atividade de edição de perfil permite ao utilizador efetivamente alterar e gravar as alterações que pretender fazer, ou também cancelar caso assim o deseje, uma vez que esta apresenta, além de locais de introdução do novo conteúdo, dois botões ***Confirm***, para confirmar as alterações, e ***Cancel*** para cancelar as mesmas.

The image shows a mobile application screen for editing a profile. It features two text input fields. The first field is labeled 'Full Name' and the second is labeled 'Display Name'. Both labels are in bold black text. Below each label is a horizontal line representing the input field. At the bottom of the screen, there are two blue buttons with white text: 'CONFIRM' on the left and 'CANCEL' on the right.

Figura 6.9: Edit Profile Activity

Quando o utilizador interage com o botão de cancelar, é simplesmente redirecionado de volta para o seu perfil. No entanto, quando decide confirmar as alterações, através do seu identificador único da Firebase, é alterado no nodo previamente existente dentro dos **users** as novas informações.

## Capítulo 7

# Conclusão e Trabalho Futuro

O presente documento visa apresentar de forma sucinta e concreta o projeto desenvolvido, desde a formulação do problema até à implementação final e respetivos resultados apresentados, tendo sempre como base a explicação do raciocínio utilizado para a resolução do problema em questão.

Reverendo todo o trabalho feito até aqui, o projeto desenvolvido encontra-se completo e a equipa acredita que este está ao nível que pretendia desde o seu início. Todos os objetivos enunciados foram alcançados e realizados sempre da forma que se pensou ser a mais correta.

Como mencionado previamente, o trabalho inicialmente idealizado não era o resultado final, mas sim algo um pouco mais complexo a vários níveis: identificação de dispositivos, tratamento de dados, apresentação dos mesmos na aplicação.

A equipa considera que ganhando alguma prática com os tópicos abordados quer no projeto quer na Unidade Curricular, bem como em programação para dispositivos móveis e interação com, neste caso, a Firebase, poderia e pode alcançar aquilo que inicialmente havia planeado.

Contudo existem mais algumas implementações que gostaríamos de ter feito e ainda poderá vir a ser implementado. Uma vez que já possuímos um chat, tornar a aplicação numa espécie de rede social do estabelecimento seria, para nós, um projeto bastante interessante de desenvolver e concretizar. Com isto pretendíamos, além do chat já implementado, adicionar, por exemplo, um sistema de amigos e conversas privadas, tal como acontece no Facebook ou Instagram.

Já num contexto de recolha e exploração dos dados provenientes da placa Arduino, gostaríamos de, por exemplo, conseguir identificar cada utilizador através do seu endereço MAC e a partir daí disponibilizar na aplicação uma listagem das pessoas que efetivamente se encontram num estabelecimento.

Após alguma discussão e olhando para os resultados que obtivemos através do uso da placa Arduino, apesar de ser útil e cumprir os requisitos para aquilo que é o contexto do enunciado, acabámos por concluir que a utilização de geofences para uma contabilização em tempo real do número de pessoas tornaria a mesma muito mais exata, permitindo até expandir para o número de pessoas que está a ocupar, por exemplo, cada mesa disponível no estabelecimento.

Numa outra perspetiva de visualização de dados, a equipa não conseguiu mais do que aquilo que é apresentado no notebook que segue em conjunto com o relatório, não tendo sido capaz de utilizar, por exemplo, o Google Data Studio para, por exemplo, visualizar o número total de utilizadores em cada dia num gráfico de linhas, ou comparar as diferentes semanas e/ou dias da semana que foram alvos da recolha de dados.

No âmbito geral, o grupo sente que desenvolveu o trabalho prático da melhor forma possível e conseguiu alcançar todos os objetivos propostos no mesmo.