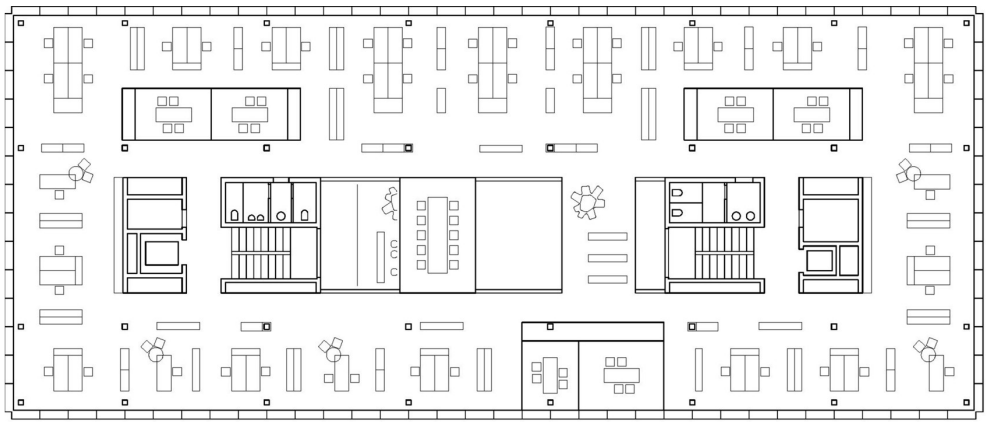
ISEL – Instituto Superior de Engenharia de Lisboa
ADEETC – Área Departamental de Engenharia de Electrónica e Telecomunicações e de Computadores

LEIM
Licenciatura em Engenharia Informática e Multimédia
Unidade Curricular de Projeto

SpaceManager – Módulo App



Ana Rita Venâncio Alves (42360)

Orientador

Professor Doutor Carlos Gonçalves

Julho, 2020

Resumo

O trabalho apresentado visa o desenvolvimento da componente de gestão de reservas de postos de trabalho e disponibiliza uma aplicação para o sistema *Android*, na qual o utilizador tem a possibilidade de reservar um posto de trabalho. As opções de reservas são, reservar um lugar aleatório, reservar um posto de trabalho através do mapa da sala, reservar um posto de trabalho aleatório com uma determinada característica ou reservar vários postos de trabalho para um grupo. Também é possível prolongar uma reserva que esteja a decorrer. A aplicação permite efetuar pesquisas para saber se um determinado colaborador da empresa tem uma reserva ativa.

Abstract

The work presented show the development of the workplace reservation management component and provides an Android system aplication in which the user has the possibility to reserve a workspace. The reservation options are, reserve a random workspace, reserve a workspace from the room map, reserve a random workspace with a specific feature or reserve multiple workspaces for a group. It is also possible to extend an ongoing reservation. The aplication allows to search if a particular employee has a reservation happening at the moment.

Agradecimentos

Em primeiro lugar, quero agradecer ao meu orientador, Prof.ª Doutor Carlos Gonçalves, pela sugestão do projeto que aceitei de imediato, pela disponibilidade durante a realização deste trabalho ao esclarecer as minhas dúvidas e apoio às minhas sugestões.

Aos meus pais, Augusto e Fátima, quero agradecer pelo apoio, paciência e compreensão quer durante todo o semestre, quer durante todo o curso.

Ao Pedro, que, de forma indireta, contribuiu para a elaboração do projeto, pela paciência e apoio demonstrado durante estes cinco anos.

A todos um muito obrigado!

Índice

Lista de Figuras

Lista de Tabelas

Lista de Exemplos

Capítulo 1

Introdução

No contexto de empresas que tem funcionários que passam pouco tempo nas instalações, e onde os locais de trabalho estão organizados em *open space*, não se justifica que cada funcionário tenha um local de trabalho fixo. Neste cenário surgiu a ideia de desenvolver o projeto *SpaceManager* que permite: i) Gerir a reserva dos postos de trabalho; ii) Verificar através de um conjunto de sensores quais os postos de trabalho efetivamente ocupados; iii) Definir a organização dos espaços *open space*.

Na figura 1.1 estão representadas as três partes que constituem o projeto *SpaceManager*. O trabalho que irá ser abordado neste relatório é o desenvolvimento da componente de *software* que permite gerir as reservas dos vários postos de trabalho.

Figura 1.1: Relação entre as três componentes do projeto *SpaceManager*

Como podemos ver na figura 1.1 este projeto é constituido por três partes. A primeira é a componente de *software*, representado pelo telemóvel, que permite gerir as reservas, como foi referido anteriormente. A segunda [1], re-

presentado pelo sensor, é a componente *hardware* cujo objetivo é desenvolver um sistema que consiga detetar se o posto de trabalho está efetivamente ocupado ou não. A terceira parte [2], representado pelo edifício, tem o objetivo de implementar uma componente de *software*, a ser executada num *browser*, que permite definir um edifício a quantidade de pisos que existem num edifício, o formato de cada piso, os postos de trabalho que existem em cada piso e os sensores que existem em cada posto de trabalho.

1.1 Objetivo

O trabalho apresentado neste relatório corresponde à componente do projeto *SpaceManager* que visa o desenvolvimento da componente de gestão de reservas de postos de trabalho. O nome atribuído a este trabalho foi *SpaceManager – App* e disponibiliza uma aplicação para o sistema *Android*, na qual o utilizador tem a possibilidade de reservar um lugar. A aplicação permite reservar lugares de vários modos: i) Reservar um posto de trabalho aleatório; ii) Reservar um posto de trabalho através do mapa da sala; iii) Reservar um posto de trabalho aleatório com uma determinada característica, como por exemplo um posto de trabalho que seja perto de uma janela; iv) Reservar vários postos de trabalho para um grupo. A aplicação permite ainda a possibilidade de prolongar uma reserva que esteja a decorrer. Como funcionalidades extra da aplicação é possível efetuar pesquisas que permitem saber se um determinado colaborador da empresa está com uma reserva ativa.

1.2 Organização do Documento

Este documento além deste capítulo contém os seguintes capítulos. O capítulo 2 onde são apresentados trabalhos relacionados e as tecnologias utilizadas para o desenvolvimento do trabalho. Os casos de utilização, os requisitos funcionais e não funcionais e os modelos de dados, entidade-associação e relacional, são apresentados no capítulo 3. No capítulo 4 é apresentada a implementação do trabalho. O capítulo 5 apresenta os resultados dos questionários que foram realizados a diferentes utilizadores que testaram a aplicação. Este trabalho termina com o capítulo 6 onde são apresentadas as conclusões e o trabalho futuro.

O dossier de projeto relativo a este trabalho, incluindo código desenvolvido, está disponível em [3].

Capítulo 2

Estado da Arte

Neste capítulo apresentam-se soluções já existentes semelhantes à temática tratada neste trabalho, secção 2.1, e as tecnologias utilizadas no desenvolvimento deste trabalho, secção 2.2.

2.1 Aplicações

Existem várias soluções com funcionalidades semelhantes ao projeto *SpaceManager*. A Steelcase, uma empresa de venda de mobiliário tem uma solução proprietária para integrar com o mobiliário que é vendido por eles. O RoomWizard [4] é uma solução de reserva de salas de reunião. Foi projetado intencionalmente para mostrar informação sobre reservas, ajudar na localização e agendamento de espaços de reuniões. O *software* esta integrado no mobiliário e não permite integrar *software* de outros fabricantes.

A Sony apresenta uma solução para a gestão de postos de trabalho inteligentes. A proposta TEOS [5] é um conjunto completo de soluções de gestão de postos de trabalho. Esta solução permite controlar todos os dispositivos de uma sala, incluindo iluminação, ecrãs e projetores. Tem a possibilidade de poupar energia, ou seja, se uma sala não está a ser usada há algum tempo todo o equipamento é desligado. Disponibiliza funcionalidades para determinar se os equipamentos estão com problemas.

A Cisco apresenta uma solução para a gestão de salas de reunião. A Appspace [6] permite que o utilizador inicie a sessão e que escolha o tipo de reunião que pretende realizar, para isso, é necessário tirar uma fotografia ao utilizador para ficar registado no sistema e de seguida é impresso um cartão

de visitante. A aplicação permite saber o caminho de uma sala para outra, cada sala tem um ecrã com a informação da temperatura, qualidade do ar e de como se pode invocar outros serviços por voz. Quando o utilizador entra na sala onde vai ocorrer uma reunião precisa de efetuar *check-in*. Se existir uma emergência, a aplicação apresenta em todos os ecrãs do edifício instruções de como agir perante a situação.

2.2 Tecnologias

Para a implementação do trabalho apresentado neste relatório foi utilizado como ambiente de desenvolvimento o `Android Studio` [7]. Para o armazenamento dos dados e autenticação foi utilizada a *framework* `Firebase` [8], suportada pela Google, cuja a arquitetura simplificada se apresenta na figura 2.1 [9].

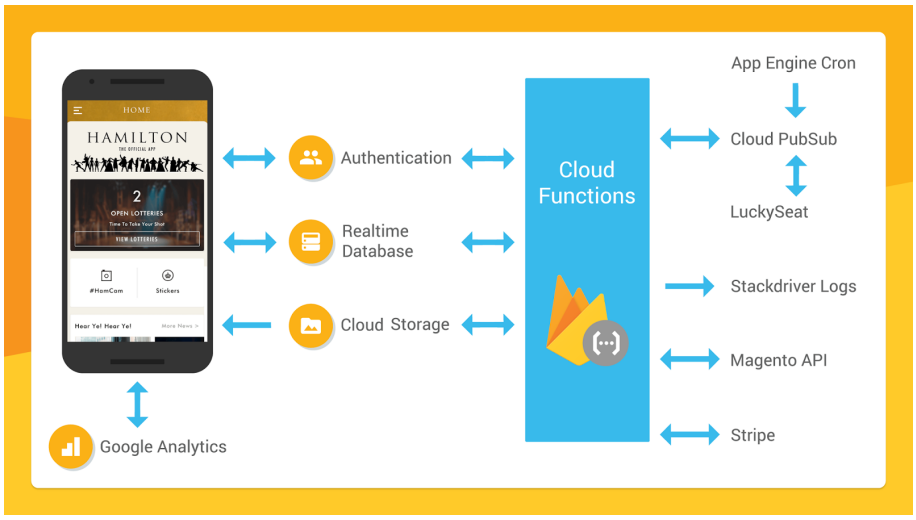


Figura 2.1: Arquitetura do `Firebase`

Esta *framework* foi utilizada pelo facto de ser a componente que auxilia o `Android Studio` na autenticação de utilizadores. Dado que o `Firebase` disponibiliza uma base de dados, neste trabalho, por uma questão de simplificação optou-se por utilizar essa base de dados. No entanto, a base de dados suportada pela *framework* `Firebase` é do tipo `NoSQL`, pelo que apenas permite um modelo de dados não relacional. Uma vez que as bases de dados `NoSQL` não garantem a integridade das relações dos dados, foi necessário

implementar a nível da aplicação a verificação e manutenção destas integridades. Na secção 3.4 apresenta-se com mais detalhe como esta verificação foi implementada.

De entre as funcionalidades disponibilizadas pela *framework* `Firebase`, neste trabalho apenas foram utilizados os mecanismos de autenticação e armazenamento de dados. Os mecanismos *Cloud Storage* e *Google Analytics*, apesar de não terem sido utilizados, seriam uma hipótese a considerar se fosse necessário, respetivamente, guardar um conjunto de recursos, tais como ficheiros `PDF` associados a uma reunião ou efetuar uma análise estatística sobre quais os postos de trabalho mais reservados ou as características que os colaboradores procuram mais quando reservam um posto de trabalho.

Capítulo 3

Modelo Proposto

Neste capítulo apresenta-se o modelo proposto para o trabalho. Na secção 3.2 são apresentados os casos de utilização. Na secção 3.1 apresentam-se os requisitos funcionais e não funcionais. O capítulo termina com a secção 3.3 onde é apresentada a arquitetura implementada.

3.1 Requisitos

Nesta secção são apresentados os requisitos funcionais e não funcionais. Um requisito é uma descrição das necessidades ou propósitos do produto. Um requisito funcional é a descrição de uma funcionalidade do sistema, enquanto que um requisito não funcional é a descrição de como é realizada uma funcionalidade do sistema. Na subsecção 3.1.1 são apresentados os requisitos funcionais e na subsecção 3.1.2 são apresentados os requisitos não funcionais.

3.1.1 Requisitos Funcionais

Os requisitos funcionais dividem-sem em três categorias: i) Evidentes – o utilizador tem que ter conhecimento da sua realização; ii) Invisíveis – não é possível ao utilizador visualizá-los; iii) Adornos – não afeta significativamente o custo ou outras funções. No contexto deste trabalho apenas foram considerados os requisitos funcionais evidentes e invisíveis. Os requisitos funcionais são apresentados na tabela 3.1.

Da análise da tabela 3.1 verifica-se que os requisitos funcionais que são realizados dentro do sistema são invisíveis, como é o caso do **Registar utilizador** e do **Iniciar sessão**. Os requisitos **Reservar lugar**, **Prolongar**

Tabela 3.1: Requisitos funcionais

Requisito	***Tipo***
1. Registar utilizador	Invisível
2. Iniciar sessão	Invisível
3. Reservar lugar	Evidente
4. Prolongar reserva	Evidente
5. Pesquisar pessoa	Evidente
6. Ver reservas	Evidente
7. Apagar reserva	Evidente

reserva, **Pesquisar pessoa**, **Ver reservas** e **Apagar reserva** como são funcionalidades que o utilizador vê, são classificados como requisitos funcionais evidentes.

3.1.2 Requisitos Não Funcionais

Os requisitos não funcionais são apresentados na tabela 3.2. Como foi referido anteriormente estes representam a forma como vão ser implementadas as funcionalidades descritas nos requisitos funcionais.

Tabela 3.2: Requisitos não funcionais

Requisito
Mecanismo de autenticação Google
Base de dados
Sensores infravermelhos

Como se apresenta na tabela 3.2 existem três requisitos não funcionais. O primeiro, **Mecanismo de autenticação Google**, é a capacidade que a aplicação tem de permitir a autenticação pelo Google através do Gmail. O segundo, **Base de dados**, é utilizar uma base de dados onde é guardada a informação sobre os utilizadores e os lugares. Por último, **Sensores infravermelhos**, são utilizados sensores infravermelhos para determinar se um lugar está ou não ocupado atualmente.

3.2 Casos de Utilização

Um caso de utilização é um tipo de classificador que representa uma unidade funcional do sistema entre um ou mais atores. Na figura 3.1 estão representados os casos de utilização considerados para a aplicação que se pretende desenvolver.

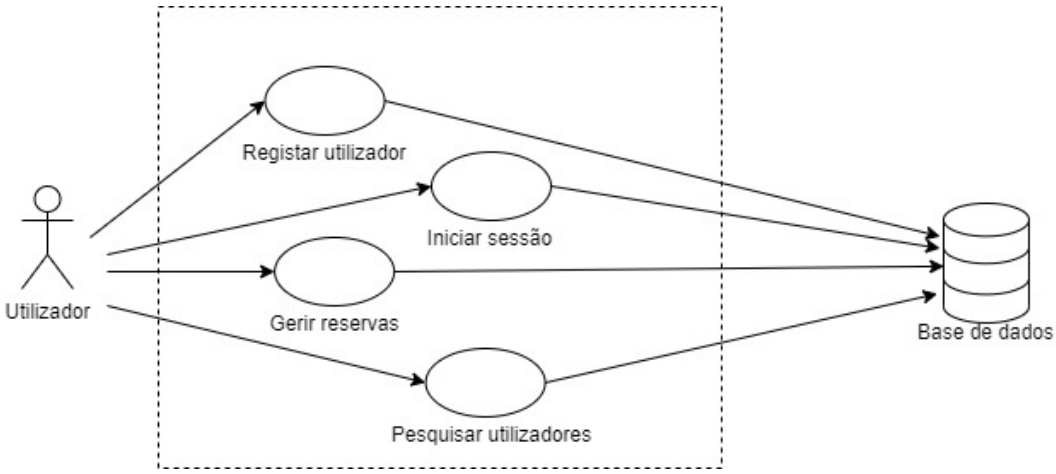


Figura 3.1: Casos de utilização

Como podemos aferir existem 4 casos de utilização. O utilizador tem a possibilidade de se **registar na aplicação**, no qual este fica guardado num base de dados. Posteriormente pode **iniciar a sessão**, em que a aplicação acede à base de dados para verificar as credenciais do utilizador e ver se as mesmas são válidas. Na **gestão das reservas** existem várias possibilidades: i) O utilizador pode fazer uma reserva, que fica guardada na base de dados; ii) Pode eliminá-la, e esta é removida da base de dados; iii) Pode prolongá-la e a reserva é alterada. Em relação à **pesquisa de utilizadores**, é possível consultar as reservas atuais de modo a perceber se existe uma reserva ativa para um colaborador que é especificado como argumento da pesquisa.

3.3 Modelo de Dados

O modelo entidade-associação é um modelo de dados utilizado para descrever as entidades e relações entre as mesmas. Na figura 3.2 está representado o modelo entidade-associação desenvolvido no contexto deste trabalho.

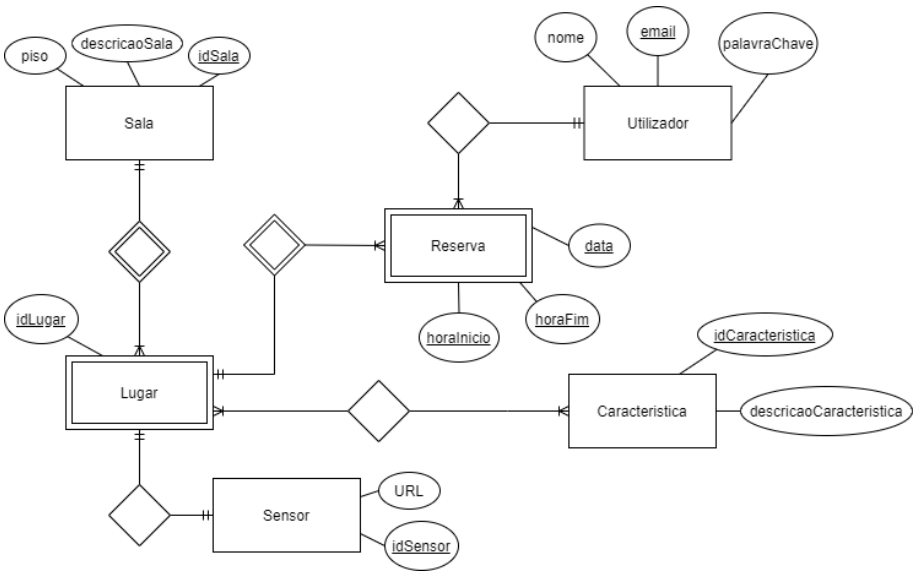


Figura 3.2: Modelo entidade-associação

Como podemos ver no modelo na figura 3.2 a entidade `Sala` é caracterizada por ter um identificador (`idSala`), uma descrição e um piso. Um `Lugar` está sempre associado a uma `Sala` e também tem um identificador (`idLugar`). Um `Lugar` está associado a um `Sensor` que é caraterizado por um identificador (`idSensor`) e por um `URL`. A `Caracteristica` está associada ao `Lugar` e tem como atributos o identificador `idCaracteristica` e a descrição. A entidade `Utilizador` define o utilizador da aplicação, é caracterizado pelo *e-mail*, nome e palavra-chave. Por fim, a entidade `Reserva`, depende de um `Lugar` e está associada a um `Utilizador` e tem os atributos que permitem caracterizar a data/hora da reserva. Do modelo entidade-associação apresentado na figura 3.2 obtém-se o modelo relacional apresentado na listagem 3.1, que permite descrever as relações entre as diferentes entidades.

Listagem 3.1: Modelo Relacional

```
1 SALA( idSala, piso, descricao )
2 CK = { idSala }
3
4 LUGAR( idLugar, idSala )
5 CK = { idLugar }
6 FK = { idSala } em SALA
7
8 CARACTERISTICA( idCaracteristica, descricao )
```

```
9  CK = { idCaracteristica }
10
11 LUGAR_CARACTERISTICA( idLugar, idSala,
       ↪idCaracteristica )
12 CK = { idLugar, idSala, idCaracteristica }
13 FK = { idLugar, idSala } em LUGAR
14
15 UTILIZADOR( email, nome, palavraChave )
16 CK = { email }
17
18 RESERVA( data, horaInicio, horaFim, idLugar, idSala,
       ↪ email )
19 CK = { data, horaInicio, horaFim, idLugar, idSala }
20 FK1 = { idLugar, idSala } em LUGAR
21 FK2 = { email } em UTILIZADOR not null
22
23 LUGAR_SENSOR( idLugar, idSala, idSensor )
24 CK = { idLugar, idSala, idSensor }
25 FK1 = { idLugar, idSala } em LUGAR
26 FK2 = { idSensor} em SENSOR
27
28 SENSOR( idSensor, URL )
29 CK = { idSensor }
```

No modelo relacional apresentado na listagem 3.1 é possível observar duas relações que não estão diretamente representadas no modelo entidade-associação, como é o caso de `LUGAR_CARACTERISTICA` e `LUGAR_SENSOR`. Estas têm o propósito de suportar a relação entre o lugar com as características e o lugar com os sensores.

3.4 Implementação do Modelo Relacional numa Base de Dados NoSQL

Do modelo entidade-associação apresentado na figura 3.2 foi derivado o modelo relacional da listagem 3.1. Como foi referido na secção 2.2 o tipo da base de dados utilizada é `NoSQL`, pelo que o modelo relacional apresentado na listagem 3.1 teve de ser implementado garantindo a integridade das relações. Foi criada uma `Database Reference` por cada tabela do modelo relacional.

Também foi criada uma `Database Reference` para simular o estado de cada sensor, ou seja, determinar se o posto de trabalho está ou não ocupado.

Houve a necessidade de representar cada identificador através de um código único. Esse código é do tipo `MD5` e cria um identificador de 32 bytes e pode ser derivado de um ou mais valores. No capítulo 4 esta explicação será complementada.

Capítulo 4

Implementação

Neste capítulo apresenta-se a implementação da aplicação que foi desenvolvida para *Android*. Na secção 4.1 apresenta-se a arquitetura utilizada. Na secção 4.2 são apresentados os métodos de autenticação disponibilizados pela aplicação. Os diversos tipos de reserva são apresentados na secção 4.3. A funcionalidade de prolongar uma reserva é apresentada na secção 4.4. Na secção 4.5 é apresentada a funcionalidade que permite visualizar as reservas futuras. A pesquisa por pessoa é apresentada na secção 4.6. A integração com o projeto *SpaceManager* – Módulo Sensores é apresentada na secção 4.7.

4.1 Arquitetura

Na figura 4.1 é apresentada a arquitetura do sistema.

Figura 4.1: Arquitetura do sistema

Como podemos observar da figura 4.1 o sistema é constituído por quatro componentes. Na figura 4.2 está apresentada a atividade inicial da aplicação.

Figura 4.2: Autenticação

Como podemos observar, o utilizador pode registar-se na aplicação e de seguida iniciar a sessão ou tem a possibilidade de iniciar a sessão pelo Google. Em qualquer dos casos a base de dados é suportada pelo `Firebase`.

4.2 Autenticação

O utilizador tem a possibilidade de se autenticar através da aplicação. No entanto, antes de utilizar a aplicação, terá que efetuar o registo, na qual insere o nome, endereço de *e-mail* e palavra-chave. De seguida inicia a sessão utilizando o endereço de *e-mail* e palavra-chave. Em alternativa é possível efetuar a autenticação através do endereço de *e-mail* do Google, o Gmail.

Quando o utilizador efetua o registo na aplicação, os seus dados são adicionados à base de dados do `Firebase` na referência `utilizadores`, e onde cada utilizador fica associado a um código `MD5`. Este é utilizado para ser mais fácil organizar a informação dos utilizadores na base de dados. O código Java referente à criação do código `MD5` para o utilizador encontra-se na listagem 4.1.

Listagem 4.1: Criação do código MD5 para o utilizador

```
1 public void onDataChange(@NonNull DataSnapshot ds) {
2  String md5 = "";
3  try {
```

```
4     MessageDigest md=MessageDigest.getInstance("MD5");
5     md.update(sEmail.getBytes(),0,sEmail.length());
6     md5 = new BigInteger(1, md.digest()).toString(16);
7    }catch(NoSuchAlgorithmException e){
8     System.err.println("Erro ao gerar o codigo MD5");
9    }
10
11   DatabaseReference user = users.child(md5);
12   DatabaseReference nome = user.child("nome");
13   nome.setValue(sNome);
14   DatabaseReference email = user.child("email");
15   email.setValue(sEmail);
16   DatabaseReference pass = user.child("pass");
17   pass.setValue(sPass);
18  }
```

Neste caso o código `MD5` é derivado do *e-mail* e é utilizado como identificador único na referência utilizadores. Este mecanismo foi utilizado em todas as situações onde foi necessário suportar o modelo `SQL` numa base de dados `NoSQL`. Quando o utilizador inicia a sessão, a aplicação verifica se o endereço de *e-mail* introduzido corresponde à palavra-chave fornecida.

Para o inicio da sessão ser realizada através do endereço de *e-mail* do Google o aplicação faz um pedido ao Google utilizando o `Firebase` como intermediário. O Google retorna um código e se este corresponder ao valor 123 significa que a sessão foi iniciada com sucesso. Nesta situação utiliza-se o *e-mail* devolvido para efetuar as reservas.

Depois do utilizador ter a sessão iniciada vai encontrar a atividade apresentada na figura 4.3.

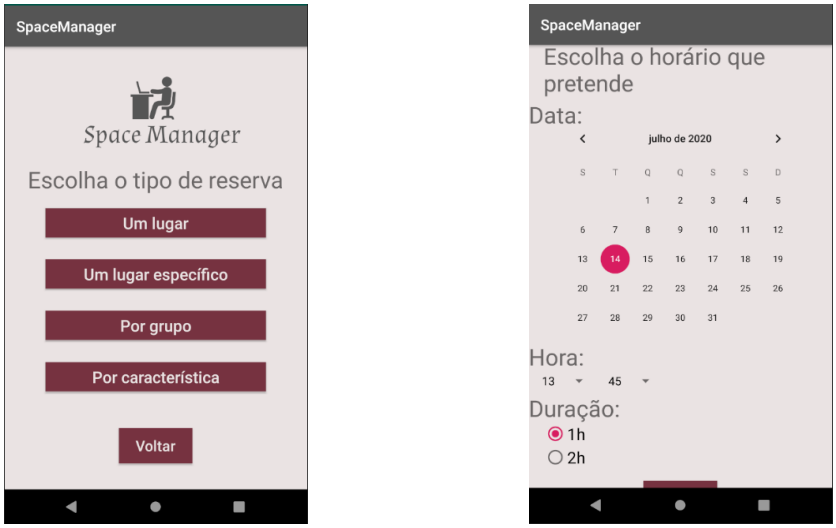
Figura 4.3: Funcionalidades da aplicação

Como podemos observar na figura, no canto superior direito aparece o nome do utilizador. Existem quatro funcionalidades que permitem respetivamente, efetuar reservas, efetuar pesquisas, verificar as reservas futuras e prolongar uma reserva que esteja a decorrer. Por fim, o utilizador tem a possibilidade de sair da sua sessão.

4.3 Reservas

Neste secção apresentam-se as funcionalidades relacionadas com as reservas. Na subsecção 4.3.1 é apresentada a implementação da reserva de um posto de trabalho aleatório. A implementação da reserva de um posto de trabalho escolhido do mapa da sala é apresentada na subsecção 4.3.2. A subsecção 4.3.3 apresenta a implementação da reserva de grupo. A implementação da reserva pela característica do posto de trabalho é apresentada na subsecção 4.3.4.

A figura 4.4a apresenta a atividade onde é escolhida o tipo de reserva.



(a) Tipos de reserva (b) Atividade da escolha do horário

Figura 4.4: Reserva de sala

Antes do utilizador efetuar qualquer reserva necessita de escolher o horário no formato data/hora. Na aplicação a seleção da data é efetuada com recurso a um `Widget` do tipo calendário (`CalendarView`). A escolha da hora é realizada com recurso a um `Container` do tipo `Spinner`. Por omissão a duração inicial de uma reserva poderá ser uma ou duas horas. Na figura 4.4b é apresentada a atividade no qual o utilizador escolhe o horário da reserva.

No contexto do `Android Studio` a forma de passar informação entre atividades é realizada através do envio de uma mensagem do tipo $<chave, valor>$ com a informação necessária. Sempre que se escolhe o tipo de reserva é enviada a mensagem correspondente para a atividade da escolha do horário (figura 4.4b). Na listagem 4.2 é apresentado o código Java que envia uma mensagem diferente por cada tipo de reserva.

Listagem 4.2: Mensagens do tipo da reserva

```
1 public void onClick(View v) {
2  switch (v.getId()) {
3   case R.id.BQualquer:
4    Intent al=new Intent(Reserva.this,Horario.class);
5    al.putExtra("tipo", "aleatorio");
6    Reserva.this.startActivity(al);
7    break;
8   case R.id.BLugar:
```

```
9     Intent mp=new Intent(Reserva.this,Horario.class);
10    mp.putExtra("tipo", "mapa");
11    Reserva.this.startActivity(mp);
12    break;
13   case R.id.BPessoas:
14    Intent re=new Intent(Reserva.this,Horario.class);
15    re.putExtra("tipo", "pessoas");
16    Reserva.this.startActivity(re);
17    break;
18   case R.id.BOpcao:
19    Intent op=new Intent(Reserva.this,Horario.class);
20    op.putExtra("tipo", "opcao");
21    Reserva.this.startActivity(op);
22    break;
23  }
24 }
```

4.3.1 Reserva de um Posto de Trabalho

O primeiro tipo de reserva que o utilizador pode realizar é a reserva de um posto de trabalho. Nesta reserva é atribuído ao utilizador um posto de trabalho aleatório dentro da possibilidade dos postos de trabalho disponíveis na data/hora pretendida. Na figura 4.5 é apresentada a atividade que mostra os detalhes sobre a reserva que foi efetuada.

Figura 4.5: Atividade que mostra a reserva efetuada

Estes detalhes incluem a data, a hora de início, a hora de fim, a sala e o posto de trabalho. A atividade apresentada na figura 4.5 é comum a todos os tipos de reserva. Depois do utilizador realizar cada uma das tarefas pedidas nos restantes tipos de reserva aparece esta atividade a mostrar os detalhes da reserva efetuada.

4.3.2 Reserva Através do Mapa da Sala

Este tipo de reserva é realizado sobre um mapa, em que o utilizador seleciona o posto de trabalho que pretende clicando sobre ele. Na versão atual da aplicação o mapa com os postos de trabalho foi implementado utilizando uma grelha de botões.

Se a reserva é efetuada hoje e a hora escolhida pelo utilizador for nos próximos 15 minutos, quando for escolher o posto que trabalho que pretende, os postos de trabalho que estiverem ocupados aparecem com a cor vermelha e não é possível selecioná-los. Se um posto de trabalho estiver reservado na data/hora pretendida pelo utilizador aparece com a cor azul e também não é possível selecioná-lo. Na figura 4.6 está apresentada a atividade no qual o utilizador efetua a reserva através do mapa da sala.

Figura 4.6: Atividade da reserva através do mapa da sala

Nesta figura podemos observar que o utilizador escolheu o posto de trabalho 12 (cor cinzento escuro), os postos de trabalho 1, 7 e 10 estão ocupados (cor vermelha), o posto de trabalho 6 está reservado (cor azul) e os restantes estão disponíveis para reserva (cor cinzento claro).

4.3.3 Reserva por Grupo

Na reserva por grupo o utilizador pode efetuar reservas até quatro pessoas, e é escolhido através de um conjunto de `RadioButtons`. Na figura 4.7 é apresentada a página onde o utilizador escolhe o número de pessoas que participam na reserva.

Figura 4.7: Atividade da reserva por grupo

Como podemos observar na figura, o utilizador escolheu a opção "4 pessoas". A aplicação vai mostrar o mapa da sala para serem escolhidos os postos de trabalho, equivalente à figura 4.6.

4.3.4 Reserva por Característica

Na reserva por característica, por exemplo um posto de trabalho com iluminação natural, o utilizador escolhe a característica que pretende que o posto de trabalho tenha. Nesta situação é lhe atribuído um posto de trabalho aleatório que esteja associado às características pretendidas. Na figura 4.8 é apresentada a página onde o utilizador escolhe a característica do posto de trabalho que quer reservar.

Figura 4.8: Atividade da reserva por característica

Caso não exista nenhum posto de trabalho com as características pretendidas é apresentada ao utilizador uma mensagem que indica que não é possível satisfazer o pedido.

4.4 Prolongar Reserva

O utilizador tem possibilidade de prolongar uma reserva que esteja a decorrer naquela momento. Pode prolongá-la por 15 minutos, 30 minutos ou 1 hora. A aplicação acede à base de dados para ver se existe alguma reserva a decorrer naquele momento e vai alterar a hora de fim da reserva na base de dados dependente da opção que foi escolhida. Na figura 4.9 é apresentada a atividade no qual o utilizador escolhe por quanto tempo quer prolongar a reserva.

Figura 4.9: Atividade para prolongar reserva

Como podemos observar na figura, o utilizador escolheu a opção "30 minutos". Quando o utilizador carrega em "Prolongar reserva"aparece uma mensagem `Toast` que mostra a que horas acaba a reserva depois de prolongada. Na figura 4.10 podemos ver a mensagem `Toast` que indica que a reserva foi prolongada para as 17:00h.

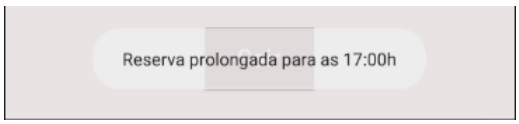


Figura 4.10: Mensagem `Toast`

4.5 As Minhas Reservas

O utilizador tem a possibilidade de poder ver as suas reservas futuras. A aplicação acede à base de dados e percorre todas as reservas e vê quais é que pertencem ao utilizador com a sessão iniciada. As reservas aparecem por ordem da mais recente para a mais futura. Existe igualmente a possibilidade de cancelar a reserva apresentada no ecrã. Na figura 4.11 está apresentada a atividade que mostra as reservas futuras.

Figura 4.11: Minhas reservas

Como podemos ver na figura, é mostrado o número total de reservas do utilizador e qual a reserva que está a ser visualizada no canto superior direito. Para cada reserva é apresentada informação sobre a mesma existindo a hipótese de a cancelar.

4.6 Pesquisa por Pessoa

A aplicação tem a funcionalidade de pesquisar um utilizador e ver se este tem uma reserva ativa no momento da pesquisa. A pesquisa tem como chave o *e-mail* do colaborador que pretende ser pesquisado e como resultado são apresentados os detalhes da reserva. Na figura 4.12 é apresentado um resultado exemplificativo de uma pesquisa.

Figura 4.12: Resultado da pesquisa

Como podemos observar na figura, a pessoa que foi pesquisada tem o *e-mail* "anarita@mail.com"e tem uma reserva ativa para o dia "2020/07/06"das $11h00$ às $12h00$ na sala 1 no posto de trabalho 4.

4.7 Integração com o Projeto *SpaceManager* – Módulo Sensores

Para ser feita a integração com o Projeto *SpaceManager* – Módulo Sensores é necessário efetuar a leitura de cada sensor de modo a determinar se um posto de trabalho está ou não ocupado (secção 4.3.2). Um sensor tem um `URL` que quando acedido envia uma mensagem a dizer se está alguém naquele posto de trabalho. Autonomamente o sensor consegue detetar se o posto de trabalho está ocupado e nessa situação ativa a luz vermelha (figura 4.13). Se o posto de trabalho não está ocupado, a luz associada ao sensor fica verde (figura 4.13).

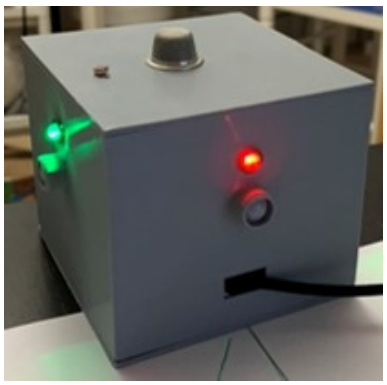


Figura 4.13: Luzes de presença do sensor

O sensor disponibiliza uma função que permite alterar a cor para azul no caso do posto de trabalho estar reservado.

A manipulação dos estado das luzes dos sensores é efetuada através do `URL`. Na figura 4.14 está representada a arquitetura revisitada do sistema já com a integração com o Projeto *SpaceManager* – Módulo Sensores.

Figura 4.14: Arquitetura

Capítulo 5

Testes de Usabilidade

Neste capítulo são apresentados os resultados do questionário de usabilidade baseado no modelo SUS (*System Usability Scale*). No apêndice A apresentam-se todas as perguntas realizadas aos utilizadores bem como os resultados obtidos. A figura 5.1 apresenta o universo alvo do questionário.

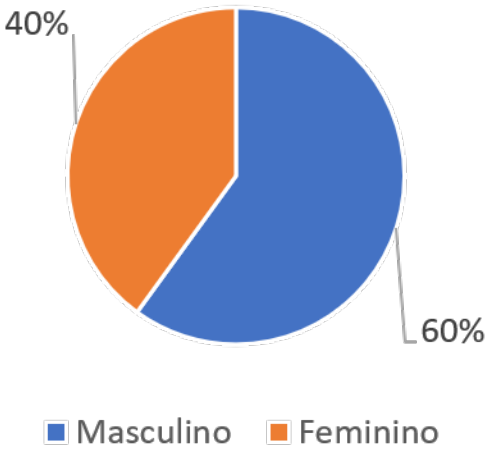


Figura 5.1: Universo do questionário *System Usability Scale*

O universo foi de 5 pessoas, 40% mulheres e 60% homens. Com idades entre os 22 e 63 anos. De acordo com a escala SUS as respostas são valores numéricos entre 1 (discordo totalmente) e 5 (concordo totalmente). A tabela 5.1 apresenta a média aritmética para cada uma das perguntas realizadas a cada um dos utilizadores.

Tabela 5.1: Resumo dos testes de usabilidade

Média	***Pergunta***
$4,4$	Usaria a aplicação com frequência
$1,2$	A aplicação é demasiado complexa
$4,2$	A aplicação é fácil de usar
1	Precisei de ajuda de alguém com conhecimentos técnicos para utilizar a aplicação
$4,4$	As várias funções da aplicação estão muito bem integradas
$1,6$	A aplicação apresenta muita inconsistência
$4,8$	A aplicação é de rápida aprendizagem
$1,4$	A aplicação é confusa
$4,6$	Senti-me confiante a utilizar a aplicação
$1,2$	Foi necessário aprender coisas novas para conseguir utilizar a aplicação

Da análise da tabela verifica-se, de um modo geral, que a aplicação é de utilização fácil e intuitiva e que não necessita de conhecimentos específicos para ser manuseada.

Capítulo 6

Conclusões e Trabalho Futuro

O objetivo inicial do trabalho apresentado neste relatório consistia no desenvolvimento de uma aplicação para o sistema *Android* que permitisse gerir as reservas de postos de trabalho em ambientes *open space*. As principais funcionalidades desta aplicação incluíam a possibilidade de efetuar reservas e pesquisar colaboradores.

O inicio de sessão da aplicação pode ser efetuada utilizando os dados próprios ou recorrendo aos serviços de autenticação disponibilizados pela Google. Em ambas as situações foi utilizado o *framework* `Firebase`. Relativamente ao mecanismo de reservas foram implementadas várias possibilidades: i) Reservar um posto de trabalho aleatório; ii) Reservar um posto de trabalho através do mapa da sala; iii) Reservar um posto de trabalho aleatório mas que tenha um conjunto de características; iv) Reservar vários postos de trabalho para um grupo. Existe ainda a possibilidade de prolongar uma reserva que esteja a decorrer e efetuar pesquisas que permitem saber se um determinado colaborador da empresa está ou não com uma reserva ativa. Neste sentido, a aplicação desenvolvida cumpre todos os requisitos inicialmente propostos.

Existem algumas funcionalidades que não foram implementadas, mas que poderiam contribuir para uma melhor experiência de utilização. Nesse sentido foram identificadas as seguintes funcionalidades:

⇒ Guardar a sessão, para que não fosse necessário inicia-la sempre que se entra na aplicação.

⇒ Permitir que o utilizador escolha o idioma utilizado na apresentação dos menus.

A utilização da sessão iria permitir melhorar a experiência do utilizador, na medida em que iria ser possível tirar partido, por exemplo, do mecanismo de notificações disponibilizado no sistema *Android*. Neste momento os menus da aplicação já se adaptam, embora que de forma automática, ao idioma do dispositivo. No entanto, pensamos que seria mais flexível dar a hipótese ao utilizador de escolher qual o idioma a ser utilizado nos menus da aplicação.

Para trabalho futuro será possível integrar as três componentes do projeto *SpaceManager* e ficar com uma sistema 100% funcional. No entanto, esta integração irá obrigar a ajustes pontuais nos três trabalhos, nomeadamente, fará sentido substituir a base de dados `NoSQL`, utilizada neste trabalho, por uma base de dados `SQL` comum as três trabalhos.

Futuras melhorias poderiam incidir no seguintes aspetos:

⇒ Melhorar as funcionalidades de pesquisa.

⇒ Tornar o processo de registo mais seguro.

Neste momento a chave de pesquisa apenas considera o *e-mail*, faz sentido considerar igualmente o nome do utilizador. O registo do utilizador de forma manual só deve ser considerado válido após um passo intermédio de validação. Com este opção consegue-se minimizar os registos efetuados por *bots*.

Bibliografia

[1] P. Marques, *SpaceManager – Sensores*. Intituto Superior de Engenharia de Lisboa, Projeto Final de Curso ed., 2020. Licenciatura em Engenharia Informática e Multimédia.

[2] B. Silva, *SpaceManager – Web Components*. Intituto Superior de Engenharia de Lisboa, Projeto Final de Curso ed., 2020. Licenciatura em Engenharia Informática e Multimédia.

[3] "SpaceManager – App," July 2020. https://github.com/ritaalves185/pfc.

[4] "Steelcase – RoomWizard," July 2020. https://www.steelcase.com/products/scheduling-systems/roomwizard/.

[5] "Sony – TEOS," July 2020. https://pro.sony/en_PW/solutions/corporate/solutions-teos-manage.

[6] "Cisco – Appspace," July 2020. https://www.appspace.com/partners/cisco/resources/.

[7] "Android Studio," June 2020. https://developer.android.com/studio/.

[8] "Firebase," June 2020. https://firebase.google.com/.

[9] "Arquitetura Firebase," June 2020. https://firebase.googleblog.com/2017/08/hamilton-app-takes-stage.html.

Apêndice A

Questionário da Usabilidade da Aplicação

De modo a testar a usabilidade da aplicação foi efetuado um questionário baseado no modelo perguntas `SUS` (*System Usability Scale*). No contexto deste trabalho foi construído um questionário com as perguntas que se apresentam de seguida.

1 Usaria a aplicação com frequência.

2 A aplicação é demasiado complexa.

3 A aplicação é fácil de usar.

4 Precisei de ajuda de alguém com conhecimentos técnicos para utilizar a aplicação.

5 As várias funções da aplicação estão muito bem integradas.

6 A aplicação apresenta muita inconsistência.

7 A aplicação é de rápida aprendizagem.

8 A aplicação é confusa.

9 Senti-me confiante a utilizar a aplicação.

10 Foi necessário aprender coisas novas para conseguir utilizar a aplicação.

O resultado do questionário encontra-se presente na seguinte tabela.

Tabela A.1: Resultados obtidos

	U1	***U2***	***U3***	***U4***	***U5***
Q1	5	4	4	4	5
Q2	1	1	1	2	1
Q3	5	5	3	4	4
Q4	1	1	1	1	1
Q5	5	5	4	3	5
Q6	1	1	3	2	1
Q7	5	5	5	5	4
Q8	1	1	1	2	2
Q9	5	4	4	5	5
Q10	1	1	2	1	1