



Trabalho Prático

Programação de Dispositivos Móveis e Multisensoriais

Jonas Andrade

a10506@alunos.ipca.pt

Ivo Gomes

a10700@alunos.ipca.pt

Hugo Gonçalves

a11600@alunos.ipca.pt

Tiago Oliveira

a21585@alunos.ipca.pt

Mestrado em Engenharia Informática

Escola Superior de Tecnologia

Instituto Politécnico do Cávado e do Ave

Ano lectivo 2020/2021

Conteúdo

1	Enquadramento e Introdução	6
2	Desenvolvimento	7
2.1	Ambiente de desenvolvimento	7
2.2	Descrição da aplicação	8
2.3	Arquitetura da solução	10
2.3.1	Serviços	11
2.3.2	Modelos	12
2.3.3	Controladores	13
2.3.4	Vistas	13
2.3.5	Utilitários	21
2.4	Metodologia de desenvolvimento	22
2.4.1	Versionamento	22
2.4.2	Planeamento	24
2.4.3	Boas práticas	26
2.5	Desafios de implementação	27
3	Conclusão	28
4	Bibliografia	29

Lista de Figuras

1	Logótipo da aplicação	8
2	Sitemap da aplicação	9
3	Arquitectura da solução	10
4	Estrutura do repositório de Blobs	12
5	Vista Splash	14
6	Vista Login	15
7	Vista Register	16
8	Vista Register (continuação)	16
9	Vista Recover	17
10	Vista About	18
11	Vista Profile	19
12	Vista Map	20
13	Exemplo de um <i>pull request</i>	23
14	Sprints do projecto	24
15	Backlog do projecto	25

Glossário

Android Sistema operativo, baseado no kernel Linux, que serve, principalmente, dispositivos móveis. 6, 10, 19, 27, 28

API Interface de programação de aplicações. 10, 19

Backlog Lista que reúne todas as funcionalidades desejadas para um determinado produto de software. 24

Blobs Binary Large Objects. Em português literal, objectos largos binários. 2, 11, 12

CRUD Create, Read, Update and Delete. Operações fundamentais de uma base de dados. 11

DRY Don't Repeat Yourself. Princípio de desenvolvimento de software que visa evitar redundância desnecessária. 26

Facebook Plataforma de mídia social, fundada em 2004, por Mark Zuckerberg. 11, 14

Firebase Plataforma, desenvolvida pela Google, que possibilita o desenvolvimento de aplicações móveis e web. 11, 27, 28

Framework Conjunto de princípios e técnicas, utilizadas para resolver um dado problema, inerente a um qualquer domínio. 24

Git Sistema de controlo de versões distribuído e de código aberto. Foi originalmente concebido para suportar o desenvolvimento do kernel Linux. 22

GitHub Plataforma orientada ao ciclo de vida de desenvolvimento de software.

Acolhe, nos dias de hoje, inúmeros projectos de software de código aberto. 22, 24

Google Empresa multinacional que se dedica ao desenvolvimento de variados produtos e serviços tecnológicos. 11, 14

Google Maps Serviço, desenvolvido pela Google, que visa a pesquisa e visualização de mapas e imagens de satélite. 19, 28

GPS Global Positioning System. Sistema de navegação por satélite, capaz de indicar a localização de um qualquer dispositivo electrónico compatível. 19

KISS Keep It Simple, Stupid. Princípio que defende que um sistema deve manter-se simples, evitando, por consequência, complexidade supérflua. 26

Kotlin Linguagem de programação tipada, multi-plataforma, orientada a objectos e funcional, de propósito geral, que corre sobre a máquina virtual Java. 10, 26, 27

Linter Ferramenta de análise estática de código fonte. É principalmente utilizada para encontrar erros e más práticas e garantir coerência sintáctica. 26

MVC Model-View-Controller. Padrão de desenvolvimento de software focado em dividir a lógica aplicacional em três elementos distintos: modelos, vistas e controladores. 10

Pinterest Rede social, focada na partilha de ideias, sob a forma de vários meios audiovisuais. 26

Sitemap Representação hierárquica da estrutura de um determinado sítio web. 2, 8, 9

SOLID Conjunto de princípios que incentivam a uma adequada implementação do paradigma de programação orientado a objectos. 26

Sprint No contexto da framework Scrum, trata-se do espaço de tempo onde tarefas são alocadas e desenvolvidas. Por norma, compreendem um período entre uma a quatro semanas. 24

XML Linguagem de marcação especificada pela W3C. É desenhada com o intuito de ser facilmente interpretada por humanos e máquinas. 10

Capítulo 1

Enquadramento e Introdução

O trabalho prático abordado no presente relatório foi desenvolvido no âmbito da unidade curricular Programação de Dispositivos Móveis e Multisensoriais da 8.^a edição do Mestrado em Engenharia Informática da Escola Superior de Tecnologia do Instituto Politécnico do Cávado e do Ave.

O referido trabalho prático incidiu no desenvolvimento de uma aplicação móvel, baseada no sistema operativo Android, cujo tema primordial é o escutismo. A aplicação foi previamente idealizada, especificada e desenhada na unidade curricular de Desenvolvimento de Interfaces Aplicacionais.

O desenvolvimento do presente documento aborda o ambiente de desenvolvimento empregue, a descrição funcional e arquitectura técnica da aplicação, a metodologia de desenvolvimento utilizada e os desafios encontrados no decorrer da implementação.

Capítulo 2

Desenvolvimento

2.1 Ambiente de desenvolvimento

O trabalho prático e correspondente relatório foram desenvolvidos com recurso às tecnologias e ferramentas abaixo listadas:

- Android Studio 4.2.1;
- Kotlin 1.5.10;
- Git 2.31.1;
- Firebase Android SDK;
- Google Maps Android SDK;
- L^AT_EX.

2.2 Descrição da aplicação

A aplicação móvel desenvolvida no contexto do presente documento, denominada de Escutas, pretende, em moldes sucintos, servir de suporte ao quotidiano de indivíduos pertencentes ao movimento escutista português. Pretende também dar a conhecer o movimento escutista a potenciais simpatizantes e visitantes.

Conforme referido no capítulo de introdução, a aplicação começou por ser concebida na unidade curricular Desenvolvimento de Interfaces Aplicacionais, com o apoio da professora Marisa Pinto. Foi no decorrer deste período que, entre diversas tarefas, se procedeu ao levantamento de requisitos e desenho gráfico da aplicação. A título de exemplo, observe-se a figura 1, que ilustra o logótipo da aplicação:



Figura 1: Logótipo da aplicação

Em termos funcionais, a aplicação concretiza diversos casos de uso, todos estes focados no ciclo de vida das actividades inerentes ao escutismo. O Sitemap, também este desenvolvido no decorrer da unidade curricular Desenvolvimento de Interfaces Aplicacionais, estabelece, de uma forma concisa e de simples interpretação, esses mesmos casos de uso. Analise-se o mesmo, na figura 2, adiante exposta:

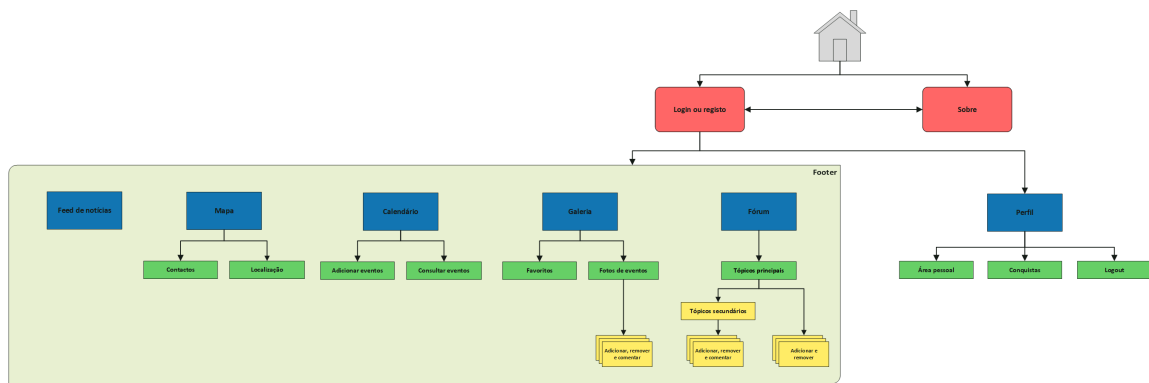


Figura 2: Sitemap da aplicação

Importa referir que, para efeitos do desenvolvimento do trabalho prático, o grupo de trabalho, juntamente com o professor Nuno Mendes, definiu alguns casos de uso como trabalho futuro. Isto porque não seria possível implementar todos os casos de uso em tempo útil. Dada a complexidade associada, optou-se pelos casos de uso associados ao fórum.

2.3 Arquitectura da solução

A solução desenvolvida baseia-se numa aplicação móvel, codificada na linguagem de programação Kotlin e linguagem de marcação XML, que é compilada contra a API nível 30 do sistema operativo Android. Além disso, recorre a variadas tecnologias na nuvem, que se caracterizam por modernas, flexíveis, extensíveis e escaláveis. Significativa parte das mesmas foram estudadas ou mencionadas no decorrer das aulas dedicadas à unidade curricular.

Numa perspectiva conceptual, é possível decompor a solução em diversos módulos distintos. Identifiquemos e avaliemos estes módulos, assim como respectivos componentes, através do diagrama presente na figura 3 adiante ilustrada:

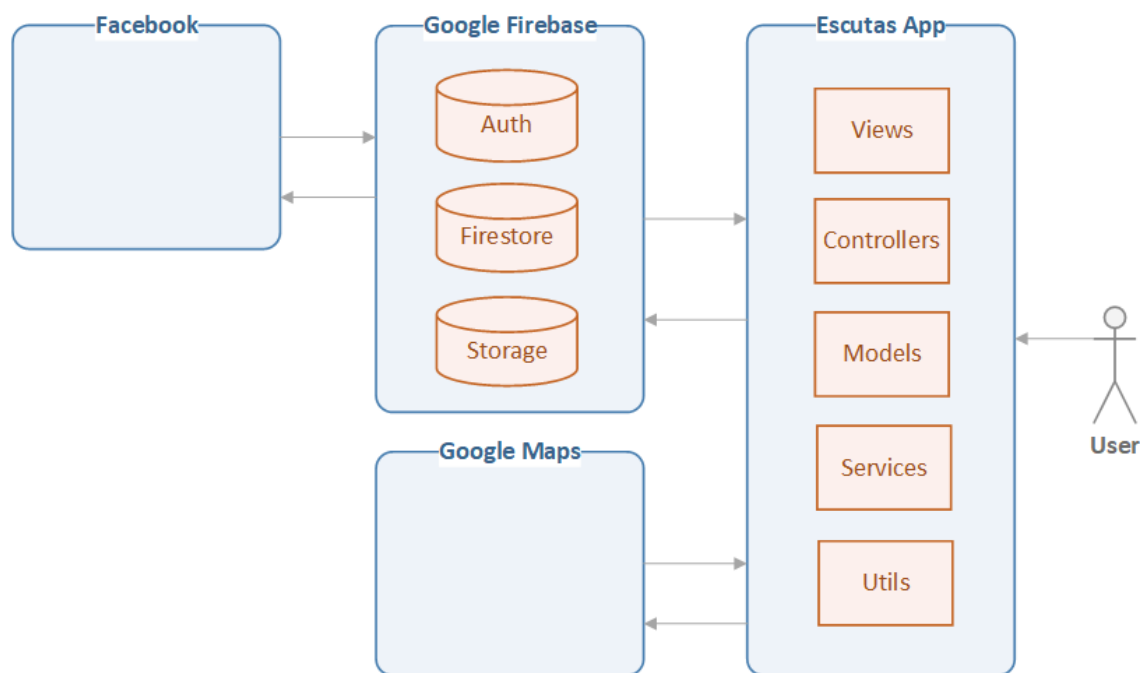


Figura 3: Arquitectura da solução

Como é possível constatar, apesar do foco ser a aplicação móvel, a solução é efectivamente distribuída. Além disso, a aplicação móvel faz uso do padrão MVC.

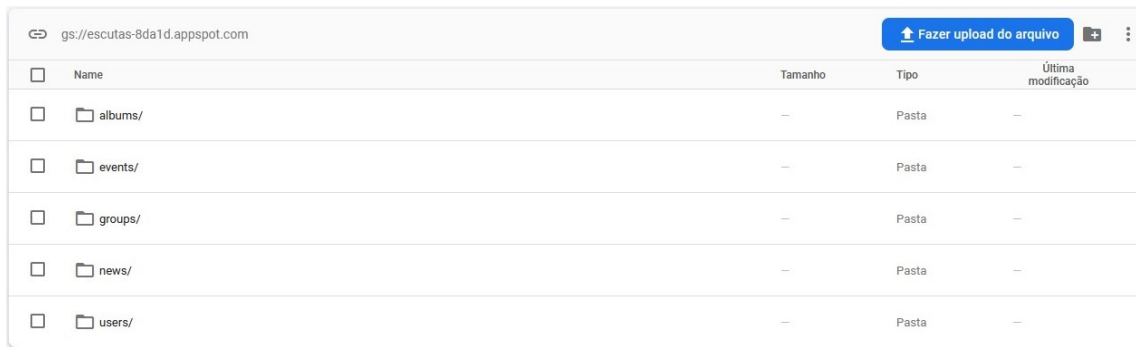
Tendo, como ponto de partida, o módulo que representa a aplicação móvel, segue-se uma breve descrição técnica de cada um dos componentes que compõem a solução.

2.3.1 Serviços

A camada de serviços da aplicação, é, na sua essência, responsável por abstrair os detalhes de comunicação da aplicação, com as respectivas dependências externas. De facto, é a camada de mais baixo nível da aplicação.

O comportamento de cada um dos serviços é primeiramente definido por um interface, sendo depois o serviço concretamente implementado, com base no dito interface. Além disto, para cada um dos serviços, existe a respectiva excepção, que é devidamente despoletada, no momento em que um determinado erro ocorre. Examine-se, adiante, o propósito de cada um dos serviços:

- *FirebaseAuthService*: serviço responsável por convencionar os mecanismos de autenticação da aplicação, sendo que, para tal, usa serviços de autenticação do Firebase, que, por sua vez, comunicam com serviços do Facebook. Em síntese, o serviço possibilita a autenticação de utilizadores através de credenciais (e-mail e palavra-chave), conta Google ou conta Facebook;
- *FirebaseDatabaseService*: serviço capaz de comunicar com o repositório de dados da aplicação. Para tal, são implementadas operações CRUD genéricas, independentes do modelo de dados. Conforme os requisitos estabelecidos no enunciado do trabalho prático, o repositório de dados utilizado é o Firebase Firestore. O mesmo segue o padrão não-relacional, que se caracteriza pela disponibilidade e escalabilidade;
- *FirebaseStorageService*: serviço focado na comunicação com o repositório de Blobs, denominado de Firebase Storage. À semelhança do serviço previamente citado, implementa também operações CRUD genéricas, agnósticas à tipologia dos Blobs em questão. É sobretudo usado para guardar imagens intrínsecas à aplicação. Embora este serviço não estivesse disposto como um requisito do trabalho prático, optou-se por o implementar, dada a sua manifesta utilidade. Compreenda-se, na figura 4 abaixo exposta, a estrutura do repositório de Blobs:



<input type="checkbox"/>	Name	Tamanho	Tipo	Última modificação
<input type="checkbox"/>	albums/	—	Pasta	—
<input type="checkbox"/>	events/	—	Pasta	—
<input type="checkbox"/>	groups/	—	Pasta	—
<input type="checkbox"/>	news/	—	Pasta	—
<input type="checkbox"/>	users/	—	Pasta	—

Figura 4: Estrutura do repositório de Blobs

2.3.2 Modelos

A aplicação define diversos modelos de dados. Os modelos de dados são depois instanciados e trocados entre as diferentes camadas da aplicação. Os mesmos apenas possuem estado e o seu desenho seguiu uma abordagem orientada ao domínio de negócio da aplicação. Veja-se os modelos de dados existentes:

- *Album*: como o nome indica, guarda um identificador, título e colecção de fotos. É expectável que o mesmo seja exibido na galeria da aplicação;
- *Event*: compreende o nome, descrição, foto, data de início e data de término de um evento. Adicionalmente, define se este é ou não partilhável entre os utilizadores. Os eventos encontram-se dispostos no calendário da aplicação;
- *Group*: integra o identificador, nome, descrição e coordenadas geográficas de um dado grupo de escutismo. Os marcadores do mapa da aplicação são desenhados com base nesta informação;
- *News*: inclui o identificador, título, corpo e foto de uma qualquer notícia a ser apresentada no feed de notícias da aplicação;
- *User*: engloba o identificador, email, nome, data de nascimento e grupo de escutismo de um determinado utilizador da aplicação.

2.3.3 Controladores

A camada de controladores da aplicação caracteriza-se por servir de intermediário entre a camada de serviços e a camada de vistas. Por consequência, contém expressiva parte da lógica de negócio da aplicação, uma vez que determina quando e de que forma os serviços são invocados e qual a informação que é disposta nas vistas.

Para cada uma das vistas presentes na aplicação, existe o respectivo controlador. Todos estes controladores herdam de um controlador comum, apelidado de *Base-Controller*, que inclui, como presumível, o estado e comportamento compartilhado. Por exemplo, os serviços de aplicação são instanciados no controlador comum, evitando assim que seja necessário instanciar os mesmos em cada um dos controladores derivados.

2.3.4 Vistas

A camada de vistas é a camada de mais alto nível da aplicação. Define-se como a camada de apresentação da solução, com a qual o utilizador interage directamente.

No que diz respeito à estrutura da dita camada, existem elementos partilhados por considerável parte das vistas associadas, nomeadamente o cabeçalho e o rodapé. Estes são fundamentais para o funcionamento da aplicação e partilham da mesma responsabilidade: potenciar a usabilidade da aplicação e a experiência do utilizador.

O cabeçalho inclui, opcionalmente, um botão que permite o utilizador regressar para a vista anterior, o título da vista actual e a imagem de perfil do utilizador. Ao pressionar a referida imagem de perfil, o utilizador é encaminhado para a sua área pessoal. Já o rodapé, compreende diversos botões, dispostos na horizontal, que possibilitam que o utilizador alterne entre as vistas que concretizam os principais casos de uso da aplicação, todos estes sujeitos a autenticação prévia. Para tal, a camada de vistas faz amplo uso de fragmentos, que são injectados entre o cabeçalho e o rodapé, através de uma vista base.

Complementarmente, a forma, disposição e dinamismo das vistas é construída com recurso a múltiplos ficheiros acessórios, que especificam cores, formas, vectores, fontes tipográficas e texto passível de ser localizado.

Considere-se, de seguida, uma explicação funcional, para cada uma das vistas presentes na aplicação, acompanhadas de figuras que ilustram a sua aparência:

- *Splash*: vista relativamente simples, na medida em que apenas exibe o logótipo da aplicação. A mesma é apresentada no arranque da aplicação, por um período de tempo programaticamente definido. Observe-se a figura 5 abaixo indicada:



Figura 5: Vista Splash

- *Login*: vista apresentada após o arranque supra citado. Na mesma, é possível realizar a autenticação através de credenciais, conta Facebook ou conta Google. Além disto, é também possível seguir para as vistas responsáveis pelo registo na aplicação ou recuperação de credenciais. Por último, é ainda exequível prosseguir para uma vista que expõe informação contextual sobre a aplicação. Averigúe-se a aparência da mencionada vista, através da figura 6, que se segue:

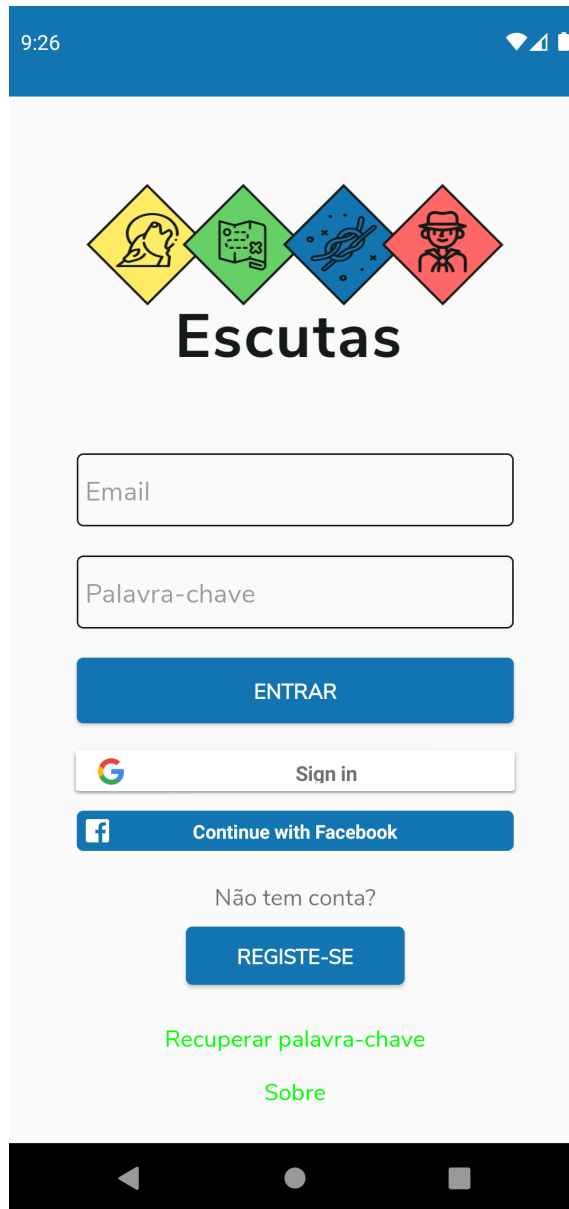


Figura 6: Vista Login

- *Register*: como o nome sugere, é a vista responsável pelo registo na aplicação, através de credenciais convencionais - email e palavra-chave. No caso de um utilizador optar pelas restantes formas de autenticação, na primeira vez que a realizar, será apresentada uma vista semelhante, contudo adaptada ao respetivo mecanismo de autenticação. Com vista a promover clareza, para efeitos de demonstração, é apresentada, através das figuras 7 e 8 infra mencionadas, a vista de registo através de credenciais convencionais. Já a vista de registo alternativa, poderá ser convenientemente consultada no código fonte da aplicação.

9:27

← Registo

Email *

utilizador@escutas.net

Nome de utilizador *

utilizador

Palavra-chave *

Palavra-chave

Repetir palavra-chave

Data de nascimento *

2021

Tue, May 25

< May 2021 >

Figura 7: Vista Register

9:27

← Registo

Data de nascimento *

2021

Tue, May 25

< May 2021 >

S	M	T	W	T	F	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Agrupamento *

Nenhum

CRIAR

Figura 8: Vista Register (continuação)

- *Recover*: se por ventura, um qualquer utilizador, registado através de credenciais convencionais, perder a sua palavra-chave, terá que a conseguir recuperar. A presente vista proporciona esse mesmo caso de uso. Verifique-se a figura 9:

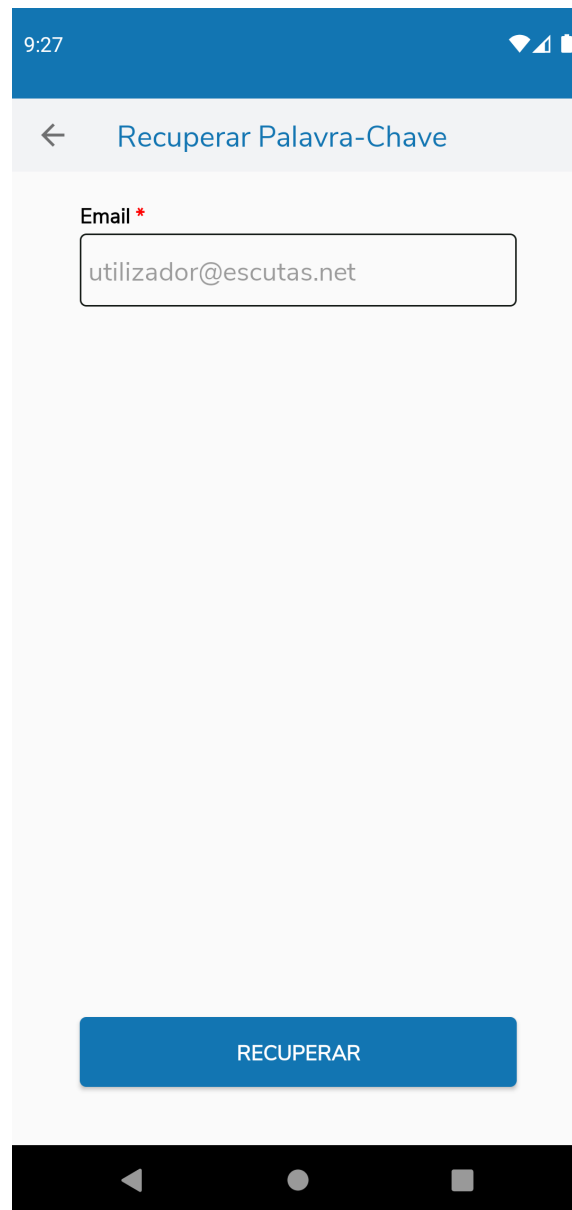


Figura 9: Vista Recover

- *About*: vista estática, e consequentemente simples, que abrange informação contextual sobre a aplicação. Nesta, está exposto o enquadramento da aplicação e o grupo de trabalho que desenvolveu a mesma, aliado aos respectivos meios de contacto. Confira-se a referida vista, por meio da figura 10, abaixo ilustrada:



Figura 10: Vista About

- *Profile*: vista que materializa a área pessoal do utilizador autenticado na aplicação. Esta é também relativamente descomplicada, na medida em que somente exhibe os detalhes do respectivo utilizador e consente que este termine, graciosamente, a sessão na aplicação. Veja-se a referida vista na figura 11 seguidamente evidenciada:

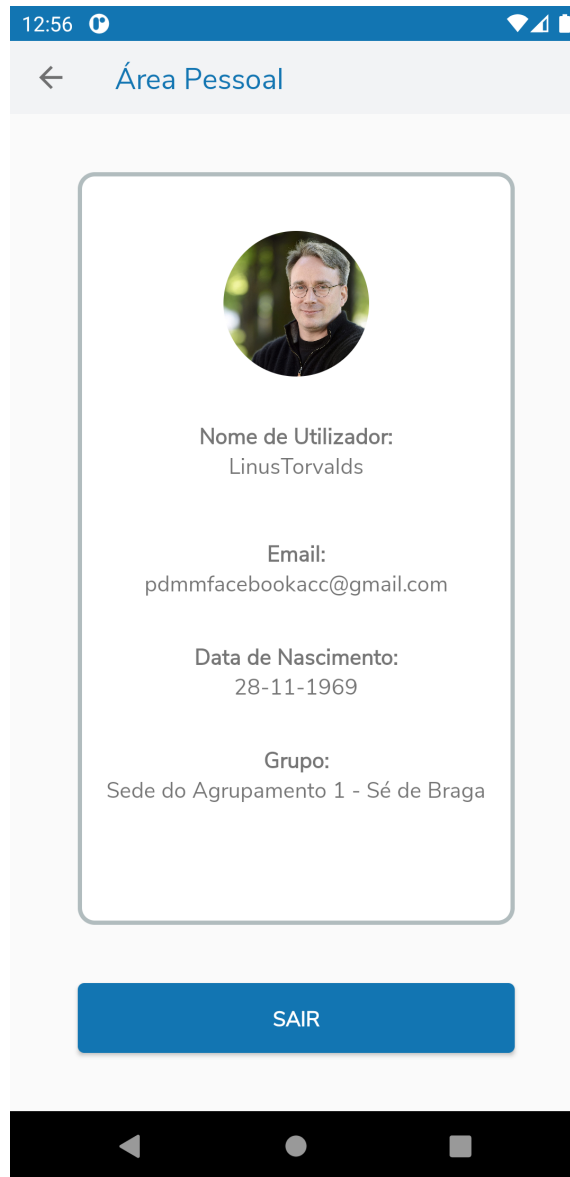


Figura 11: Vista Profile

- *Map*: vista encarregue de apresentar, através de um mapa interactivo, os grupos de escutismo que se encontram nas proximidades do utilizador da aplicação. Para tal, recorre à API do Google Maps e a mecanismos de localização do sistema operativo Android. Por sua vez, o referido sistema operativo utiliza o sensor GPS do dispositivo. Seguindo as boas práticas de desenvolvimento Android, as permissões de localização são meramente requeridas no momento em que se tornam indispensáveis. Perceba-se a vista na figura 12 que se segue:



Figura 12: Vista Map

2.3.5 Utilitários

Os utilitários são componentes genéricos e transversais, que podem ser utilizados por uma qualquer camada da aplicação, e que visam promover a reutilização de código. Os mesmos não têm estado, encontram-se organizados por tipo de dados e incluem exclusivamente métodos estáticos, que realizam tarefas pequenas e repetidas.

Considere-se uma síntese de cada um dos utilitários:

- *StringUtils*: utilitário que guarda métodos intrínsecos ao tipo de dados *string*. Por exemplo, um métodos de extensão contido neste utilitário, valida se uma determinada *string* segue o padrão de um endereço de email, através do uso de expressões regulares;
- *DateUtils*: utilitário que armazena métodos inerentes ao tipo de dados *Date*. Exemplificando, neste utilitário, existe um método, que recebe como argumentos, um dia, mês e ano - no tipo de dados *int*. Devolve, consequentemente, a respectiva data - no tipo de dados *Date*.

2.4 Metodologia de desenvolvimento

A metodologia de desenvolvimento da aplicação focou-se em três vectores de acção: versionamento, planeamento e boas práticas. As sub-secções que se seguem abordam, para cada um dos vectores de acção supra citados, os tópicos considerados relevantes.

2.4.1 Versionamento

As operações de versionamento da aplicação, recorrem ao sistema de controlo de versões Git, cujo repositório encontra-se alojado na plataforma GitHub. Os elementos do grupo de trabalho têm o devido acesso ao repositório e podem, conseqüentemente, contribuir para o mesmo, conforme a organização e regras estipuladas. Começando pela organização do repositório, o mesmo é composto por diversos ramos, que servem diferentes propósitos. Considere-se cada um deles:

- *dev*: ramo de desenvolvimento ativo, para onde as contribuições provenientes dos ramos de cada elemento do grupo de trabalho, são fundidas. Dada a sua natureza, os binários resultantes da compilação do código fonte do presente ramo poderão encontrar-se num estado instável;
- *qa*: ramo exclusivamente dedicado a testes, com vista a assegurar a qualidade e estabilidade da aplicação desenvolvida. Ao contrário do ramo anteriormente referido, o presente ramo não acolhe contribuições individuais, sendo que é simplesmente sincronizado com o ramo anterior, em momentos considerado oportunos pelo grupo de trabalho;
- *main*: quando, depois dos testes realizados na linha previamente citada, uma determinada versão da aplicação é considerada plenamente funcional e estável, o seu conteúdo é sincronizada com a presente linha. A partir desta, são geradas *releases* com binários aptos para produção. A título de exemplo, os binários submetidos no trabalho prático, são provenientes da presente linha.

No que diz respeito a regras, uma determinada contribuição de um elemento do grupo de trabalho, é necessariamente realizada através de um *pull request*. Neste, o elemento terá que responder a uma série de questões sobre o conteúdo e finalidade da sua contribuição, previamente fixadas num *pull request template*.

Após a submissão do *pull request*, o mesmo é eventualmente revisto pelos restantes elementos do grupo de trabalho. Mediante a revisão, o grupo de trabalho poderá aceitar, rejeitar ou requerer modificações à contribuição. Assim que a contribuição for aceite por, pelo menos, dois elementos do grupo de trabalho, o autor da contribuição poderá fundir a sua contribuição com a linha alvo. Observe-se os detalhes de um *pull request*, realizado durante o decorrer do desenvolvimento do trabalho prático, na figura 13, adiante ilustrada:

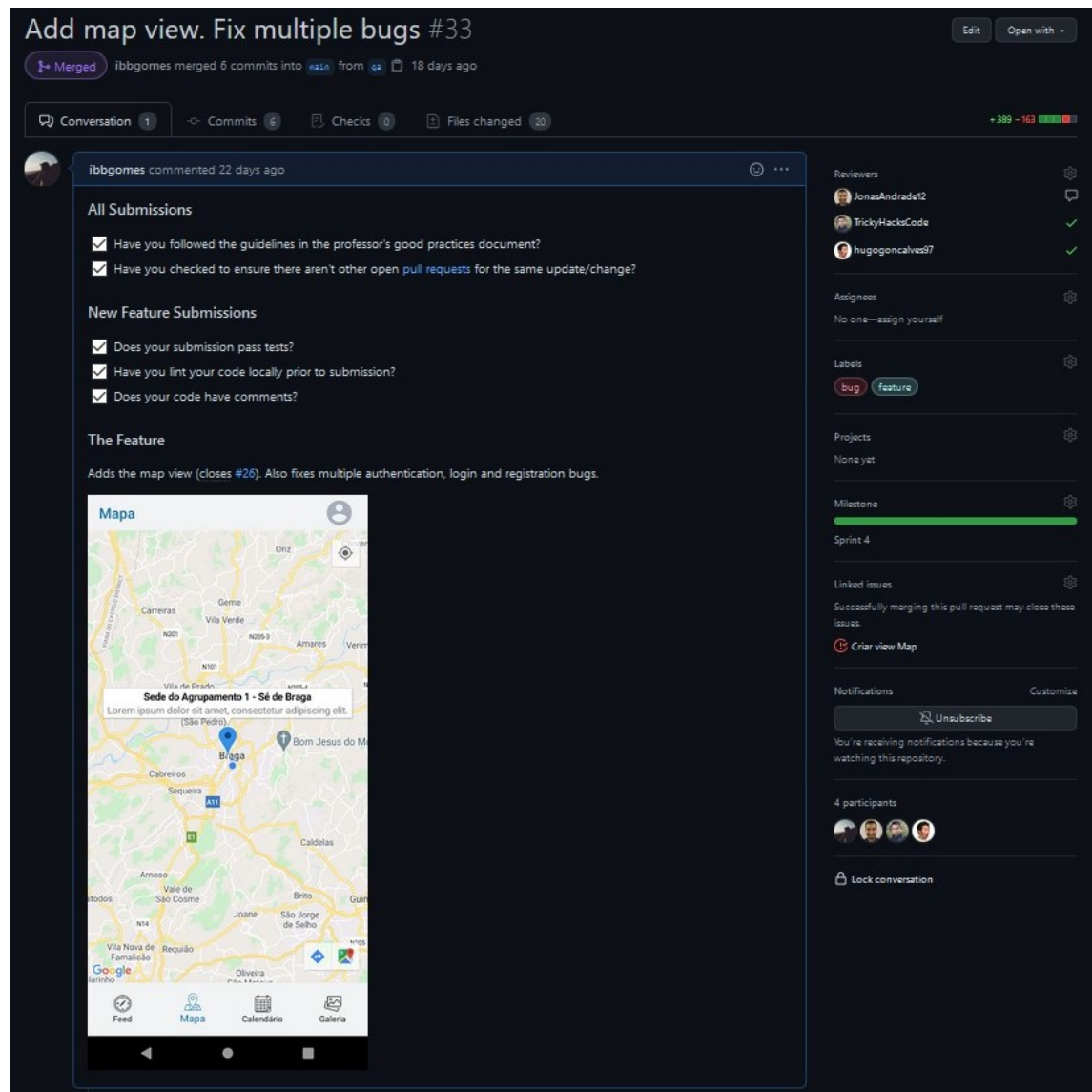


Figura 13: Exemplo de um *pull request*

2.4.2 Planeamento

O desenvolvimento da aplicação foi planeado e gerido com recurso a metodologias de desenvolvimento ágeis, nomeadamente a Framework Scrum, sendo que cada Sprint teve a duração de uma semana, decorrida entre aulas dedicadas à unidade curricular.

Com vista a auxiliar as tarefas de planeamento e gestão associadas ao processo de desenvolvimento da aplicação, recorreu-se às ferramentas de gestão de projecto da plataforma GitHub. Primeiramente, começou-se por especificar Sprints até à data estabelecida para a entrega do trabalho prático, em forma de *milestones*. Veja-se algumas destas Sprints, através da figura 14, seguidamente disposta:

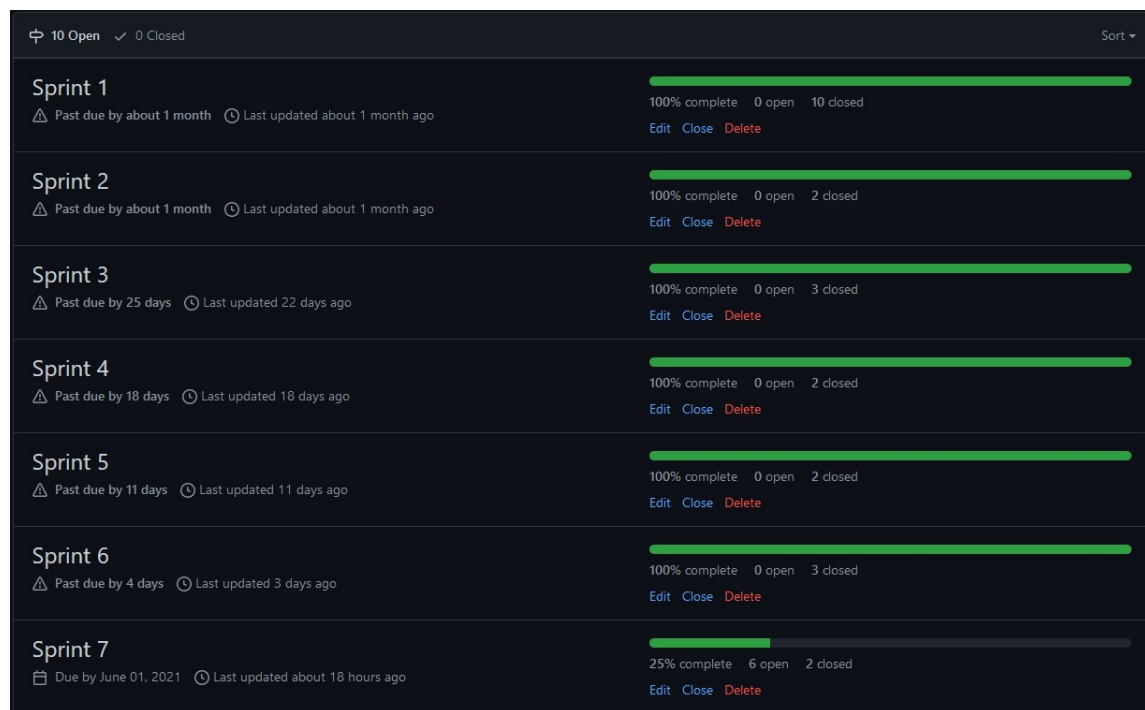


Figura 14: Sprints do projecto

Em cada uma das aulas da unidade curricular, onde o grupo de trabalho estava presencialmente ou remotamente reunido, procedeu-se ao planeamento da Sprint seguinte, através da alocação de tarefas provenientes do Backlog do projecto, regularmente mantido sob a forma de *issues*. Analise-se, em baixo, a figura 15, que demonstra o Backlog do projecto num dado instante:

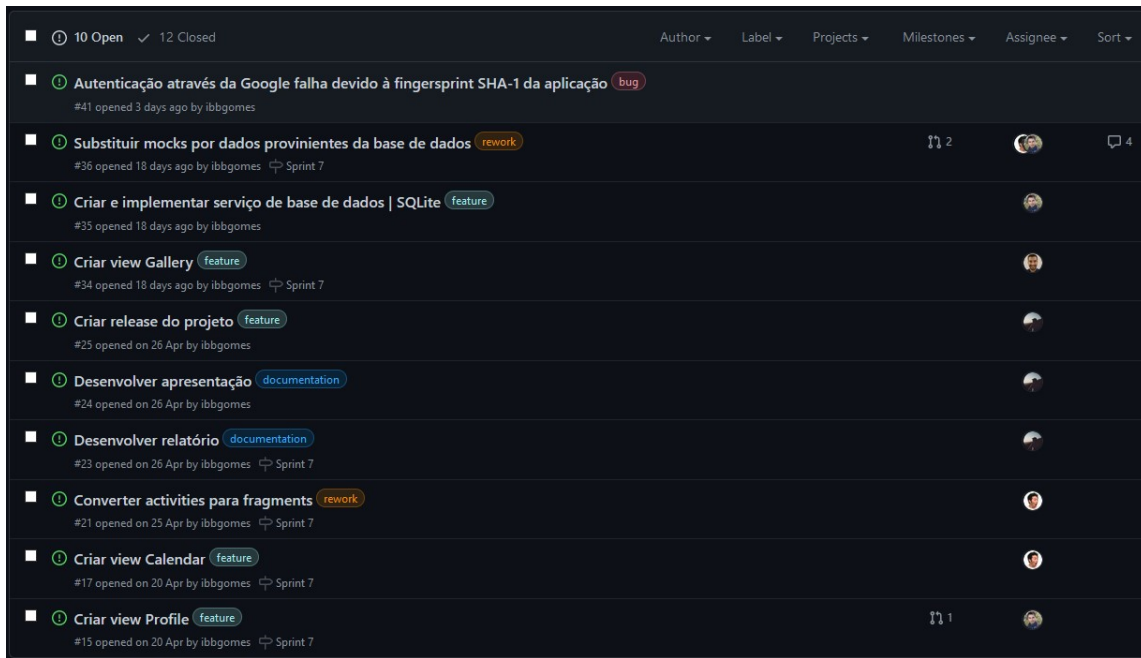


Figura 15: Backlog do projecto

Como é possível constatar na figura anteriormente citada, as tarefas assumem variadas tipologias. Para concretizar tal coisa, foram definidas *labels*, através de cores sugestivas, que refletem a positividade da tarefa no contexto do desenvolvimento do projecto. Faça-se uma breve descrição de cada uma delas:

- *Feature*: representa uma tarefa que diz respeito ao desenvolvimento de uma qualquer nova funcionalidade na aplicação;
- *Documentation*: retrata uma tarefa de documentação. Por exemplo, a escrita do presente documento, foi definida como tarefa de documentação;
- *Rework*: corresponde a uma tarefa que implica a reestruturação de uma dada funcionalidade existente na aplicação;
- *Bug*: refere-se a uma tarefa que incide sobre a resolução de um determinado erro identificado na aplicação.

2.4.3 Boas práticas

No decorrer da construção da aplicação, fez-se por seguir ou implementar aquilo que são transversalmente reconhecidas como boas práticas de desenvolvimento de software. Examine-se algumas das práticas empregues:

1. Dado que a aplicação foi desenvolvida através do paradigma de programação orientada a objectos, fez-se por cumprir os princípios SOLID. Adicionalmente, com vista a promover a manutibilidade e simplicidade do código fonte, seguiu-se também os princípios DRY e KISS;
2. De modo a detectar erros e garantir a uniformidade do código fonte, fez-se uso de um Linter, direccionado à linguagem de programação Kotlin, que se encontra convenientemente integrado com o sistema de automação de compilação da aplicação. O mesmo é apelidado de *ktLint*, é activamente desenvolvido pelo Pinterest e encontra-se ao abrigo de uma licença de código aberto;
3. O código fonte da aplicação, quando compilado para ambientes de produção, gera binários que se encontram adequadamente compactados, otimizados e ofuscados. Para tal, são utilizadas as opções de compilação *minifyEnabled* e *shrinkResources*.

2.5 Desafios de implementação

O principal desafio de implementação da aplicação, foi o facto de nenhum elemento do grupo de trabalho, ter conhecimento ou experiência prévia no desenvolvimento de aplicações móveis. Como expectável, este facto traduziu-se numa velocidade de implementação inicialmente reduzida, que foi progressivamente aumentando, à medida que o grupo de trabalho adquiria conhecimento, experiência e agilidade na plataforma Android, Firebase e tecnologias associadas.

Existiram ainda alguns contratempos na implementação de rotinas assíncronas através da linguagem de programação Kotlin. Não obstante, estes foram trivialmente ultrapassados, através de espírito de entreajuda e auxílio do professor.

Capítulo 3

Conclusão

Terminado o trabalho prático e considerando a natureza empírica do mesmo, cremos que o trabalho desenvolvido foi essencial para a consolidação dos conhecimentos transmitidos pelo professor no decorrer das aulas dedicadas às temáticas abordadas. Acreditamos também que o desenvolvimento deste permitiu amadurecer as nossas competências técnicas na área de programação móvel - nomeadamente no que diz respeito ao ecossistema Android. Tivemos ainda oportunidade de explorar o referido ecossistema através de ferramentas até então não empregues, como foi o caso do Firebase e Google Maps, o que foi uma experiência altamente gratificante.

Relativamente ao trabalho desenvolvido, julgamos que este cumpre competentemente os requisitos mencionados no desafio que nos foi lançado pelo professor. Gostaríamos de ter desenvolvido um trabalho mais aprofundado - particularmente na implementação de pequenos pormenores funcionais que tornariam a aplicação mais rica - contudo, a carga de trabalho característica de uma pós-graduação, aliada às nossas obrigações profissionais, limitou o tempo passível de ser despendido a trabalhar para cada uma das unidades curriculares.

Capítulo 4

Bibliografia

- [1] *Android Developers Documentation*. Google Developers, 2021. <https://developer.android.com/docs>.
- [2] *Firebase Documentation*. Google Developers, 2021. <https://firebase.google.com/docs>.
- [3] *Google Maps Platform Documentation*. Google Developers, 2021. <https://developers.google.com/maps/documentation>.
- [4] *Kotlin Docs*. Kotlin Foundation, 2021. <https://kotlinlang.org/docs>.
- [5] *Git Documentation*. Git Community, 2021. <https://git-scm.com/doc>.
- [6] N. F. Mendes, *Programação de Dispositivos Móveis e Multisensoriais*. Instituto Politécnico do Cávado e do Ave, 2021.