



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

Speedy Manager #24



Rui Inácio (45109)

João Folgado (45128)

Orientador

Professor Doutor António Teófilo

Julho, 2021

Resumo

O Speedy Manager é uma aplicação móvel que permite a organização de tarefas e listas de tarefas. Esta lista de tarefas pode ser partilhada e manipulada por vários utilizadores simultaneamente.

O principal objectivo é dispor de uma interface amigável para o utilizador, mas que consiga incorporar diferentes elementos de aplicações do mesmo género tais como sistema de chat entre utilizadores, partilha de conteúdos multimédia nas tarefas e uma interface actualizada em tempo real com alterações feitas por outros utilizadores.

A aplicação foi construída em Kotlin, uma linguagem de programação multi plataforma com interoperabilidade total com Java, e que é referenciada no desenvolvimento de aplicações móveis. Além do Kotlin a aplicação integra o sistema de Firebase da Google que permite que os utilizadores utilizem a sua conta Google na aplicação e ainda, este sistema, dispõe de ferramentas tais como armazenamento em cloud, base de dados em tempo real, autenticação de utilizadores, entre outros.

Motivação

A motivação para a realização desta aplicação provem do facto de, embora existirem várias aplicações com o mesmo intuito, muitas delas possuem elementos/funcionalidades distintas que podem ser reunidas numa só aplicação. Ainda que a partilha entre utilizadores da organização de listas/tarefas possa ser encontrada em várias aplicações deste género de renome, notamos que, estas muitas vezes não possuem funcionalidades básicas como partilha de multimédia e conversação em tempo real ou suporte multi-plataforma. O Speedy Manager visa então entregar ao utilizador uma aplicação simples de se manusear, mas que disponha de várias ferramentas e funcionalidades que enriquecem a experiência na organização de tarefas.

Abstract

We live in a world were, now more than ever, we have increasingly more tasks to do. Being that every year that passes our society goes further and further into the digital realm, our calendars are now inside our phones rather than a physical notebook, which in turn, makes it so having a mobile application that helps us manage what we must do and when more useful than having it written in a notebook.

The Speedy Manager comes with the ambition of answering the need for the necessity explained in the previous sentence, it will allow users to create and manage task lists as well as tasks inside those lists. It will also allow users to share those lists among other users which will grant users the ability to have multi-user lists which are updated in real-time everyone time one of those users makes a change.

This project will have two main strands, "front-end" and "back-end", the first one will focus mainly on the visual part of the application ensuring an easy to use, clean and amicable interface and the latter will ensure the security and integrity of the data to be displayed in the "front-end" part of the application.

With this goal in mind and, with all the knowledge obtained during development as well as throughout the course the objective is to develop both parts of this application to the best of our abilities.

Agradecimentos

Aqui deixamos o nosso agradecimento ao ISEL e aos seus intervenientes por todo o conhecimento, formação e crescimento pessoal. Um especial agradecimento a todos os docentes que de qualquer forma contribuíram para melhorarmos ou expandirmos a nossa maneira de este mundo.

Agradecemos também ao Professor Doutor António Teófilo orientação que nos ofereceu durante a realização deste projecto.

Um obrigado à família, amigos e colegas de licenciatura pelo apoio ao longo da realização deste projecto.

Índice

Resumo	i
Motivação	iii
Abstract	v
Agradecimentos	vii
Índice	ix
Lista de Tabelas	xiii
Lista de Figuras	xv
1 Introdução	1
2 Trabalho Relacionado	3
2.1 Contexto da aplicação	3
2.2 Estudo de mercado	3
2.2.1 Estrutura de aplicações deste tipo	4
2.2.2 Comparação entre outras plataformas	8
3 Modelo Proposto	11
3.1 Requisitos	11
3.1.1 Casos de utilização	11
3.1.2 Funcionalidades do sistema	13
3.2 Fundamentos	15
3.2.1 Google FireBase	15
3.2.2 Aplicação Android	16

3.2.3	Princípios de um sistema na nuvem	18
3.2.4	Serviços <i>Cloud</i> e aplicações <i>Cloud Based</i>	19
3.3	Abordagem	20
4	Implementação do Modelo	23
4.1	Análise Inicial	23
4.2	Design	24
4.2.1	Realização de wireframes	24
4.2.2	Realização de Mockups	25
4.2.3	Obtenção de feedback para Design proposto	28
4.2.4	Analise dos resultados obtidos no questionário	29
4.3	Desenvolvimento	31
4.3.1	Criação de tarefas	31
4.3.2	Construção de listas de tarefas	32
4.3.3	Desenvolvimento de um protótipo	34
4.4	Implementação da navegação e funcionalidades	36
4.4.1	Navegação entre fragmentos	36
4.4.2	Tarefas urgentes e listas recentes	38
4.4.3	Menu das tarefas	39
4.4.4	Tarefas com imagens	42
4.4.5	Sistema de procura por contactos	44
4.4.6	Partilha de listas entre contactos	44
4.4.7	Tarefas prioritárias	46
4.4.8	Chat	48
4.4.9	Monitorização da lista	49
4.4.10	Menu da lista	51
4.4.11	Integração com o <i>DarkMode</i>	53
5	Validação e Testes	55
5.0.1	Pagina inicial	56
5.0.2	Pagina das listas de um utilizador	57
5.0.3	Paginas do utilizador e contactos	58
5.0.4	Pagina de uma lista de tarefas	60
6	Conclusões e Trabalho Futuro	61

<i>CONTENTS</i>	xi
A Repositório web	63
B Sistema de controlo de versões	65
Bibliografia	69

Lista de Tabelas

2.1	Tabela de funcionalidades	8
3.1	Caracterização das funções do sistema	13
3.2	Caracterização dos requisitos da aplicação	14
3.3	Computação na nuvem	18

Lista de Figuras

2.1	Aplicação Movel do Trello	5
2.2	Interface Web do Trello	6
2.3	Interfaces Microsoft To Do e Google Tasks comparados	7
3.1	Diagrama de casos de utilização	12
3.2	Firebase	15
3.3	Kotlin	17
3.4	exemplos de plataformas/serviços cloud	19
4.1	Wireframes Ecrãs de login e registo	24
4.2	Wireframes Ecrãs principais bottom bar navigation	25
4.3	Wireframes Ecrãs dentro de uma lista	25
4.4	MockUps Ecrãs de login e registo	26
4.5	MockUps Ecrãs principais bottom bar navigation	27
4.6	MockUps Ecrãs dentro de uma lista	27
4.7	Exemplo de perguntas realizadas	28
4.8	Estudo da população questionado	29
4.9	Respostas á presença de chat numa lista	29
4.10	Popup criação de uma nova tarefa	31
4.11	Exemplo de tarefa simples	31
4.12	Exemplo de tarefa composta	32
4.13	Comando Swipe associado a uma tarefa	33
4.14	Comando Swipe associado a uma tarefa	34
4.15	Modelo inicial da aplicação	35
4.16	Menu inferior de navegação	36
4.17	Ecrã inicial	38
4.18	Tarefa simples com menu fechado	39

4.19 Tarefa simples com menu aberto	39
4.20 Tarefa composta com menu aberto	40
4.21 Menu da sub-tarefa	40
4.22 Progresso de uma tarefa composta	41
4.23 Escolha da imagem	42
4.24 Tarefa com imagem	43
4.25 Exemplo de pesquisa	44
4.26 Processo de convidar um utilizador para uma lista	45
4.27 Utilizadores da lista	45
4.28 Exemplo de um tarefa prioritária	46
4.29 Exemplo de um tarefa prioritária	47
4.30 Exemplo de uma conversa no chat de uma lista	48
4.31 Exemplo de um histórico de alterações numa lista	49
4.32 Gráfico de participação	49
4.33 Gráfico de quem completa mais tarefas	50
4.34 Gráfico de eventos	50
4.35 Menu de uma lista	51
4.36 Exemplo de lista com imagem e outra com cor	52
4.37 Aplicação tema normal e com tema escuro activo	54
5.1 Ecrã inicial	56
5.2 Ecrã das listas do utilizador	57
5.3 Ecrã de contactos e de pesquisa de utilizadores respectivamente	58
5.4 Ecrã de perfil próprio, de contacto, utilizador não adicionado, respectivamente	59
5.5 Lista usada como tutorial da aplicação.	60
A.1 Repositório GitHub	63
A.2 Acesso ao .apk da aplicação no repositório	64
B.1 Armazenamento das versões iniciais da aplicação	66

Capítulo 1

Introdução

Este projecto aborda o desenvolvimento de uma aplicação para dispositivos moveis Android, que permita a gestão de tarefas e listas de tarefas entre utilizadores, com principal foco na rápida utilização e interface intuitiva.

Existem diferentes tipos de aplicação que trabalham a temática de anotação de tarefas para ajuda na organização pessoal e profissional. As aplicações deste género tipicamente associam uma tarefa a uma descrição e um senso de estado, entre outros anexos e funcionalidades que o sistema possua. Por norma a organização de tarefas, neste tipo de aplicações, passa por agrupar estas listas, em que cada tarefa seria um ponto a completar de algo maior.

O principal objetivo deste projeto é o desenvolvimento de uma aplicação com uma interface amigável que permita a gestão de tarefas através de comandos de swipe ou drag and drop, que possam ser manipuladas por vários utilizadores com um funcionamento multi-utilizador em real time.

A abordagem seguida no desenvolvimento deste projecto começou por fazer um estudo de mercado para perceber e clarificar os requisitos da aplicação bem como pontos chave, pontos de inovação e casos de utilização em aplicações do mesmo género/tema.

Foi seguida a típica abordagem de desenvolvimento de uma aplicação móvel onde, através de uma analise do problema em vários pontos, se concebe *sketches e mockups* de modo a compor um desenho interactivo inicial que possa ser avaliado através de questionários respondidos por pessoas fora ao desenvolvimento. Esta abordagem permite obter *feedback* sobre diferentes pontos e ideias em fases iniciais do desenvolvimento.

Seguidamente começamos a implementar um prototípico funcional da

aplicação. Este serviu de motor base para futuro encaixe dos outros diferentes componentes da aplicação. Este protótipo consistiu essencialmente num sistema de registo e login de utilizadores na aplicação juntamente com um modelo inicial criação de tarefas e listas de tarefas e partilha das mesmas.

Seguido do desenvolvimento do protótipo foram depois implementadas neste as restantes funcionalidades da aplicação relacionadas com a interação entre utilizadores bem com funcionalidades das tarefas e listas.

O grande foco de desenvolvimento desta aplicação cabe dentro do front-end, sendo que este trata de tudo o que contribui para a visualização da informação armazenada no back-end. O back-end é um serviço cloud, a Firebase, de auxílio ao desenvolvimento de aplicações móveis prestado pela Google. A comunicação entre o front-end e o back-end é feita através de DataSnapshots, que são um método de troca de informação entre o dispositivo e o banco de dados. Este mecanismo de troca de informação é nativo do sistema da Firebase e permite troca de dados em tempo real.

Com isto será possível que, quando existam alterações no back-end, sejam estas criar uma nova tarefa, adicionar uma imagem, entre outras, consigamos actualizar seguidamente no front-end a informação em tempo real.

Capítulo 2

Trabalho Relacionado

2.1 Contexto da aplicação

Esta aplicação insere-se no grupo de aplicação de auxilio pessoal em quesitos de memorização e organização de tarefas. Existe um mercado enorme de aplicações deste tipo, o que será abordado na sub-secção seguinte deste capítulo assim como um estudo desse mercado centrado naquelas que consideramos as três aplicações deste tipo mais influentes.

O contexto para o desenvolvimento desta aplicação consiste em associar a tarefas numa lista mecanismos de controlo como o comando *drag/drop* ou *Swipe*, elementos visuais como cor e imagens, e ainda, partilha de listas entre utilizadores. Isto porque como poderemos observar no estudo de mercado seguinte, as aplicações mais influentes não possuem algumas destas funcionalidades e são bastante elementares visualmente.

2.2 Estudo de mercado

Como esta aplicação tem como principal objectivo permitir aos seus utilizadores, através de uma interface gráfica simpática, uma fácil gestão de tarefas, fizemos um estudo do mercado com o objectivo de ver e comparar as diferentes aplicações que operam sobre a mesma temática.

Esse estudo de mercado serviu não só para visualizar diferentes concretizações de aplicações neste tema, mas também para obter uma melhor ideia de como devem ser construídas interfaces gráficas bem como componentes constituintes importantes deste tipo de aplicações.

Para tal, decidimos comparar as funcionalidades entre várias aplicações semelhantes à nossa, nomeadamente entre [MicrosoftToDo, 2017], [GoogleTasks, 2018], e o [Trello, 2011].

2.2.1 Estrutura de aplicações deste tipo

Entre as diferentes aplicações acima mencionadas podemos observar essencialmente duas maneiras de como executam o processo de guardar uma tarefa descrita pelo utilizador. As ditas tarefas são descritas/mostradas ao utilizador num sistema de listagem ou um sistema de "quadro" que contém secções para as tarefas. De seguida serão apresentadas imagens de algumas das aplicações mencionadas bem como uma descrição/considerações sobre as mesmas.

Trello

O [Trello, 2011] é uma aplicação que apresenta uma interface diferenciada e que se pode facilmente tornar bastante complexa. A organização das tarefas é feita arrastando através do movimento de tarefas ou listas . Um utilizador pode ter vários quadros e estes podem ser partilhados entre pessoas. No [Trello, 2011] as tarefas podem incorporar elementos de multimédia.

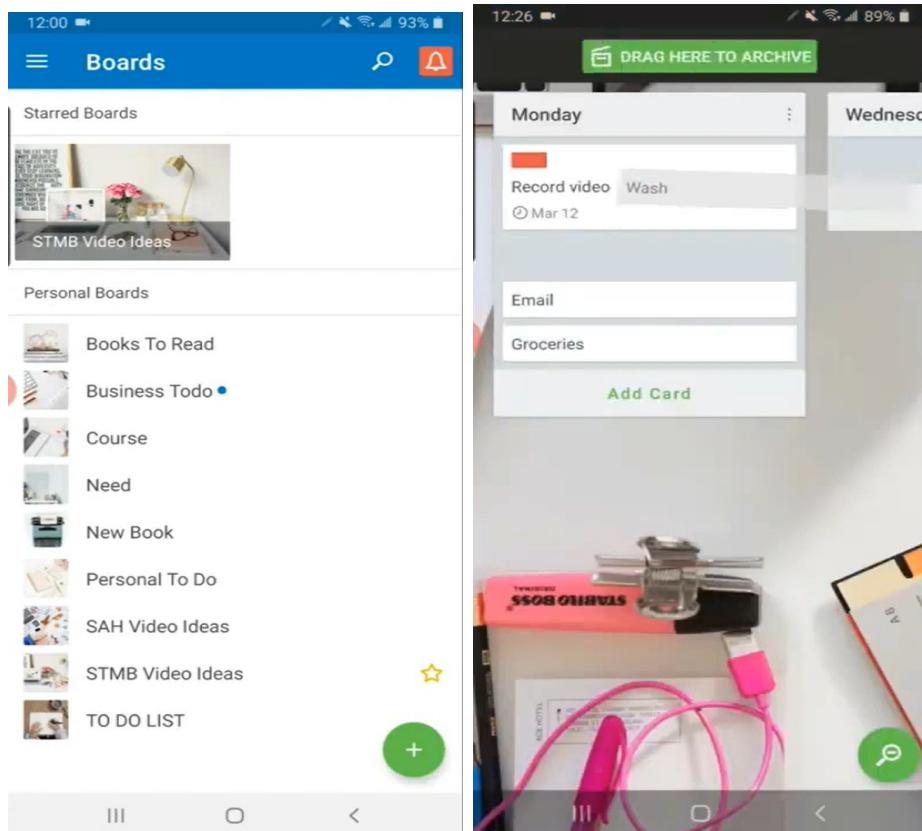


Figura 2.1: Aplicação Movel do Trello

O [Trello, 2011] é uma aplicação que embora siga a mesma temática do SpeedyManager não possui os mesmo fundamentos de ser simples, *user-friendly* e de rápida utilizacão.

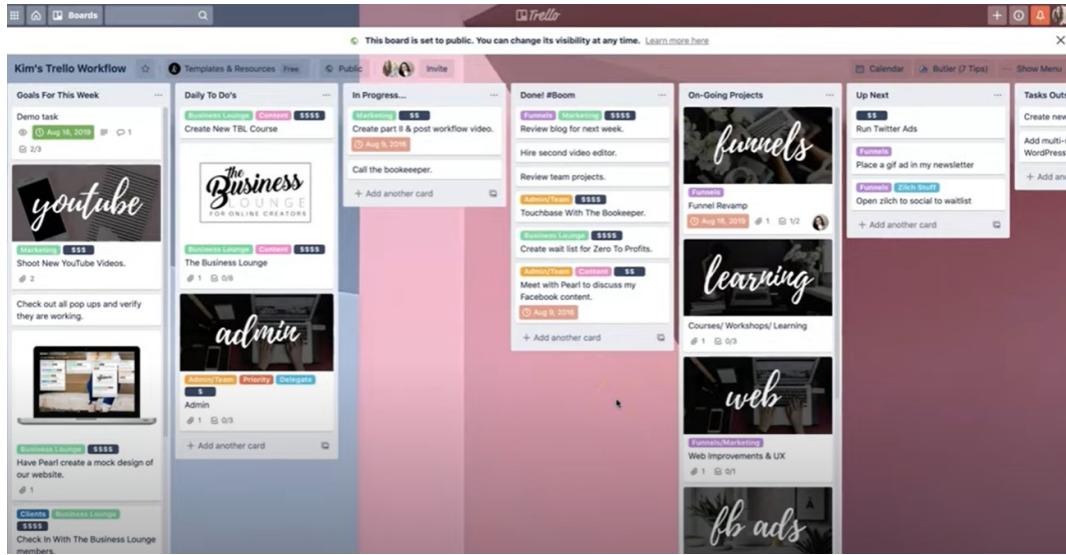


Figura 2.2: Interface Web do Trello

Microsoft Todo e Google Tasks

O [MicrosoftToDo, 2017] e o [GoogleTasks, 2018] são aplicação da mesma temática que se encaixam melhor no contexto do SpeedyManager. Essencialmente a diferença entre as duas é que o [MicrosoftToDo, 2017] permite a partilha de tarefas e listas entre Utilizadores e o [GoogleTasks, 2018] não.

Estas aplicações estão mais dentro dos mesmos fundamentos do SpeedyManager devido ao facto de, a sua utilização ser consideravelmente mais simples que o Trello e a abordagem usada na interface ser baseada no conceito de uma lista de tarefas.

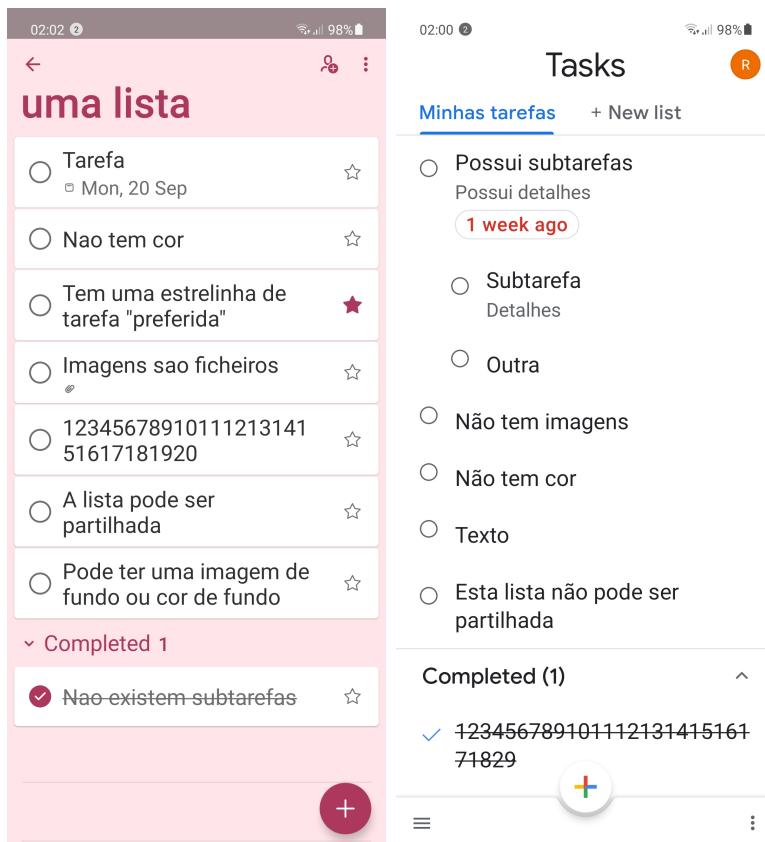


Figura 2.3: Interfaces Microsoft To Do e Google Tasks comparados

Concluímos então que o [Trello, 2011] é, entre as 3, a aplicação mais completa, porém nós temos o objectivo de implementar algo que seja mais "*on the go*", prático e rápido de utilizar.

2.2.2 Comparação entre outras plataformas

Após essa investigação e a utilização das aplicações mencionadas anteriormente conseguimos ter uma melhor percepção dos diferentes componentes/funcionalidades que constituem uma aplicação deste tipo e decidimos compará-las com as nossas ideias. Colocando estes componentes em perspectiva com outros que consideramos serem também importantes a uma aplicação deste género elaboramos a seguinte tabela que compara diferentes funcionalidades.

Features	Trello	Microsoft To Do	Google Tasks	Speedy Manager
Friendly, accessible, intuitive interface	✓	✓	✗	✓
Task creation	✓	✓	✓	✓
Creating Task Lists	✓	✓	✓	✓
Swipe commands to complete tasks	✗	✗	✓	✓
Swipe commands to delete tasks	✗	✓	✗	✓
Tasks with colors	✓	✗	✗	✓
Tasks with images	✓	✗	✗	✓
Tasks with files (txt, pdf)	✓	✓	✗	✗
Colored task lists	✓	✓	✗	✓
Task lists with default application images	✓	✓	✗	✗
Task lists with images of the user's choice	✓	✗	✗	✓
Task lists with task lists inside	✓	✓	✓	✓
Progress bar within task lists	✓	✗	✗	✓
Chat function	✓	✗	✗	✓
Shared task lists	✓	✓	✗	✓
Real-time synchronization	✓	✓	✗	✓
Drag tasks up and down	✓	✓	✓	✓
Tasks organized by date	✓	✓	✓	✓
Tasks organized by priority	✓	✓	✗	✓

Tabela 2.1: Tabela de funcionalidades

A análise a retirar desta tabela não é que o SpeedyManager vise conter todas as funcionalidades e mais algumas que estas aplicações possuem mas sim demonstrar ao leitor que mesmo aplicações deste tipo possuem funcionalidades diferentes e destinas umas das outras. Por exemplo consideramos o [GoogleTasks, 2018] e o porque de ele pontuar tantas cruzes nesta lista.

O [GoogleTasks, 2018] não possui um sistema de partilha de tarefas entre utilizador e por tal é essencialmente uma aplicação nesta temática mas ”*singleplayer*”e por tal não pontua em inúmeras funcionalidades relacionadas a integração de tarefas entre diferentes utilizadores.

Um outro ponto importante mencionar é a pontuação do [Trello, 2011] nesta tabela que na sua integra contem todas as funcionalidades(excepto os comandos associados ao *swipe*, no entanto, como já foi referido anteriormente este possui uma interface de funcionamento substancialmente diferente das duas outras aplicações comparadas e aquilo que visamos ser a interface do SpeedyManager.

Por tal podemos considerar que de facto existe um espaço a ser preenchido pelo o SpeedyManager no que toca a apresentar ao utilizador uma interface simples e prática tal como a do Google Tasks e To Do mas capaz de integrar varias funcionalidades não disponíveis nestas plataformas de planeamento mais ”rudimentares”.

Capítulo 3

Modelo Proposto

3.1 Requisitos

Esta aplicação tem como requisitos funcionais a criação e gestão de listas de tarefas sob o uso de uma *cloud database* que permita sincronização em tempo real de uma lista entre os vários utilizadores. A interface da aplicação deve ser simples e apelativa ao utilizador.

3.1.1 Casos de utilização

Esta analise inicial decorre temporalmente junto com o estudo de mercado e a definição do aspectos da nossa aplicação abordados no capítulo 2.

Para percebermos a interacção do utilizador com este tipo de aplicações, e baseado na nossa experiência a interagir com as aplicações relacionadas, criamos um diagrama de casos de utilização.

Este diagrama divide os utilizadores na sua relação com listas. De seguida serão abordadas também, as diferentes funcionalidades associadas a cada caso de utilização.

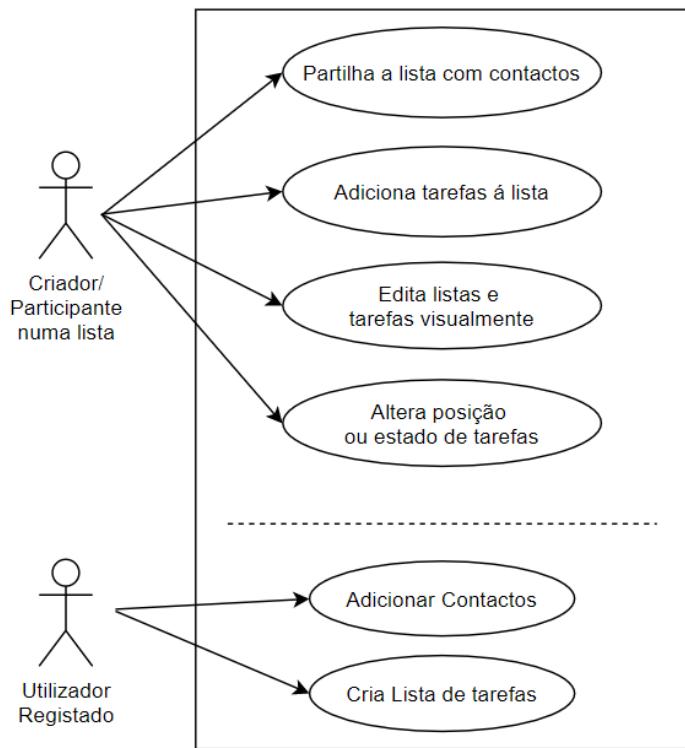


Figura 3.1: Diagrama de casos de utilização

Essencialmente nas aplicações de temática semelhante, o sistema de interacção é relacionado a listas e gestões da mesma, não havendo tarefas sem listas. O papel do utilizador consiste em dar valor a elementos nas listas e alterar a posição destes ou estado.

3.1.2 Funcionalidades do sistema

De modo a comparar as acções do utilizador nos casos de utilização, com as funcionalidades que queríamos ver no sistema, construímos uma tabela onde caracterizamos cada tipo de função que a aplicação dispõe ao utilizador numa categoria.

Classificamos como requisito aquilo que a aplicação não pode perder relativamente ao mercado e como extra aquilo que outras aplicações tenham ou não mas que a nossa tem como função no sistema.

	Função no sistema	Categoria
Utilizador Registado	Login e registo na aplicação	Requisito
	Criar lista de tarefas	Requisito
	Ver tarefas cuja data está proxima	Extra
	Alterar foto de perfil	Requisito
	Alterar nome de perfil	Requisito
	Pesquisa de contactos por nome	Requisito
Criador/ Participante da lista	Criar tarefa	Requisito
	Editar texto da tarefa	Requisito
	Editar cor da tarefa	Extra
	Adicionar imagem a tarefa	Extra
	Completar tarefa	Requisito
	Sinalizar tarefa como prioritarias	Requisito
	Convidar contacto para participar na lista	Requisito
	Adicionar data á tarefa	Requisito
	Movimento vertical de tarefas	Requisito
	Swipe de tarefas	Requisito
	Apagar tarefa	Requisito
	Alterar nome da lista	Extra
	Sincronizar a lista entre pessoas	Requisito
	Alterar cor da lista	Extra
	Adicionar imagem de fundo á lista	Extra
	Chat para os utilizadores de uma lista	Extra
	Ver historico de eventos	Extra
	Ordenar tarefas por prioridade	Requisito

Tabela 3.1: Caracterização das funções do sistema

Na tabela 3.2 caracterizamos os requisitos presentes na tabela 3.1 em obrigatório e desejável de modo a destingir funcionalidades associadas ao funcionamento da aplicação e funcionalidades associadas a experiência do utilizador.

Requisitos do sistema	Categoria
Login e registo na aplicação	Obrigatório
Criar lista de tarefas	Obrigatório
Alterar foto de perfil	Desejável
Alterar nome de perfil	Desejável
Pesquisa de contactos por nome	Desejável
Criar tarefa	Obrigatório
Editar texto da tarefa	Obrigatório
Completar tarefa	Obrigatório
Sinalizar tarefa como prioritarias	Desejável
Convidar contacto para participar na lista	Obrigatório
Adicionar data á tarefa	Desejável
Movimento vertical de tarefas	Obrigatório
Swipe de tarefas	Obrigatório
Apagar tarefa	Obrigatório
Sincronizar a lista entre pessoas	Obrigatório
Ordenar tarefas por prioridade	Desejável

Tabela 3.2: Caracterização dos requisitos da aplicação

3.2 Fundamentos

Sendo o Speedy Manager uma aplicação com o objectivo, acima de tudo, de ser algo prático, eficiente de rápido uso decidimos utilizar as tecnologias seguintes.

3.2.1 Google FireBase

A [Moroney, 2017] Firebase é uma plataforma de serviços na *Cloud* que foi adquirida e por sua vez distribuída pela Google destinada a auxiliar o desenvolvimento de aplicações moveis e para a web.

Os serviços disponibilizados por esta plataforma permitem, a quem a implemente na sua aplicação, funcionalidades tais como um sistema de base de dados em tempo real, sistema de armazenamento de ficheiros (à parte da base de dados), autenticação de utilizadores , *web hosting* utilizando uma rede de fornecimento global, entre outras ferramentas de auxílio ao desenvolvimento e controlo da aplicação tais como análise e monitorização dos utilizadores/dados da aplicação.

A Firebase é, portanto, um serviço muito completo prestado pela Google no auxílio ao desenvolvimento de aplicações em Android e também para a web, e por tal será a plataforma que constitui o *back-end* da nossa aplicação e, portanto, assume o papel de servidor.

Decidimos utilizar do serviço da Google devido ao facto de este ser de fácil integração e concebido para o auxilio em diferentes aspectos do desenvolvimento de uma aplicação móvel. Iremos utilizar as funcionalidades de ”Autentication”para gestão de utilizadores, ”Storage”para guardar todos os ficheiros carregados pelos utilizadores, e ”Real Time Database”para gerir toda a informação alterável em *real-time*, seja esta as tarefas,utilizadores ou listas.



Figura 3.2: Firebase

3.2.2 Aplicação Android

Uma aplicação Android pode ser escrita em Java, Kotlin e linguagens c++. O código é compilado através das ferramentas do Android SDK(software development kit) e gera um ficheiro de sufixo .apk que é um executável para dispositivos Android que permite instalar a aplicação no sistema operativo. Tipicamente quando se cria um projeto de uma aplicação móvel no Android Studio, e executamos o mesmo num emulador, aquilo que é mostrado no ecrã do dispositivo é a representação de um objeto do tipo *Activity*.

O funcionamento base de uma aplicação android assenta sobre o lançamento ou a manutenção destas *Activitys*. As aplicações Android têm esta abordagem de ”actividades”(objetos do tipo *Activity*) para que estas sejam o ponto de entrada para um ”estado” da aplicação, isto significa que uma aplicação pode lançar outra numa determinada actividade com um determinado contexto sem ter que seguir uma ordem prévia.

”Embora as actividades funcionem juntas para formar uma experiência do usuário coesa em um app, cada uma é vagamente vinculada às outras. Geralmente, há dependências mínimas entre as actividades em um app. Na verdade, com frequência elas iniciam actividades pertencentes a outros apps. Por exemplo, um app de navegação pode iniciar a actividade ”Compartilhar” de um app de media social.”

[Android, a]

Um outro ponto constituinte de uma aplicação Android são os recursos da aplicação.

”Os recursos são os arquivos adicionais e o conteúdo estático usado pelo seu código, como bitmaps, definições de layout, strings da interface do usuário, instruções de animação, entre outras coisas.”

[Android, b]

Como linguagem de programação decidimos utilizar o Kotlin com a ambição de tornar o Speedy Manager uma aplicação multi plataforma, pois esta linguagem dispõe funcionalidades que irão facilitar essa implementação

e teria como um extra a familiarização com uma nova linguagem de programação em torno do uso do Java com que já trabalhamos em outras cadeiras.



Figura 3.3: Kotlin

3.2.3 Princípios de um sistema na nuvem

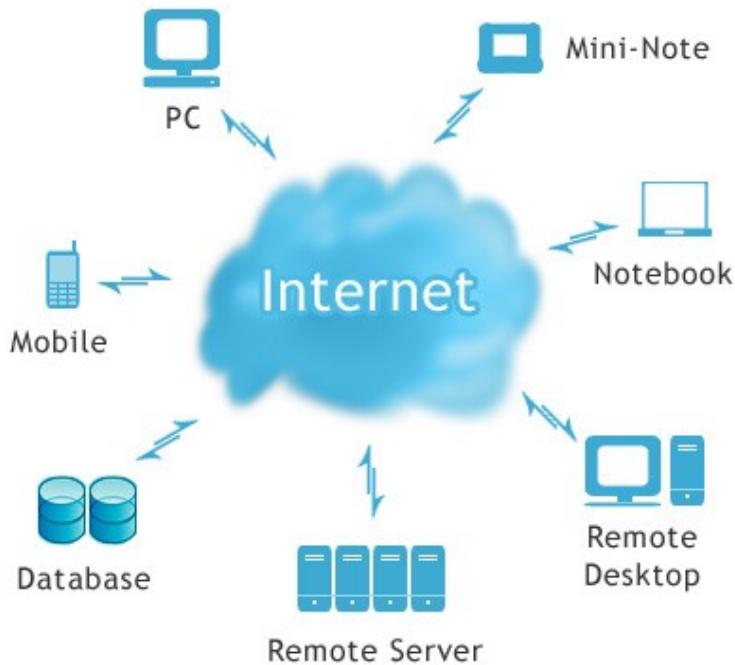


Tabela 3.3: Computação na nuvem

Uma das palavras chave da proposta no nosso projeto é "*Cloud Database*" que essencialmente sugere a utilização de um serviço *Cloud* para a base de dados da aplicação.

Um serviço *Cloud* é tipicamente uma plataforma online com determinadas funcionalidades que está pronta a integrar uma aplicação ou um projecto que vise fazer uso desse mesmo tipo de funcionalidades.

"Simply put, cloud computing is the delivery of computing services—including servers, storage, databases, networking, software, analytics, and intelligence—over the Internet ("the cloud") to offer faster innovation, flexible resources, and economies of scale. You typically pay only for cloud services you use, helping you lower your operating costs, run your infrastructure more efficiently, and scale as your business needs change." [Azure,]

3.2.4 Serviços *Cloud* e aplicações *Cloud Based*

Para a integração de um serviço cloud num projeto/aplicação tipicamente o serviço apresenta uma API que neste contexto é serve como porta de entrada para a utilização do serviço nesse mesmo projeto/aplicação.

Uma Aplicação cloud based é uma aplicação que não só utiliza um ou varios serviços cloud mas tambem depende destes para o seu correto funcionamento. Isto apresenta vantagens e desvantagens, que em suma pode-se dizer que, uma aplicação baseada em serviços cloud é tipicamente mais barata de se produzir dado que parte do software ja está desenvolvido e é computado na internet, mas de contra, é uma aplicação que está dependete de um serviço externo encontrado na internet.



Figura 3.4: exemplos de plantformas/serviços cloud

3.3 Abordagem

Tendo em conta que o Speedy Manager tem como principal objectivo ser uma aplicação que permite uma rápida e intuitiva gestão de tarefas e de listas de tarefas e a partilha destas rapidamente começamos a pensar nas tecnologias que mais facilmente iriam permitir alcançar esse objetivo.

Para tal tivemos em consideração as linguagens de programação Java e Kotlin para desenvolver a nossa aplicação android bem como algumas opções de tecnologias para serem utilizadas como base de dados tais como o Firebase e o SQL. Após pesquisa sobre ambos os temas chegámos à conclusão de que Kotlin seria a linguagem mais vantajosa devido a esta ser concebida para o desenvolvimento de aplicações moveis, bem como o Firebase devido à sua capacidade de funcionamento em real-time utilizando a sua funcionalidade “Real-time database”, o qual seria benéfico para para a sincronização entre os vários utilizadores.

Após termos decidido que tecnologias utilizar começamos a realizar esboços em papel de qual seria o aspecto da nossa aplicação, ou seja, as *wireframes*, sendo estas uma representação técnica de cada ecrã da nossa aplicação Android. No fim de concluir estes *wireframes*, utilizando a ferramenta Figma implementamos esses *layouts* e criamos uma demonstração (uma demo) da aplicação a qual foi enviada para alguns utilizadores para ser testada, bem como um questionário sobre melhorias a fazer, pontos fortes ou outras observações de utilizadores que achassem pertinentes.

No inicio do desenvolvimento começamos a implementar as principais mecânicas da aplicação. Num modo simplificado, começamos por implementar um sistema de *login*, com *email* e *password*, ou como com a conta da Google. Isto de modo a conseguir já distinguir os diversos utilizadores da nossa aplicação.

Achamos este um ponto de partida inteligente pois a aplicação funciona em torno de listas de tarefas que cada utilizador criou ou é parte de.

Seguidamente começamos a testar ferramentas Android que permitissem simultaneamente os comando de *drag* e *swipe*. Esta exploração de ferramentas premitio chegar à utilização de RecyclersViews com uso de ItemTouchHelpers para suportar as funcionalidades de *drag and drop* bem como as funcionalidades de *swipe* associados a elementos visuais. Estes testes foram feitos utilizando interfaces muito básicas sem grande detalhe estético

utilizando recursos XML para defini-las visualmente.

Na seguinte etapa fomos avaliar outras opções para implementar as interfaces gráficas referidas no paragrafo acima, as quais encontramos o Jetpack que oferece a ferramenta “Compose” para a fácil e estética criação de interfaces gráficas e começamos a desenvolver os *wireframes* utilizando esta nova ferramenta, o que foi trabalho em vão pois após uma investigação mais profunda reparamos que o Jetpack não oferece ferramentas equivalentes às dos RecyclerViews e ao ItemTouchHelper , os quais são duas das principais ferramentas da nossa aplicação pois vão ser estas as responsáveis pelo mecanismo de comportamento das tarefas dentro de uma lista, e no fim decidimos voltar à abordagem utilizando o Kotlin nativamente.

Posteriormente à implementação das *wireframes* decidimos avançar para a realização das *mockups* da nossa aplicação, novamente utilizando a ferramenta Figma, com a qual realizamos uma “demo” que simulava a utilização da aplicação, enviamos essa demo bem como mais um questionário sobre a utilização desta de modo a obter algum *feedback* para melhorar/implementar funcionalidades que os utilizadores tivessem achado pertinentes.

Tendo em conta os resultados dos questionários, passámos à implementação destes *mockups*, utilizando o XML para definir os *layouts*, na nossa aplicação de modo a já ter uma versão esteticamente apelativa desta.

Após esta implementação dos *mockups* chegou a fase de implementar as funcionalidades principais de um modo mais específico, nomeadamente a criação de listas de tarefas, associar imagens a listas ou a tarefas, mudar a cor da lista, editar textos, apagar tarefas, completar tarefas, alterar a data-limite de uma tarefa entre outras, de modo a cumprir todas as funcionalidades anunciadas anteriormente.

Quando a mecânica de funcionamento das listas de tarefas estava completa passamos à fase de criar um sistema de contactos onde o utilizador pode adicionar um contacto aos seus amigos, e posteriormente se quiser, adicioná-lo a uma lista, isto de modo a permitir a partilha de listas de tarefas entre diversos utilizadores.

Com este sistema implementado conseguimos então passar à fase onde testámos a funcionalidade em *real-time* da nossa aplicação, criando uma lista e convidando um utilizador conseguimos testar o funcionamento de uma lista para dois utilizadores em simultâneo e corrigir possíveis erros nesta fase.

Por fim implementamos as funcionalidades de carácter menos principal da nossa aplicação, nomeadamente um *chat* por lista, onde os utilizadores dentro de uma lista podem trocar mensagens uns com os outros, bem como uma *dashboard* na qual são apresentados os eventos que aconteceram de maneira mais recente com uns gráficos referentes a quem participa mais ou a quem completa mais tarefas. Implementamos também na página inicial um *display* que mostra as três ultimas listas visitadas pelo utilizador, e ainda uma área onde mostra as tarefas, de qualquer lista, que estão perto de expirar ou que a data limite está a chegar, de modo a dar jus ao nome “Speedy Manager”.

Capítulo 4

Implementação do Modelo

O desenvolvimento da nossa aplicação distingue-se essencialmente em três fases, a análise, o design e implementação. Junto ao fim da fase de implementação decidimos fazer uma nova iteração sobre os modelos obtidos na fase de design de modo obtermos uma interface superior feita também com base na experiência obtida no desenvolvimento e realização até este ponto.

4.1 Análise Inicial

Esta fase data o inicio do desenvolvimento do projecto no qual faz parte o Trabalho Relacionado(capítulo 2.2), onde através do estudo de mercado analisámos aplicações parecidas ao que o SpeedyManager deveria ser.

Esse estudo permitiu também para deferir quais os requisitos e casos de utilização da aplicação abordados no capítulo 3(capítulo 3.1).

Neste ponto de desenvolvimento começámos também por integrar projectos Android na consola do Firebase, os quais serviram para testarmos sistemas de registo, login e autenticação. Um destes projectos veio a tornar-se no produto final.

Fomos ainda investigar outras opções para implementação de uma interface gráfica e ficamos divididos entre o uso de XML e do Jetpack compose. O Jetpack compose apresentou ser uma opção mais prática e de uso mais simples devido à sua natureza declarativa porém após investigação descobrimos que não era suportado ainda um mecanismo semelhante ao recycler view com ItemTouchHelper como é com uso do XML o que nos levou a optar por utilizar o XML para a criação da interface gráfica.

4.2 Design

Para a criação de uma aplicação móvel de gestão de tarefas e listas de tarefas apresentado uma interface gráfica simples e apelativa é necessário várias iterações sobre diferentes modelos gráficos para analisarmos quais deles se mostram mais eficientes e ao mesmo tempo apelativos ao utilizador.

4.2.1 Realização de wireframes

Numa fase inicial da resolução deste projecto começamos por perceber o contexto da aplicação que iríamos desenvolver. Para tal exploramos aplicações que abordam o mesmo contexto/temática para não só adquirirmos a experiência de utilização, mas também, para fazer um estudo de mercado para ver o que este oferece no âmbito de aplicações para gestão de listas de tarefas assim como analisado no capítulo 2

Este processo de desenho dos *wireframes* consistiu na discussão de ideias entre vários intervenientes de modo a ser produzido um esquema visual básico contendo os requisitos e funcionalidades da aplicação.

As figuras seguintes mostram o esboço inicial da aplicação.



Figura 4.1: Wireframes Ecrãs de login e registo



Figura 4.2: Wireframes Ecrãs principais bottom bar navigation



Figura 4.3: Wireframes Ecrãs dentro de uma lista

4.2.2 Realização de Mockups

Após a construção dos *wireframes* recorremos à criação de uma breve ”demo”, utilizando a ferramenta Figma de modo poder ser enviada para utilizadores para estes a testarem, foi ainda enviado também um questionário de usabilidade com o objectivo de obter algum *feedback* da demonstração e

alguns pontos que os utilizadores tivessem achado de interesse a melhorar.

Com base nas *wireframes* e nas respostas dos questionários passámos à fase de desenvolvimento dos *mockups*, sendo que estas já são uma versão mais detalhada e muito próximo de como a aplicação irá parecer, de um ponto de vista estético.

Realizamos também de novo uma demonstração para ser enviada com um questionário de usabilidade de modo a obter *feedback* e poder melhorar eventuais pontos que os utilizadores tenham achado mais fracos.

Nas figuras seguintes são apresentados os *mockups* desenvolvidos.

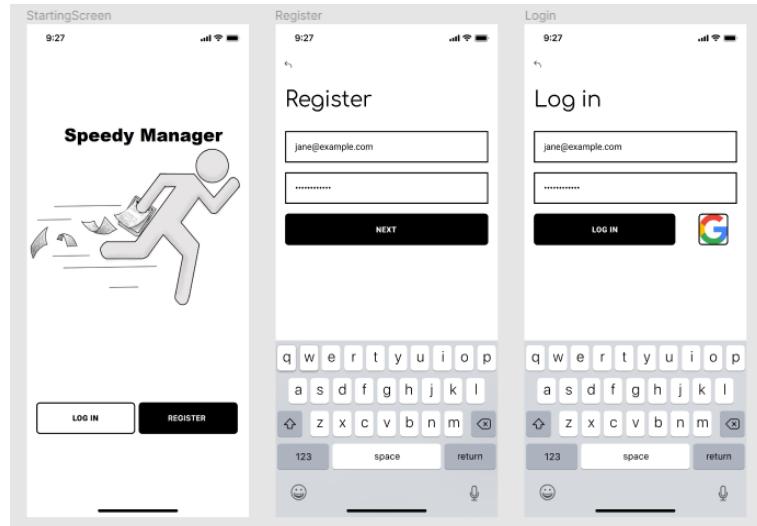


Figura 4.4: MockUps Ecrãs de login e registo

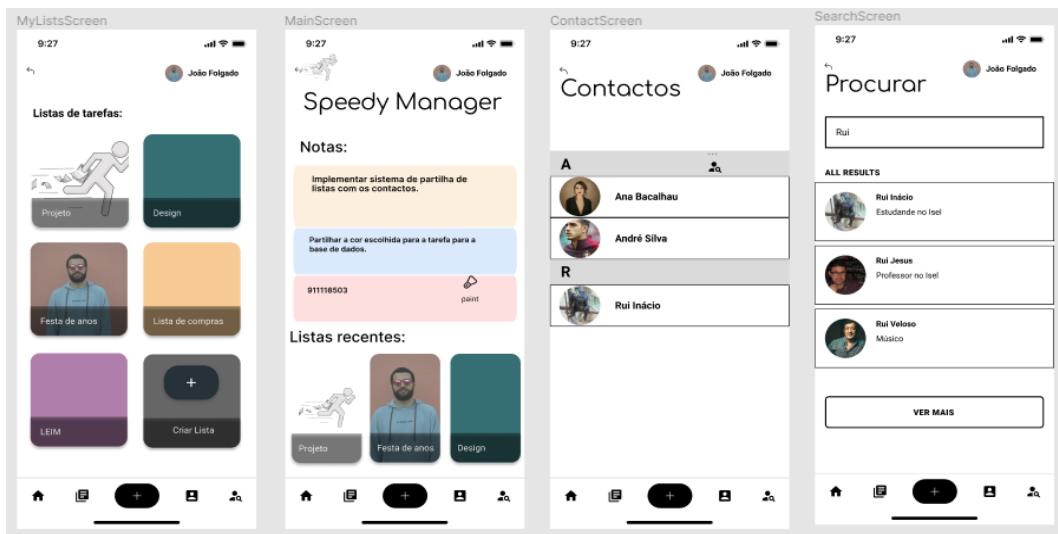


Figura 4.5: MockUps Ecrãs principais bottom bar navigation

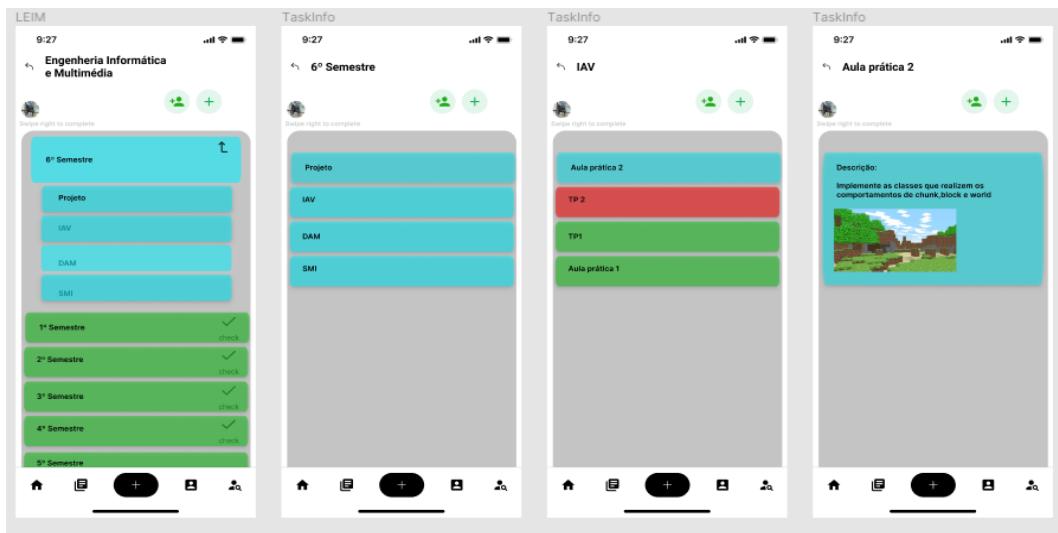


Figura 4.6: MockUps Ecrãs dentro de uma lista

4.2.3 Obtenção de feedback para Design proposto

Para avaliarmos as ideias/conceitos relacionados com projecto bem como este progresso no desenvolvimento de modelos visuais para a representação das mesmas, criamos um questionário com 10 perguntas que achámos pertinentes colocar ao público que à partida sabíamos que iria responder.

De seguida apresentamos a descrição presente no questionário bem como as duas perguntas iniciais.

Link para o questionário:

<https://docs.google.com/forms/d/1v09wkguetqlYwzhXG3oQkmllLpPiD1-2cGsBedit>

Projeto: SpeedyManager

O SpeedyManager é uma aplicação para o telemóvel que permite aos seus utilizadores criarem e organizarem tarefas num género de uma to-do-list. O manuseamento de tarefas na aplicação é feito através do drag and drop destas tarefas numa lista conforme a vontade do utilizador. As tarefas integram diferentes tipos de funcionalidades tais como detalhes, calendário, upload e download de ficheiros/fotos, votações e ainda sub-tarefas. As listas de tarefas podem ser partilhadas com diferentes utilizadores que terão acesso a essa mesma lista sincronizada em tempo real com alterações de qualquer membro.

Conhece/ já utilizou aplicações como o “Google Task”, “Microsoft To Do”, “Trello” ou outras aplicações do género de uma To-Do-List? *

Sim

Não

Se possui alguma experiência com este tipo de aplicações o que considera de importante/inovador na aplicação que usa/usou ou experimentou?

Sua resposta

Figura 4.7: Exemplo de perguntas realizadas

4.2.4 Analise dos resultados obtidos no questionário

Seleccionamos algumas das respostas obtidas nos questionários, que revelam a opinião dos utilizadores não só à ideia da nossa aplicação, mas, também à sua vertente estética e funcionalidades. As respostas seleccionadas podem ser observadas nas figuras seguintes.

Conhece/ já utilizou aplicações como o “Google Task”, “Microsoft To Do”, “Trello” ou outras aplicações do género de uma To-Do-List?

11 respostas

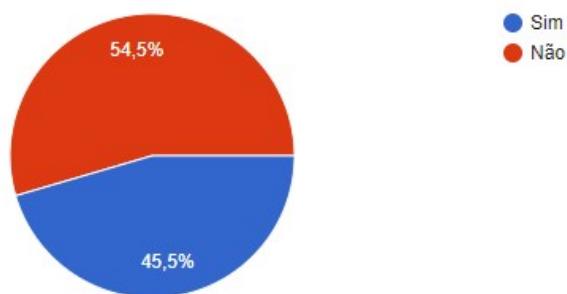


Figura 4.8: Estudo da população questionado

Considera um sistema de mensagens publico para todos os participantes de um lobby algo positivo/útil?

11 respostas

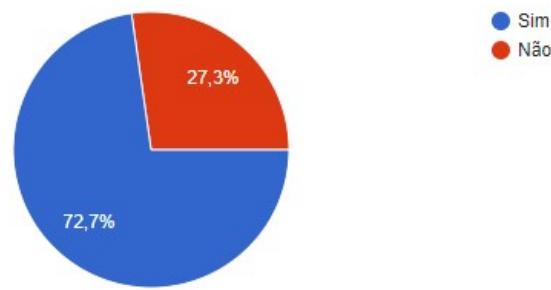


Figura 4.9: Respostas á presença de chat numa lista

As principais conclusões a retirar dos resultados questionários são essencialmente que a seria útil a implementação de um sistema de *chat* dentro

de cada lista onde os participantes da mesma conseguissem comunicar entre si bem como tentar melhorar um pouco a vertente estética de modo a esta ser mais simples porém mais apelativa.

Tivemos também em conta algumas funcionalidades adicionais de modo a melhorar o funcionamento bem como o *QOS (Quality of Service)* da nossa aplicação para melhorar a experiência do utilizador a usar a nossa aplicação em termos de interactividade com o utilizador e navegação da dentro da aplicação.

4.3 Desenvolvimento

Nesta secção cada subsecção seguinte contem o desenvolvimento mais detalhado dos diferentes aspectos que compõem a aplicação bem como serão apresentadas imagens das diferentes funcionalidades.

4.3.1 Criação de tarefas

Na nossa aplicação dispomos de dois tipos diferentes de tarefas, as tarefas simples e as tarefas compostas, o tipo de tarefa a adicionar pode ser escolhido pelo slider presente na figura seguinte ao criar uma nova tarefa.

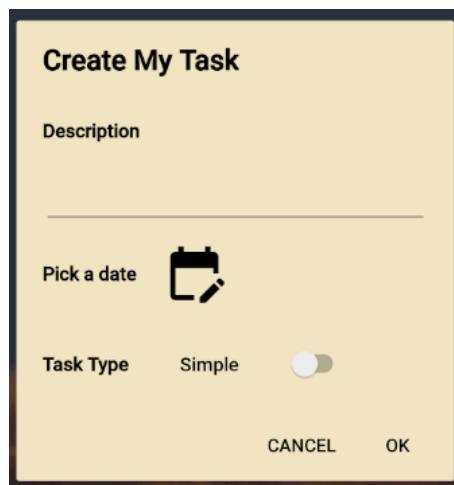


Figura 4.10: Popup criação de uma nova tarefa

Caso a tarefa criada seja do tipo simples esta será uma tarefa singular com apenas um texto, data, cor, prioritária ou não prioritária e pode ou não ter imagens.

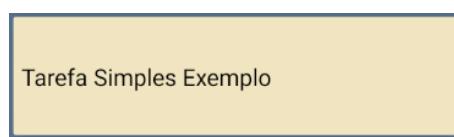


Figura 4.11: Exemplo de tarefa simples

Caso a tarefa criada seja do tipo composta, esta dispõem de todas as funcionalidades da tarefa simples e ainda dispõem de uma lista de uma ou mais sub tarefas.



Figura 4.12: Exemplo de tarefa composta

Programaticamente uma tarefa é uma View construída a partir de um recurso que a desenha, onde é associado toda a informação que esta contem. Esta View é depois acoplada a um RecyclerView e tal será abordado na subsecção seguinte.

4.3.2 Construção de listas de tarefas

A construção e visualização das listas de tarefas foi feita com base na ferramenta RecyclerView. Esta ferramenta permite associar a Views comandos de *drag and drop* bem como comandos de *swipe*.

Para a construção da representação das tarefas utilizamos um recurso XML para definir o formato e todos os campos que esta deve conter. Para a visualização das tarefas estas podem ser "instanciadas" num elemento de actividade ou fragmento ou ainda um contentor de Views que as organize na vertical horizontal ou em grelha. Aquando uma View é inflacionada para a interface gráfica os diferentes campos são preenchidos com a informação encontrada na base de dados associados a essa determinada tarefa.

Utilização dos Comandos *Swipe* , *Drag and Drop*

Um dos requisitos desta aplicação é a utilização dos comandos *drag and drop* para a alteração da ordem das tarefas na lista. Existem varias formas de reproduzir este comportamento num sistema Android, no entanto, nós decidimos também fazer uso do comando *swipe* e associa-lo à edição e ao processo de completar uma tarefa. Nisto surge a necessidade de uma ferramenta capaz de incorporar estes dois tipos de comandos numa lista de

Views pronta a ser populada em tempo real. A figura seguinte demonstra um drag associado a uma tarefa.

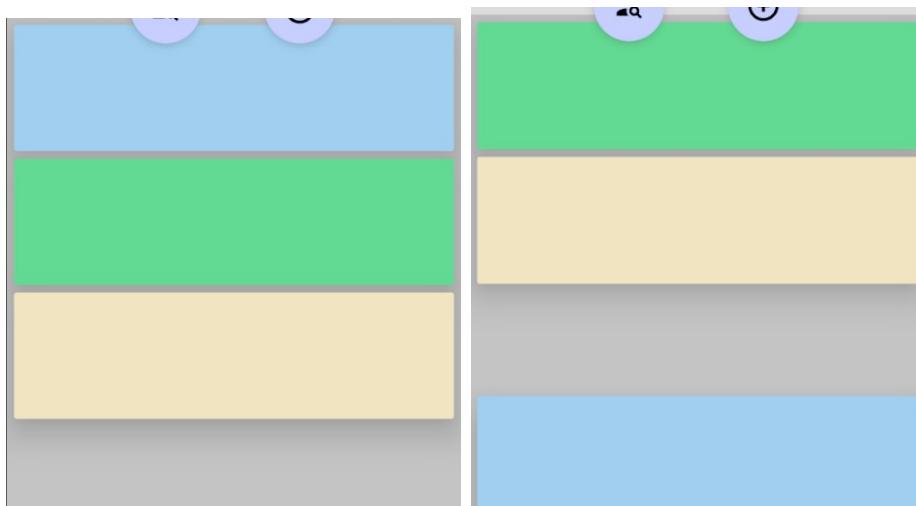


Figura 4.13: Comando Swipe associado a uma tarefa

Devido ao facto o RecyclerView ser essencialmente um *container* para itens(Views) já com as funcionalidades de movimentos por arrasto prontras a serem activas num dos seus constituintes o RecyclerAdapter, e por possuir um suporte para um ItemTouchHelper que é um objecto também nativo do Android que permite incorporar um mecanismo de *swipe* a itens de um RecyclerView, utilizámos este extensivamente nos diferentes algoritmos que implementam a interface e os comandos da mesma.

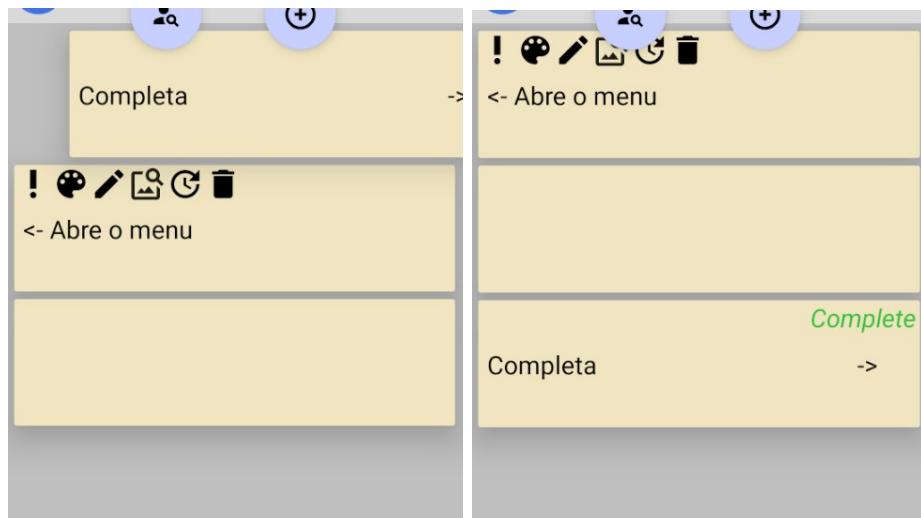


Figura 4.14: Comando Swipe associado a uma tarefa

Utilização das Snapshots

Como referido anteriormente a utilização de Snapshots implementam o sincronismo na troca de informação entre o utilizador e base de dados. Além disto estas são também responsáveis por comunicar alterações relativas à parte visual da aplicação. Por tal tivemos em atenção ao funcionamento destas em conjunto com o resto das funcionalidades da aplicação por desenvolver. Isto aplica-se na medida em que é necessário que funcionalidades que invoquem alterações na base de dados não interajam entre si causando falhas ou pontos de ruptura no sistema.

4.3.3 Desenvolvimento de um protótipo

Após a criação dos diferentes sistemas que compõem os elementos básicos das tarefas, listas de tarefas e utilizadores, obtemos um protótipo funcional da aplicação que essencialmente consiste num sistema caixas de texto que podem ser editadas ou movidas no ecrã para cima e para baixo através do comando *drag* por vários utilizadores.

Os algoritmos desenvolvidos começam por agrupar a informação relativa às listas de tarefas para que esta informação seja então ordenada e disposta conforme a vontade dos utilizadores e para que o motor base da aplicação fosse independente da interface gráfica que se lhe aplica.

Começamos também por desenvolver os mecanismos de troca de informação com a Firebase para termos uma ideia de como transferir e manipular a informação entre diferentes dispositivos sobre uma interface bastante rudimentar.

Estes mecanismos são algoritmos que ordenam e classificam a informação da base de dados, a título de exemplo, um algoritmo que ordena uma lista de tarefa pelo seu estado. Os mesmos são feitos recorrendo a objectos DataSnapshot que em suma implementam sistema de troca de informação da Firebase que permitem sincronismo de informação entre vários dispositivos.

É importante compor este tipo de métodos numa fase inicial pois nas fases seguintes servem de base para outros pontos da aplicação bem como mecanismos de teste.

Esta abordagem de foco no motor base de troca de informação na aplicação, numa fase inicial, permite chegar a um protótipo também num ponto inicial do desenvolvimento e que contribui para uma agilização no desenvolvimento da interface e outras funcionalidades futuras. A figura seguinte demonstra um protótipo da aplicação a exemplificar um movimento numa lista partilhada.

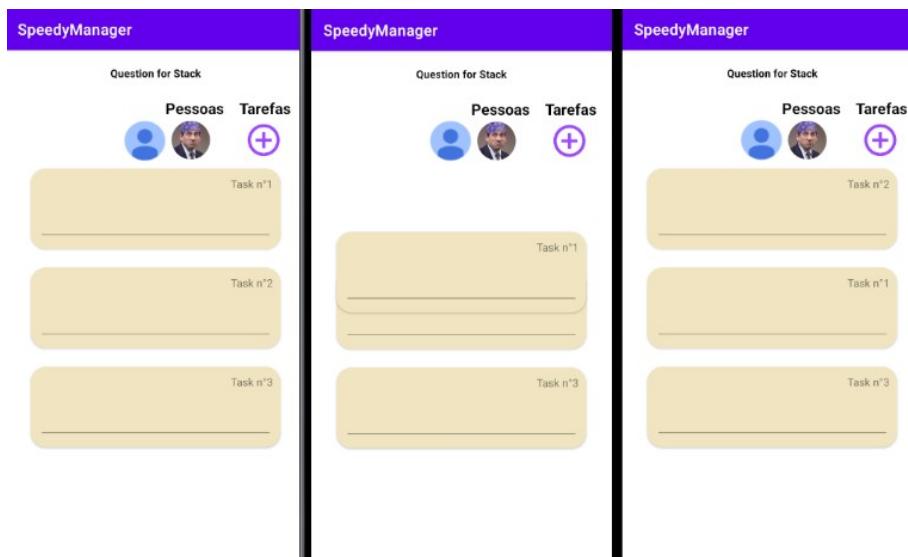


Figura 4.15: Modelo inicial da aplicação

4.4 Implementação da navegação e funcionalidades

Após o desenvolvimento do protótipo abordado na secção anterior implementamos as diferentes funcionalidades que constituem a aplicação. Assim como na secção anterior, as subsecções seguintes descrevem a implementação de determinada funcionalidade.

4.4.1 Navegação entre fragmentos

Com objectivo de manter uma navegação fácil e rápida em qualquer ecrã/página da nossa aplicação decidimos manter um menu na parte inferior do ecrã através do qual o utilizador rapidamente consegue aceder aos diferentes ecrãs principais.

Esse menu possui cinco botões, quatro deles navegavam para ecrãs principais/importantes de modo a manter a vertente ”speedy” da nossa aplicação.

No ecrã inicial o utilizador terá acesso às listas recentemente aceidadas bem como tarefas, de qualquer lista, que estejam quase a expirar ou já tenham expirado. O ecrã dois mostra todas as listas criadas por este utilizador ou às quais ele pertence. Existe também um ecrã com os contactos adicionados e onde se pode adicionar novos contactos, e ainda, outro para o perfil de utilizador onde se pode editar informação como o seu *username*, imagem de perfil ou realizar o *logout*.



Figura 4.16: Menu inferior de navegação

Os quatro botões de navegação referidos no paragrafo anterior são, respectivamente, os dois da esquerda e os dois da direita.

Existe ainda um botão no centro com o símbolo de um ”+”, este altera a sua função dependendo o ecrã ou fragmento em que se encontra, caso o fragmento actual seja a página inicial, este botão irá abrir o fragmento onde mostra todas as listas do utilizador, caso o fragmento actual seja o que mostra as listas do utilizador este botão tem a funcionalidade de criar uma lista nova, no caso de actualmente estar aberto o fragmento dos contactos este

botão abre um novo fragmento onde o utilizador poderá pesquisar outros utilizadores (não adicionados) por nome e adiciona-los aos contactos caso assim o deseje, por fim, caso o fragmento actual seja dentro de uma lista de tarefas esse botão tem a funcionalidade de criar uma tarefa nova.

4.4.2 Tarefas urgentes e listas recentes

Para uma melhor navegação na aplicação, no ecrã inicial o utilizar dispõe de um histórico das ultimas três listas que acedeu, e um sistema de tarefas urgentes que filtra todas as tarefas de todas as listas que este pertence por tarefas cuja data esteja próxima de expirar ou já tenha expirado e a tarefa continue por completar. A figura seguinte demonstra o ecrã inicial da aplicação.



Figura 4.17: Ecrã inicial

Este desenvolvimento destinge o ecrã inicial do ecrã onde se encontram as listas a que utilizador pertence. Isto permite ao utilizador uma

melhor gerência das suas listas de tarefas, pois caso este possua diversas listas, é importante ter um sistema que filtre tarefas pela sua importância temporal para que este não tenha que andar á procura das mesmas.

Este sistema permite ao utilizador navegar a partir do ecrã principal para as listas que tiveram um acesso recente bem como navegar para as listas que possuam uma tarefa anunciada nas tarefas urgentes, facilitando a navegação e a organização.

4.4.3 Menu das tarefas

Para gestão de vários campos/opções de uma tarefa decidimos criar um menu para estas onde o utilizador pode mudar a cor, alterar o texto, meter imagem, entre outras opções.



Figura 4.18: Tarefa simples com menu fechado

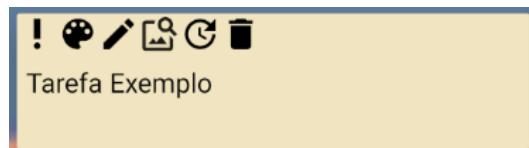


Figura 4.19: Tarefa simples com menu aberto

Podemos observar também pela figura anterior e pela figura seguinte que em ambas as tarefas simples e tarefas compostas o menu é igual.



Figura 4.20: Tarefa composta com menu aberto

Como podemos observar pelas imagens anteriores as tarefas dispõem de um menu com 6 botões, o qual se torna visível após um *swipe left*, estes botões tem as funcionalidades de, respectivamente, da esquerda para a direita, tornar esta tarefa uma tarefa prioritária, mudar a cor de fundo da tarefa, alterar/editar o texto da tarefa, adicionar uma imagem à tarefa, mudar/adicionar uma data limite à tarefa e por fim eliminar tarefa.

No caso das tarefas compostas temos ainda as sub tarefas que dispõem de sub-menu próprio no qual podem, alterar o seu texto, adicionar imagem ou eliminar a sub tarefa.

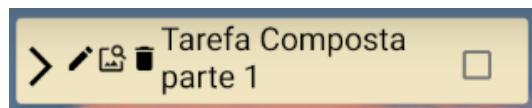


Figura 4.21: Menu da sub-tarefa

As sub tarefas dispõem ainda de uma *check box* do lado direito para definir o seu estado como completas ou não, as quais vão contribuir para o estado da tarefa composta ”pai”, o qual podemos observar no canto inferior esquerdo da tarefa composta pai na figura seguinte.



Figura 4.22: Progresso de uma tarefa composta

O contador de progresso indica ”2/3” pois das sub tarefas tem 2 de 3 completadas ou indicadas com os check marks.

Por opção decidimos fazer o menu de uma determinada tarefa um processo local ao utilizador, isto para que a visualização da lista para diferentes utilizadores não contenha a informação do estado de abertura do menu de cada tarefa.

Para tal junto com a informação de cada tarefa nos *recyclers* é necessário guardar também um booleano do estado do menu. No entanto, devido ao facto de esta informação de cada tarefa poder ser alterada e reposicionada a qualquer momento por qualquer outro utilizador é necessário que o sistema saiba destingir repositionamentos de eliminações de tarefas. Sempre que uma tarefa é eliminada da lista é necessário manter o estado dos menus que estavam em aberto e de modo a manter estes abertos não alterando a visualização local do utilizador.

4.4.4 Tarefas com imagens

Para que as tarefas suportem imagens é necessário criar uma sistema de *upload* de imagens para uma determinada tarefa e guardar informação da mesma.

Para guardarmos a informação de cada imagem utilizamos o sistema de Storage da Firebase que é essencialmente um sistema de armazenamento de ficheiros.

Ao realizar o *upload* de uma imagem para o Firebase Storage poderíamos acede-la utilizando o seu link referente à base de dados o que funcionaria mas teria um tempo de carregamento que não seria o ideal para o que pretendíamos com funcionamento em *real-time*. Para contornar esse problema reparámos que sempre se realizava o upload de uma imagem esta também ficava associada a um URL, que é o URL de onde, na base de dados da Google, essa imagem ficou guardada, guardando esse URL ao invés do URL referente à nossa base de dados local torna o download e visualização das imagens um processo instantâneo o que é o pretendido na nossa aplicação.

Por fim tanto o URL da imagem, bem como a restante informação da tarefa são guardadas na *Real-Time Database* para que mais tarde voltarem a ser visualizadas.

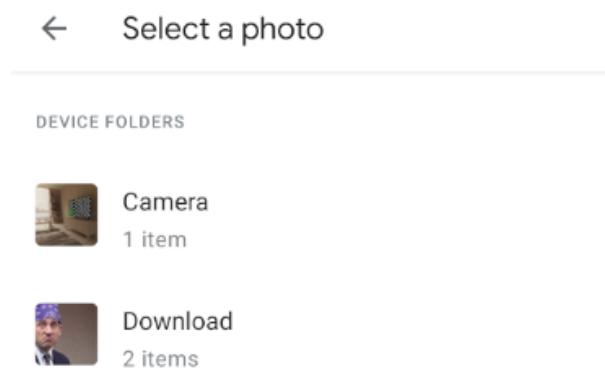


Figura 4.23: Escolha da imagem

Para o sistema de selecção de imagem dos ficheiros do telefone o sistema de Android possui um conjunto de operações(Intents) pré-definidas

que permitem logo a navegação e a selecção de uma imagem da galeria. Após a selecção da imagem na galeria regressamos á actividade da lista de tarefas, onde a informação da imagem escolhida é enviada para o Storage no qual é gerado um URL da mesma que é armazenado junto com a informação da lista na RealTime Database.



Figura 4.24: Tarefa com imagem

Para a visualização das imagens provindas da base de dados sob a forma de um url uilizamos uma biblioteca o Glide [Glide,] que faz *download* e mostra dinamicamente as imagens. Isto é importante para que a restante informação na lista não tenha que esperar pelo *download* das imagens.

4.4.5 Sistema de procura por contactos

Implementamos um sistema de pesquisa por nome de utilizador de modo a permitir uma rápida ligação entre dois utilizadores.

Este sistema funciona através da implementação da interface **Filtrable** de modo a definir um comportamento de filtro na nossa search view a qual actualiza os resultados após cada input pelo utilizador (neste caso letra).

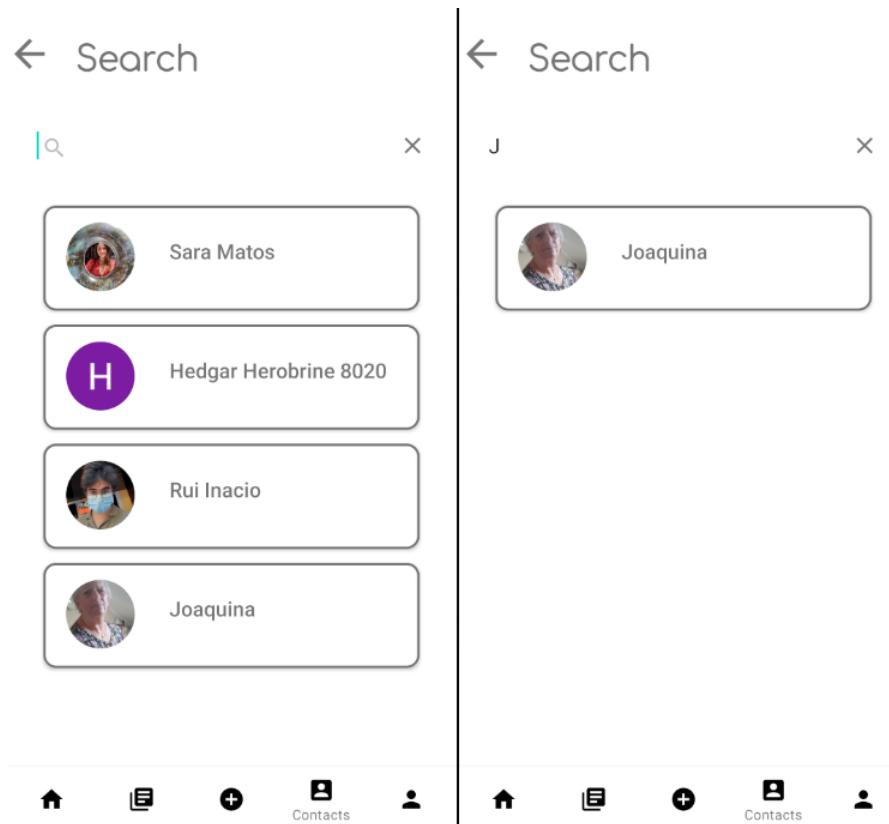


Figura 4.25: Exemplo de pesquisa

4.4.6 Partilha de listas entre contactos

Para partilhar uma lista de tarefas entre utilizadores é necessário que estes sejam contactos um do outro.

O seguinte exemplo exemplifica demonstrar convidar o utilizador "Sara Matos" para a lista e de seguida ela presente nos participantes da lista actual "Relatório".

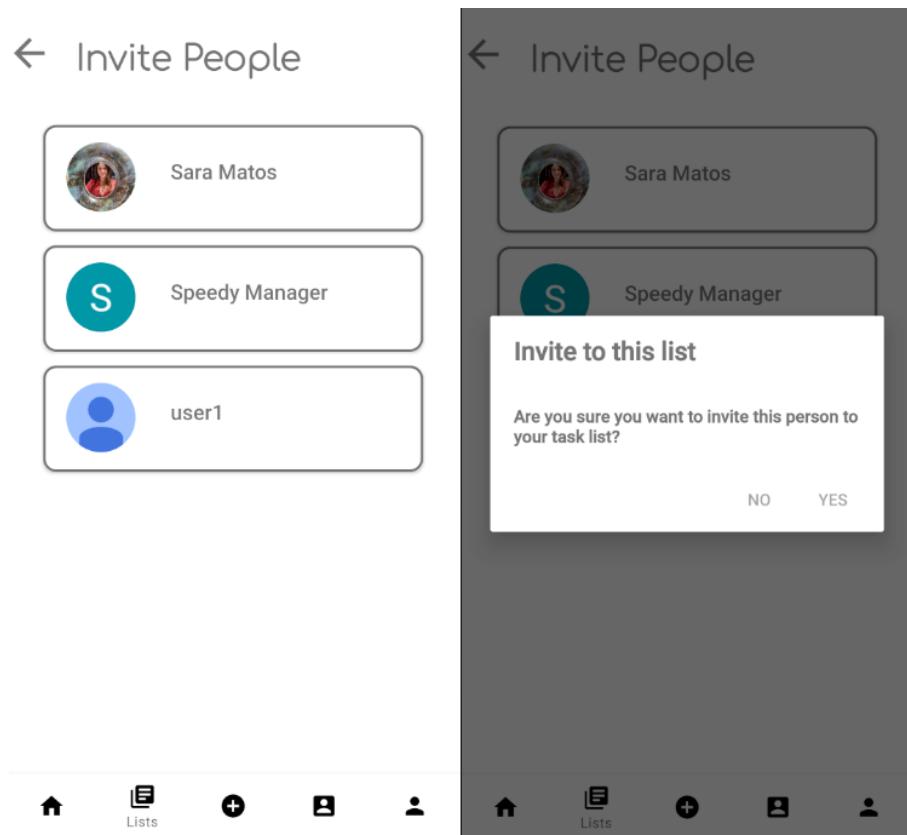


Figura 4.26: Processo de convidar um utilizador para uma lista

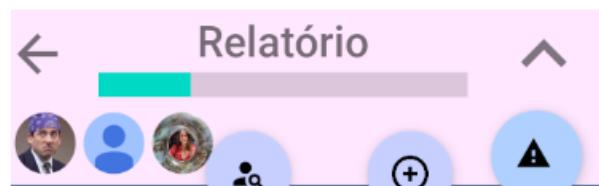


Figura 4.27: Utilizadores da lista

4.4.7 Tarefas prioritárias

Para um controlo superior de tarefas numa determinada lista, o utilizador, pode sinalizar esta como importante.

Uma tarefa importante será aquela que, independentemente da data, se considere uma tarefa prioritária a todas as outras, para tal o menu das tarefas dispõe de um botão com o símbolo “!” para sinalizar a respectiva tarefa como prioritária. Para tal foi necessário implementar um estado adicional nas tarefas para que esta possua uma distinção de importância das que não são sinalizadas como tal.



Figura 4.28: Exemplo de um tarefa prioritária

As tarefas sinalizadas de prioritárias passam a ter no seu canto superior direito um triângulo vermelho, funcionando não só para feitos de distinção entre tarefas prioritárias e não prioritárias, mas também como um símbolo de alerta, daí a natureza da sua cor vermelha que sobressai em relação ao resto.

As listas dispõem ainda de um menu que possui um botão para ordenar a lista de tarefas colocando sempre em primeiro, ou seja, no topo do ecrã, as tarefas que foram sinalizadas como prioritárias. Esse botão, como podemos observar na figura seguinte, é o que dispõe de um símbolo de um triângulo com um ponto de exclamação no meio.

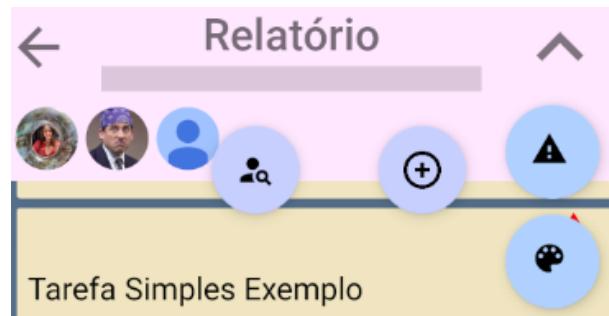


Figura 4.29: Exemplo de uma tarefa prioritária

Em relação a datas associadas a tarefas, aquando da criação ou edição via menu de uma tarefa, implementamos um sistema de escolha de data que recorre a um calendário familiar, suportado pelo sistema Android ,que permite ao utilizador escolher uma data a associa-la a determinada tarefa. Para que o utilizador tenha uma ideia de todas as tarefas, cuja data atribuída se está a aproximar, pertencentes a todas as listas que este está inserido, na pagina inicial construímos um sistema de tarefas urgentes. Este sistema procura em todas as listas que o utilizador está inserido por tarefas cuja data associada esteja suficientemente próxima da data presente num limiar de uma semana.

4.4.8 Chat

Para integrar um sistema de chat na aplicação criamos uma fragmento que implementa um *layout* adequado que permite aos utilizadores numa determinada lista comunicarem livremente entre si.

O desenvolvimento desta funcionalidade foi também feito com recurso à Firebase onde registamos as mensagens enviadas pelos diferentes utilizadores junto com a data e hora de envio da mesma. A figura seguinte demonstra o *chat* de uma lista.



Figura 4.30: Exemplo de uma conversa no chat de uma lista

4.4.9 Monitorização da lista

Para a monitorização do que vai acontecendo a lista durante o tempo decidimos implementar um sistema de eventos que guardam a informação dos diferentes tipos de alteração que vão sendo feitas na lista ao longo do tempo. Esta informação é depois utilizada para construir um histórico de acontecimentos bem como construção de gráficos de participação e produtividade que permitem a rápida visualização da participação de todos.

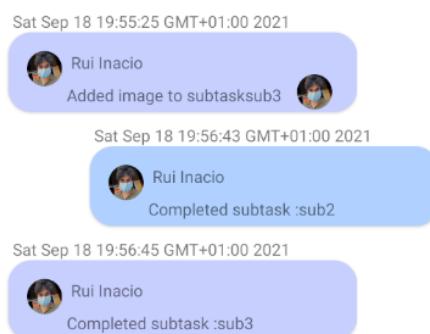


Figura 4.31: Exemplo de um histórico de alterações numa lista

O gráfico de participação contabiliza o numero de alterações que cada utilizador faz.

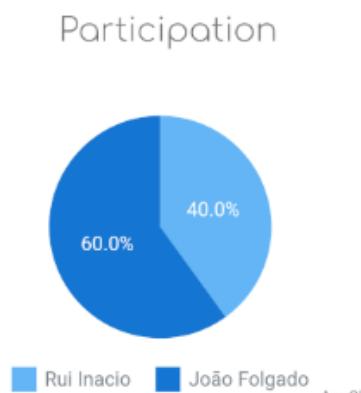


Figura 4.32: Gráfico de participação

O gráfico de produtividade contabiliza o numero de tarefas completas por cada utilizador.

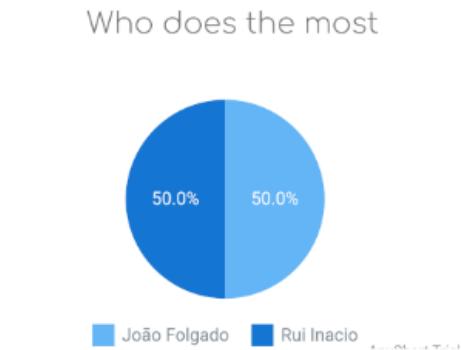


Figura 4.33: Gráfico de quem completa mais tarefas

Adicionalmente implementamos um gráfico que contabiliza o tipo de alteração que mais é efectuada na lista, uma estatística importante que ajuda na monitorização de vários eventos ao longo do tempo.

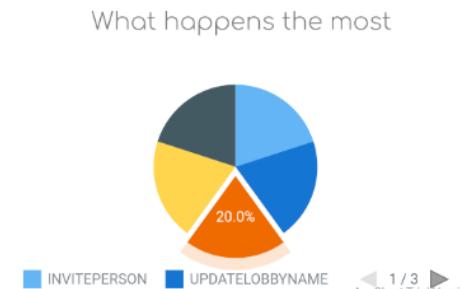


Figura 4.34: Gráfico de eventos

Para a criação deste gráficos foi utilizada a biblioteca AnyChart[AnyChart,] que disponibiliza uma Api para o Android[AnyChartAndroid,]. A selecção desta biblioteca foi apenas devido á simplicidade de implementação de gráficos de estilo "pizza"que pretendíamos.

4.4.10 Menu da lista

De modo a permitir uma gestão não só de tarefas mas também da lista em si adicionamos um menu que permite a gestão da informação desta, seja para mudar o nome, mudar a cor, associar imagem, entre outros.

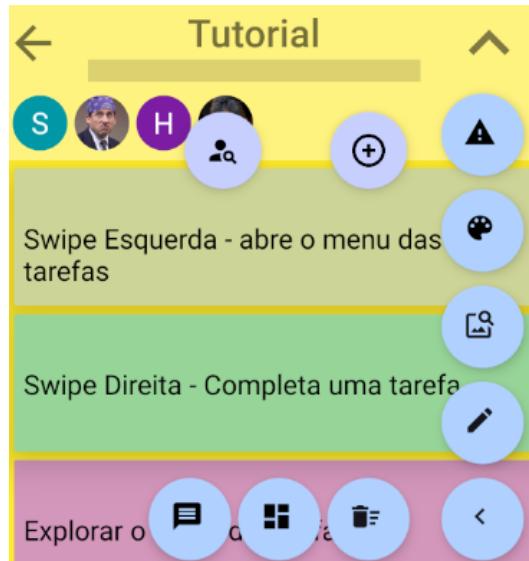


Figura 4.35: Menu de uma lista

Como podemos observar pela imagem acima, existem vários botões com um fundo azul claro do lado direito do ecrã, esses botões fazem parte do menu da lista, esse menu pode ser ocultado de modo a não ocupar espaço no ecrã.

Cada botão tem a sua funcionalidade, sendo esta, respectivamente de baixo para cima, ordenar a lista de modo a meter tarefas sinalizadas como prioritárias em primeiro, mudar a cor de fundo da lista, que como podemos ver pela imagem, neste caso é amarela, meter uma imagem de fundo na lista o que mudaria o *background* do local onde se encontram as tarefas de amarelo para a própria imagem, bem como, no ecrã da listas do utilizador aparecia a imagem ao invés da cor, o que pode ser observado pela figura seguinte.

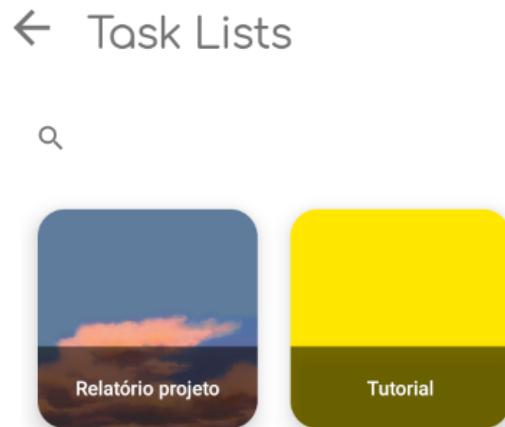


Figura 4.36: Exemplo de lista com imagem e outra com cor

De seguida temos o botão para editar/alterar o nome da lista. Disponemos ainda de um sub-menu que quando aberto apresenta mais três botões, sendo estes da esquerda para a direita respectivamente,acceder ao chat da lista onde todos os participantes da lista em questão poderão trocar mensagens, aceder ao fragmento *dashboard*(histórico da lista), o qual faz um registo de todos os acontecimentos/alterações da lista e por fim eliminar a lista.

4.4.11 Integração com o *DarkMode*

Dado que a escolha da temática de cor para a aplicação foi o branco vimos a necessidade de implementar uma integração com o modo escuro do sistema Android. Isto porque uma aplicação com o fundo branco, perante este modo do dispositivo, deve mudar a sua temática de cores para um tema escuro de modo a não causar desconforto a um utilizador habituado ao tema escuro.

A escolha dos temas de cor da aplicação é feita pela própria aplicação que verifica se o sistema possui o tema escura activo ou não, e conforme escolhe a temática para a aplicação. A figura seguinte mostra os dois temas aplicados.

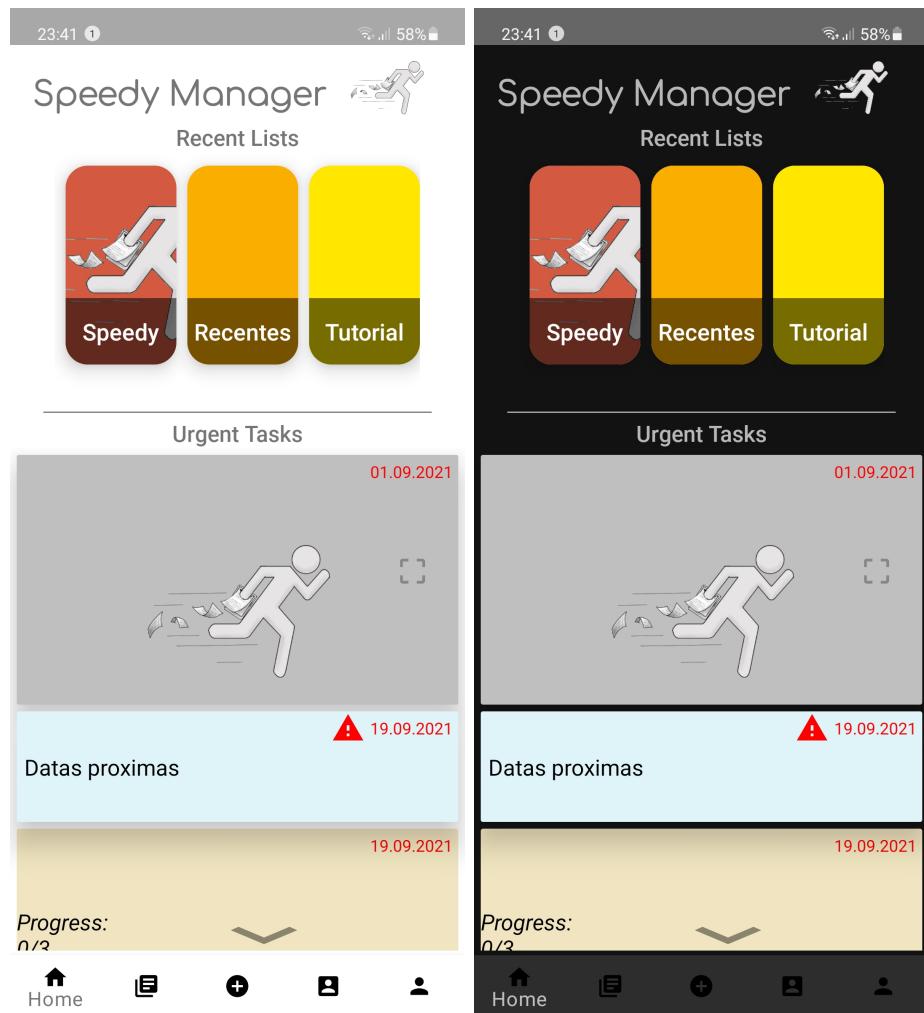


Figura 4.37: Aplicação tema normal e com tema escuro activo

Capítulo 5

Validação e Testes

Neste capítulo iremos demonstrar, através de figuras, o resultado final da aplicação. As imagens apresentadas buscam visualmente validar os diferentes requisitos da aplicação aplicados no seu respectivo contexto.

5.0.1 Pagina inicial

Começando pelo ecrã inicial,e já mencionado anteriormente,este dispõem de um menu de listas recentes que contem as ultimas três listas ace-didas pelo utilizador de modo a permitir um rápido acesso a estas, dispõem ainda, na parte inferior do ecrã de um com as tarefas urgentes, isto é, aquelas que o prazo final está a aproximar ou que já expirou mas continua incompleta.



Figura 5.1: Ecrã inicial

5.0.2 Pagina das listas de um utilizador

De seguida apresentamos o ecrã onde se encontram todas as listas de tarefas a que o utilizador pertence, quer seja por ser ele a cria-las quer seja por ter sido convidado.

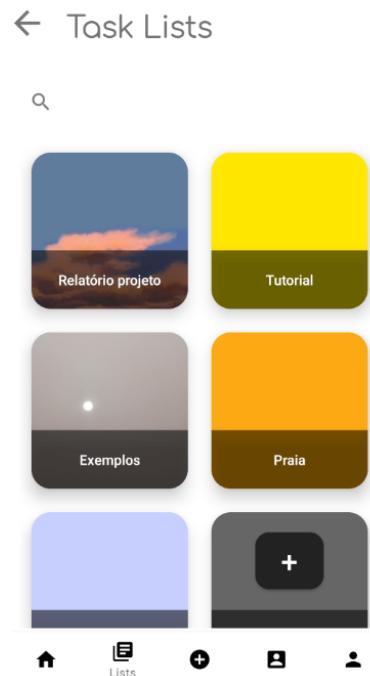


Figura 5.2: Ecrã das listas do utilizador

5.0.3 Páginas do utilizador e contactos

Temos dois ecrãs que tratam da parte dos contactos, o primeiro mostra os contactos já adicionados pelo utilizador, do género de uma lista de amigos, e existe ainda outro, acedido pelo botão em cima do lado direito do ecrã, que vai para um novo ecrã onde se pode pesquisar por utilizadores ainda não adicionados e adiciona-los aos seus contactos se assim o desejar.

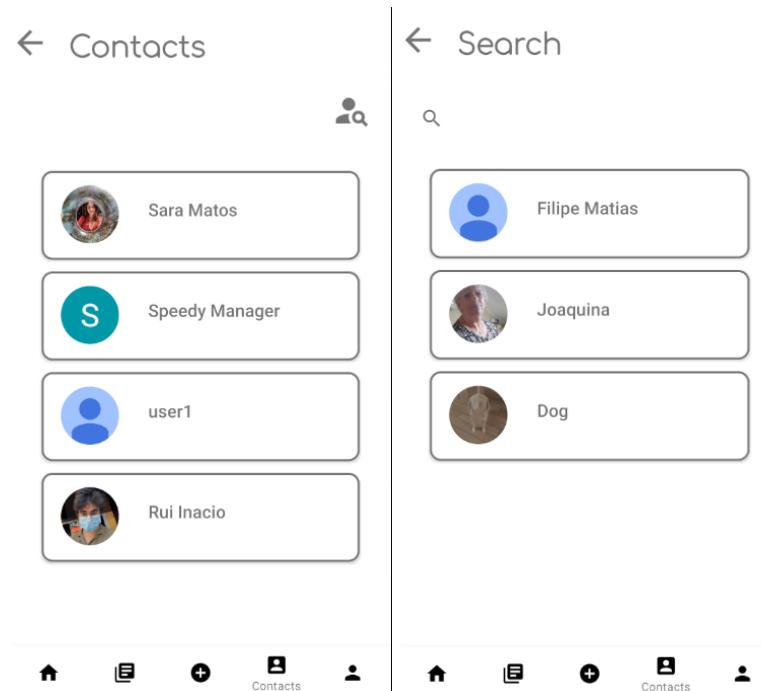


Figura 5.3: Ecrã de contactos e de pesquisa de utilizadores respectivamente

Existem 3 ecrãs semelhantes que tratam dos ecrãs de perfil de utilizadores, o ecrã do nosso perfil, de perfil de um contacto, e perfil de um utilizador que não faça parte dos contactos, enquanto que o primeiro tem funcionalidades como alterar imagem de utilizador ao clicar nela ou alterar o nome clicando no botão ao lado direito deste, o segundo e terceiro apenas têm a funcionalidade de de remover ou adicionar o contacto caso desejado.



LOGOUT

Remove from
friends

Add to friends



Figura 5.4: Ecrã de perfil próprio, de contacto, utilizador não adicionado, respectivamente

5.0.4 Pagina de uma lista de tarefas

A visualização de uma lista apresenta as tarefas ocupando a maior parte do ecrã e junto á posição de conforto do polegar temos os botões para adicionar novas tarefas ou convidar pessoas, estrategicamente posicionados.

A Figura seguinte demonstra uma lista a ser usada como tutorial para a aplicação.

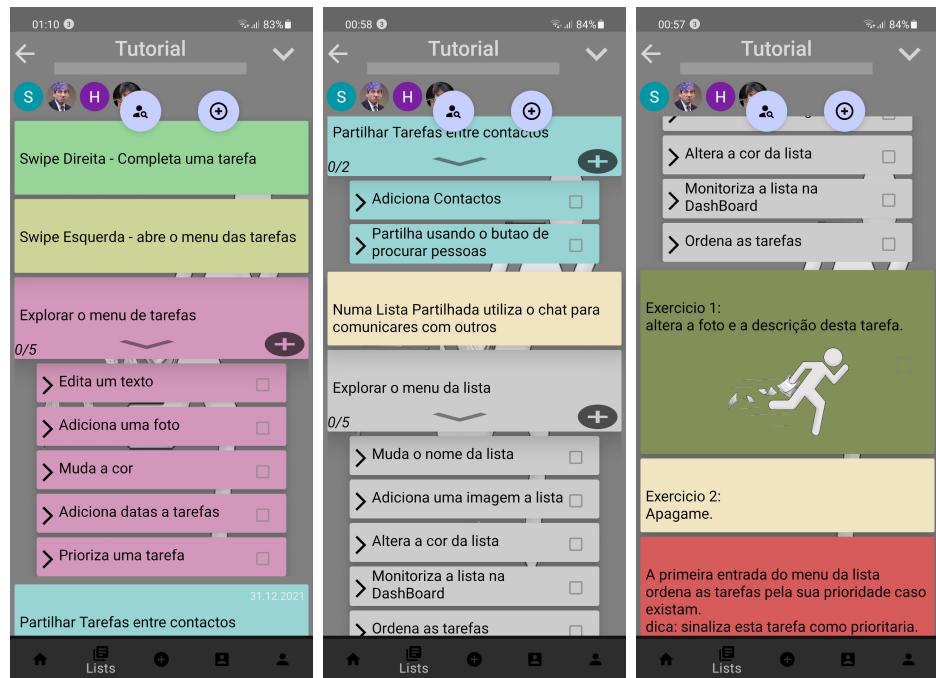


Figura 5.5: Lista usada como tutorial da aplicação.

O ecrã dentro de uma lista apresenta um menu no canto superior direito que pode ser aberto revelando botões que editam aspectos da lista. Este menu pode ser expandido para revelar o acesso ao *chat* da lista e histórico de eventos e as suas funcionalidades foram abordadas no capítulo 4.

Capítulo 6

Conclusões e Trabalho Futuro

Com a realização deste projecto conseguimos produzir uma aplicação com uma interface simples e amigável que permite aos seus utilizadores partilharem e editarem listas de tarefas em tempo real cumprindo assim os requisitos base estabelecidos para a mesma.

Ficamos satisfeitos com o resultado final da aplicação no entanto consideramos que alguns detalhes possam ainda ser refinados de modo a melhor a experiência de utilização bem como compatibilidade com diferentes resoluções de ecrã.

Consideramos que este projecto ainda não está completamente terminado dado existir ainda algum trabalho futuro para ter um sistema deste género capaz de entrar competitivamente no mercado de aplicações com esta temática.

Outros pontos de desenvolvimento futuro surgem da ideia de integrar a aplicação num sistema multi-plataforma para que a aplicação possa ser usada no contexto de um web browser.

Para tal é necessário desenvolver um *website* que apresente aos utilizadores uma interface familiar como a que é encontrada no telefone, e que possua todas as funcionalidades da aplicação. Para alojar esse *website* poderia ser usado o serviço de *web-hosting* providenciado pela Firebase.

Apêndice A

Repositório web

Utilizamos a plataforma GitHub para integrarmos o nosso projecto num repositório *web*. Este repositório segue a arquitectura apresentada na imagem seguinte. Cada *branch* contém itens associados ás diferentes fase do desenvolvimento do nosso projecto.

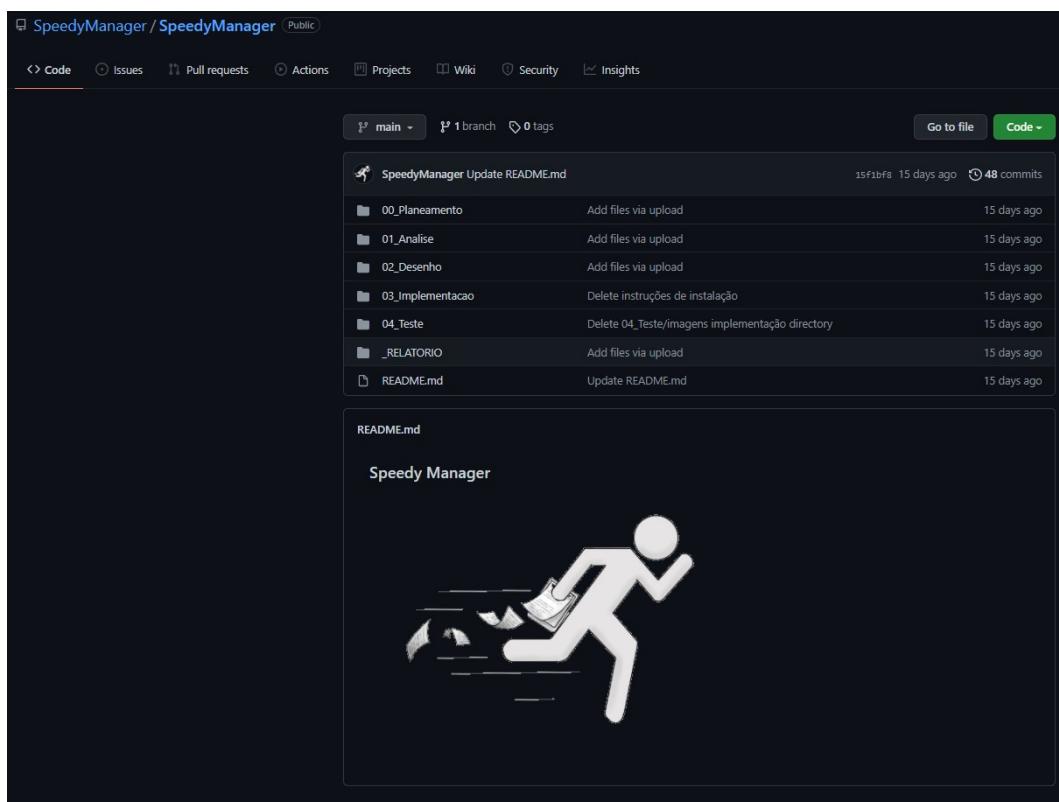


Figura A.1: Repositório GitHub

A aplicação pode ser obtida seguindo o link presente no *branch 4* do nosso repositório, assim como mostra a figura seguinte.

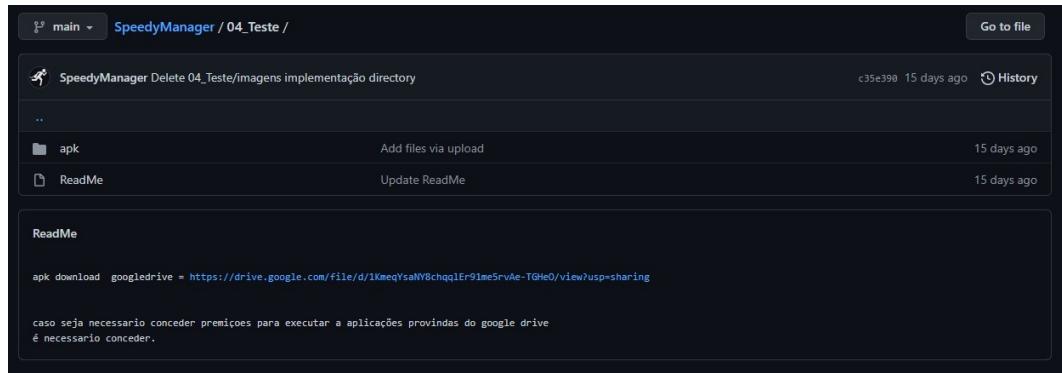


Figura A.2: Acesso ao .apk da aplicação no repositório

Apêndice B

Sistema de controlo de versões

O sistema adoptado para o controlo de versões consistiu em exportar e guardar a pasta src da nossa aplicação. Isto permite transferir e guardar todo código da nossa aplicação facilmente num tamanho de ficheiro reduzido não alterando as propriedades do projecto.

Decidimos tomar esta abordagem de guardar localmente e via histórico de transferências entre nós as varias versões do nosso projecto devido á simplicidade e eficiência deste método. A figura seguinte demonstra as versões iniciais guardadas. Mantivemos esta consistência ate á versão 80.

 src (2)	29/04/2021 19:59	ZIP Archive	1 995 KB
 src (3)	29/04/2021 23:07	ZIP Archive	1 996 KB
 src (4)	29/04/2021 23:42	ZIP Archive	1 996 KB
 src (5)	30/04/2021 17:28	ZIP Archive	1 996 KB
 src (6)	04/05/2021 09:54	ZIP Archive	1 997 KB
 src (7)	05/05/2021 18:37	ZIP Archive	2 017 KB
 src (8)	05/05/2021 21:49	ZIP Archive	2 021 KB
 src (9)	04/08/2021 14:39	ZIP Archive	2 216 KB
 src (10)	12/05/2021 18:31	ZIP Archive	2 031 KB
 src (11)	12/05/2021 22:55	ZIP Archive	2 031 KB
 src (12)	12/05/2021 23:10	ZIP Archive	2 031 KB
 src (13)	14/05/2021 11:53	ZIP Archive	2 032 KB
 src (14)	14/05/2021 16:39	ZIP Archive	2 031 KB
 src (15)	14/05/2021 19:09	ZIP Archive	2 033 KB
 src (16)	20/05/2021 17:02	ZIP Archive	2 034 KB
 src (17)	20/05/2021 17:30	ZIP Archive	8 243 KB
 src (18)	20/05/2021 17:53	ZIP Archive	8 246 KB
 src (19)	20/05/2021 17:54	ZIP Archive	2 037 KB
 src (20)	24/05/2021 16:22	ZIP Archive	2 025 KB
 src (21)	25/05/2021 17:13	ZIP Archive	2 026 KB
 src (22)	25/05/2021 17:20	ZIP Archive	2 026 KB
 src (23)	25/05/2021 22:21	ZIP Archive	2 027 KB
 src (24)	26/05/2021 02:57	ZIP Archive	2 029 KB
 src (25)	26/05/2021 17:46	ZIP Archive	2 030 KB
 src (26)	26/05/2021 18:05	ZIP Archive	2 031 KB
 src (27)	26/05/2021 18:12	ZIP Archive	2 031 KB
 src (28)	27/05/2021 02:48	ZIP Archive	2 136 KB
 src (29)	27/05/2021 03:02	ZIP Archive	2 139 KB
 src (30)	27/05/2021 03:05	ZIP Archive	2 139 KB
 src (31)	27/05/2021 23:42	ZIP Archive	2 139 KB
 src (32)	03/06/2021 21:20	Arquivo WinRAR	2 119 KB
 src (32)	03/06/2021 22:18	ZIP Archive	2 151 KB
 src (33)	03/06/2021 21:58	ZIP Archive	2 151 KB
 src (34)	11/06/2021 17:17	ZIP Archive	2 151 KB
 src (35)	13/06/2021 03:06	ZIP Archive	2 153 KB
 src (36)	16/06/2021 00:18	ZIP Archive	2 154 KB
 src (37)	21/06/2021 18:58	ZIP Archive	2 160 KB
 src (38)	22/06/2021 02:22	ZIP Archive	2 162 KB

Figura B.1: Armazenamento das versões iniciais da aplicação

O “apêndice” utiliza-se para descrever aspectos que tendo sido desenvolvidos pelo autor constituem um complemento ao que já foi apresentado no corpo principal do documento.

Neste documento utilize o apêndice para explicar o processo usado na **gestão das versões** que foram sendo construídas ao longo do desenvolvimento do trabalho.

É especialmente importante explicar o objetivo de cada ramo (“branch”)

definido no projeto (ou apenas dos ramos mais importantes) e indicar quais os ramos que participaram numa junção (“merge”).

É também importante explicar qual a arquitetura usada para interligar os vários repositórios (e.g., Git, GitHub, DropBox, GoogleDrive) que contêm as várias versões (e respetivos ramos) do projeto.

Notar a diferença essencial entre “apêndice” e “anexo”. O “apêndice” é um texto (ou documento) que descreve trabalho desenvolvido pelo autor (e.g., do relatório, monografia, tese). O “anexo” é um texto (ou documento) sobre trabalho que não foi desenvolvido pelo autor.

Para simplificar vamos apenas considerar a noção de “apêndice”. No entanto, pode sempre adicionar os anexos que entender como adequados.

Bibliografia

- [Android, a] Android, G. Introdução a atividades. <https://developer.android.com/guide/components/activities/intro-activities>.
- [Android, b] Android, G. Visão geral dos recursos de aplicativo. <https://developer.android.com/guide/topics/resources/providing-resources>.
- [AnyChart,] AnyChart. Anychart library. <https://www.anychart.com/>.
- [AnyChartAndroid,] AnyChartAndroid. Anychart for android. <https://github.com/AnyChart/AnyChart-Android>.
- [Azure,] Azure, M. Computação na nuvem. <https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/>.
- [Glide,] Glide. Glide web repository. <https://github.com/bumptech/glide>.
- [GoogleTasks, 2018] GoogleTasks (2018). Google tasks. https://play.google.com/store/apps/details?id=com.google.android.apps.tasks&hl=pt_PT&gl=US.
- [MicrosoftToDo, 2017] MicrosoftToDo (2017). Microsoft to do. <https://todo.microsoft.com/tasks/>.
- [Moroney, 2017] Moroney, L. (2017). The firebase realtime database. Technical report, Springer, TR/DCC-2006-8.
- [Trello, 2011] Trello (2011). Aplicação trello. <http://www.trello.com/>.