



ToDo Task Reminder – App para recordar tarefas por data e local

Carlos Alves

Orientador: Prof. Nuno Leite

Relatório Final realizado no âmbito de Projeto e Seminário
Licenciatura em Engenharia de Informática e de Computadores

Julho de 2022

INSTITUTO POLITÉCNICO DE LISBOA
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

ToDo Task Reminder

45835 Carlos Alves

Orientador: Nuno Leite

Relatório Final realizado no âmbito de Projeto e Seminário
Licenciatura em Engenharia Informática e de Computadores

Julho de 2022

Resumo

Com o crescente aumento da utilização de *Smartphones*, é possível observar que todo o tipo de serviços que eram outrora realizados em papel, tem agora uma *Aplicação* (App) que permite a sua realização de forma fácil, rápida, portátil e com funcionalidades que seriam impossíveis de ter anteriormente. O presente projeto tem como objetivo desenvolver uma *App* que não só substitui os antigos blocos de notas em papel, onde se tomava nota de tarefas a realizar, assim como acrescentar funcionalidades, fazendo uso das capacidades do dispositivo móvel. Funcionalidades estas que passam por recordar o utilizador de uma tarefa a realizar numa data pré-determinada, assim como num local (ou conjunto de locais) quando o utilizador se aproxima deste(s). A partilha dessas tarefas é também uma funcionalidade que tira partido das capacidades do dispositivo. É fácil concluir numa breve pesquisa que estas funcionalidades já existem individualmente, mas à data da realização deste relatório, desconhece-se que todas estas funcionalidades tenham sido incluídas numa única *App* gratuita e de fácil utilização, em que não existe necessidade da recolha de dados do utilizador.

Palavras-Chave: Lembrete de tarefas, Android, *App*, Geolocalização, Partilha de tarefas.

Lista de Acrónimos

API Application Programming Interface

App Aplicação

BD Base de dados

DAO Data Access Object

ID Identificador

MVVM Model View ViewModel

RGB Red-Green-Blue

UI User Interface

Conteúdo

Resumo	i
Lista de Acrónimos	ii
Lista de Figuras	v
1 Introdução	1
2 Formulação do Problema	5
2.1 Estado da Arte	6
2.2 Requisitos	7
2.3 Arquitetura da solução	8
2.4 Arquitetura de classes	10
3 Implementação	13
3.1 Ferramentas utilizadas	14
3.2 Modelo entidade-associação da BD	14
3.3 Permissões	14
3.4 Gestão de localizações	16
3.4.1 Criação de um local	16
3.4.2 Edição de um local	18

3.4.3	Consulta de um local	19
3.5	Tarefas	19
3.5.1	Criação de uma tarefa	19
3.5.2	Edição de tarefa	22
3.5.3	Apagar uma tarefa	23
3.5.4	Consultar uma tarefa	24
3.5.5	Partilha de tarefas	24
3.6	Notificações	26
3.6.1	Notificar por data	26
3.6.2	Notificar por local	28
3.6.3	Reagendamento de tarefas	29
3.7	Definições	29
4	Conclusões e Trabalho Futuro	31
4.1	Trabalho Futuro	32
A	Modelo entidade associação da BD	34
B	Guia de instalação e utilização	36
	Referências	38

Lista de Figuras

2.1	Diagrama da arquitectura MVVM.	9
3.1	Exemplo de um pedido de autorização para acesso à localização.	16
3.2	Aparência da <i>Activity</i> do <i>Google Maps</i>	17
3.3	Aparência da <i>Activity</i> de criação de novo local, devidamente preenchida. .	18
3.4	Aparência da <i>Activity</i> de consulta de detalhes de um novo local.	19
3.5	Aparência da <i>Activity</i> de criação de tarefas para recordar por data.	21
3.6	Aparência da <i>Activity</i> de criação de tarefas para recordar por local e selecção do mesmo.	23
3.7	Aparência da <i>Activity</i> de edição de tarefas.	24
3.8	Aparência da <i>Recycler View</i> para apagar tarefas.	25
3.9	Aparência da <i>Activity</i> para consultar tarefas.	26
3.10	Aparência do ecrã aonde é apresentada toda a informação de uma tarefa partilhada, com o botão “Apagar”.	27
3.11	Aparência de uma notificação a recordar a tarefa por data.	28
3.12	Aparência de uma notificação a recordar a tarefa por local.	29
3.13	Aparência dos ecrãs onde é possível escolher qual a nova cor, e aonde aplicá-la.	30
A.1	Modelo entidade associação da BD.	35

Capítulo 1

Introdução

Com o crescente aumento da utilização de computadores nos últimos anos, temos assistido a uma tendência cada vez maior da digitalização de dados e serviços que outrora seriam armazenados e realizados utilizando papel. São de destacar como principais vantagens da era digital:

- espaço físico: Registos em papel que chegavam a ocupar salas inteiras apenas para o seu armazenamento, hoje em dia cabem numa pen drive ou num *Smartphone*, que podem ser guardados num bolso. Com a crescente utilização dos *Smartphones*, é possível observar que todo o tipo de serviços que eram outrora realizados em papel, tem agora uma *Aplicação* (App) que permite a sua realização de forma fácil, rápida, portátil e com funcionalidades que seriam impossíveis de ter anteriormente.
- rapidez de pesquisa: A procura de informação guardada em papel implica elevado esforço e concentração por parte de uma pessoa, assim como cuidadosa organização da informação. Isto para assegurar uma pesquisa que seja o mais rápida e eficiente possível. Contudo, mesmo no melhor cenário possível de organização de informação em papel, dificilmente será tão rápida como uma pesquisa de informação que esteja armazenada em formato digital. Estamos a falar numa ordem de grandeza de horas de pesquisa em papel, versus poucos segundos em formato digital.
- persistência de dados: O papel sofre de envelhecimento com o passar do tempo, mesmo estando armazenado nas melhores condições ambiente possíveis. Com isto, corre-se o risco de se perder informação ao longo do tempo. Os componentes de armazenamento digital também não são imunes ao tempo, verificando-se a sua degradação com o passar do tempo, mas também pela sua frequência de utilização. Mas fazendo foco nos dois pontos referidos anteriormente, criar uma cópia de segurança dos dados, é um processo relativamente rápido e com uma ocupação de espaço físico marginal. Já o mesmo não se pode dizer de armazenamento em papel, cuja cópia de um arquivo de grandes dimensões poderia levar anos e/ou a ocupação

de muita mão de obra humana, assim como o seu espaço ocupado seria expectável de ser o mesmo dos dados originais.

Após as referidas vantagens, não é então de estranhar a forte adesão e migração para a era digital. O aparecimento do *Smartphone* veio impulsionar ainda mais a adesão à era digital, pois sem prejuízo das vantagens referidas, ainda veio ser somada uma nova vantagem: a portabilidade. Com o *Smartphone*, temos a conveniência de um equipamento tão pequeno que cabe num bolso, aliado à sua capacidade generosa de armazenamento de dados e acesso a serviços. Existem hoje em dia mais de 1 milhão de *Apps* disponíveis com todo o tipo de funcionalidades.

Este projecto consiste no desenvolvimento de uma *App* que tem como objectivo fazer o papel de um lembrete de tarefas, onde o utilizador tem a possibilidade de:

- criar tarefas para serem recordadas por data, fazendo uso do serviço de alarme do *Android*;
- por local, utilizando o serviço de localização do *Google Maps*;
- partilha das tarefas, com recurso às funcionalidades do *Firestore*.

O que distingue esta *Apps* das restantes referenciadas em *nTask (2021)*, é o facto de combinar determinadas funcionalidades que até à data de elaboração deste relatório, não há conhecimento de mais nenhuma *App* que o faça.

A restante estrutura deste relatório está organizada em três capítulos. No capítulo 2, Formulação do Problema, é feito o enquadramento do projeto. Começa-se com a descrição de outras *Apps* similares ao sistema proposto. De seguida, definem-se os requisitos do sistema a implementar bem como decisões tomadas na escolha de tecnologias e metodologias.

No capítulo 3, Implementação, é exibida a elaboração dos diversos componentes do projecto, tendo em conta as tecnologias e os métodos escolhidos.

Por fim, no capítulo 4 formulam-se as conclusões retiradas da elaboração do projecto e propõe-se trabalho futuro e melhorias que possam enriquecer o mesmo.

Capítulo 2

Formulação do Problema

No presente capítulo, é feito o enquadramento do projeto, onde se começa com a descrição de outros sistemas similares ao sistema proposto. De seguida, definem-se os requisitos do sistema a implementar bem como decisões tomadas na escolha de tecnologias e metodologias.

2.1 Estado da Arte

Existem actualmente uma grande variedade de *Apps* que têm como objectivo o armazenamento de tarefas e o recordar das mesmas ao utilizador. Tipicamente esta funcionalidade de recordar é baseada numa data, ou seja, as referidas *Apps* têm como foco uma data, e possíveis tarefas a realizar na mesma. Neste projecto, o foco reside na tarefa. Este foco na tarefa significa que a mesma tarefa pode ser recordada em diferentes contextos, sendo eles uma data, um (ou vários) local(ais), ou até nem ser recordada de todo, funcionando apenas como bloco de notas, ou histórico de algo que já foi feito. Até à data de início da elaboração deste projecto, foi constatado que *Apps* desta natureza que recordem o utilizador de uma tarefa baseadas num local, são escassas. Mais escassas ainda são aquelas que permitem a fácil partilha dessas mesmas tarefas por vários utilizadores, de acordo com a lista de *Apps* referenciada em *nTask (2021)*,

Existe hoje em dia enorme preocupação com a recolha e processamento de dados do utilizador, dados esses que o utilizador se vê forçado a fornecer caso pretenda usufruir de determinado serviço. Já são várias as leis criadas para assegurar a protecção de dados em que, antes da existência das mesmas, os dados acabavam frequentemente nas mãos de terceiros para fins publicitários, e até maliciosos. No presente projeto, definiu-se como objetivo disponibilizar todas as funcionalidades mencionadas, sem que para isso o utilizador tenha de fornecer um único dado que o identifique. Portanto, qualquer possível ataque ou mau funcionamento do serviço de partilha de tarefas, não irá colocar em risco quaisquer dados sensíveis do utilizador.

2.2 Requisitos

De forma a cumprir com os objectivos propostos, esta *App* deverá implementar os seguintes requisitos:

- o armazenamento de dados de forma persistente em base de dados. Este requisito é fundamental, pois tanto uma tarefa como um local precisam de persistir após o encerramento da *App*, assim como persistir após o reiniciar do dispositivo;
- acesso ao sistema de Geolocalização do *Android*. Apesar deste requisito não ser fundamental, a sua não implementação limita fortemente a experiência de utilização, pois desta forma não é possível à *App* ter acesso à localização do dispositivo, e assim não poderá ser utilizada a funcionalidade de recordar o utilizador baseado num local;
- acesso à Internet. Apesar deste requisito não ser fundamental, a sua não implementação limita a experiência de utilização, pois desta forma não é possível o acesso à funcionalidade de partilha de tarefas. Fica também reduzida a experiência de utilização no que toca à adição de novos locais, pois o acesso à internet possibilita que o *Google Maps* facilite o processo de navegação e aquisição de locais;
- reagendamento de tarefas. Apesar deste requisito não ser fundamental, a sua não implementação pode levar ao não funcionamento da funcionalidade de recordar o utilizador, tanto por data como por local. Em ambos os casos, o reiniciar do dispositivo leva a que todos os agendamentos desapareçam, sendo necessário reagendar. Aquando do reinício do dispositivo, o sistema operativo *Android* informa a *App* dessa ocorrência, o que por sua vez vai desencadear o reagendamento de todas as tarefas;
- interacção com *App* de navegação. Este requisito visa apenas acrescentar a possibilidade de exportar um local para a *App* de navegação do *Google Maps* quando

é lançada uma notificação de aproximação a um local. A não implementação em nada compromete nenhuma outra funcionalidade da *App*.

2.3 Arquitetura da solução

Sendo o *Android* e o *iOS* os principais sistemas operativos hoje em dia utilizados para dispositivos móveis, já existem diversas soluções que permitem a criação de uma *App* multiplataforma. Ou seja, o mesmo código permite criar uma *App* que tanto corre em *Android* como em *iOS*. Mas para efeitos deste projecto, ficou definido que esta seria exclusiva para *Android* pelos seguintes motivos:

- experiência significativa do programador em interagir com *Android*, e nenhuma com *iOS*;
- dispositivo móvel pessoal *Android* facilita testes e depuração de erros;
- uma *App* feita exclusivamente para uma única plataforma irá sempre usufruir de melhor desempenho e suporte por parte do fabricante.

Uma vez definido o projecto como sendo uma *App* exclusiva para *Android*, a próxima decisão seria a linguagem a adoptar. Apesar de *Java* ser uma linguagem consolidada e amplamente utilizada no desenvolvimento de aplicações *Android*, foi escolhido *Kotlin* como linguagem a ser utilizada, pelos seguintes motivos:

- trata-se de uma linguagem relativamente nova, mais fácil de utilizar que *Java*, e com um crescente número de bibliotecas auxiliares;
- foi recentemente declarada pela *Google* como a linguagem oficial do *Android*;
- por ter sido declarada como linguagem oficial do *Android*, cada vez mais a documentação disponibilizada pela *Google* é orientada ao *Kotlin* e cada vez menos ao *Java*.

Relativamente à arquitectura da *framework*, foi escolhida a *Model View ViewModel* (MVVM) por ser uma arquitectura relativamente recente, e que melhor se enquadra no fluxo pretendido de informação na *App*, em que o foco principal (o ponto inicial e final do fluxo) é a *View*, como se pode observar na figura 2.1

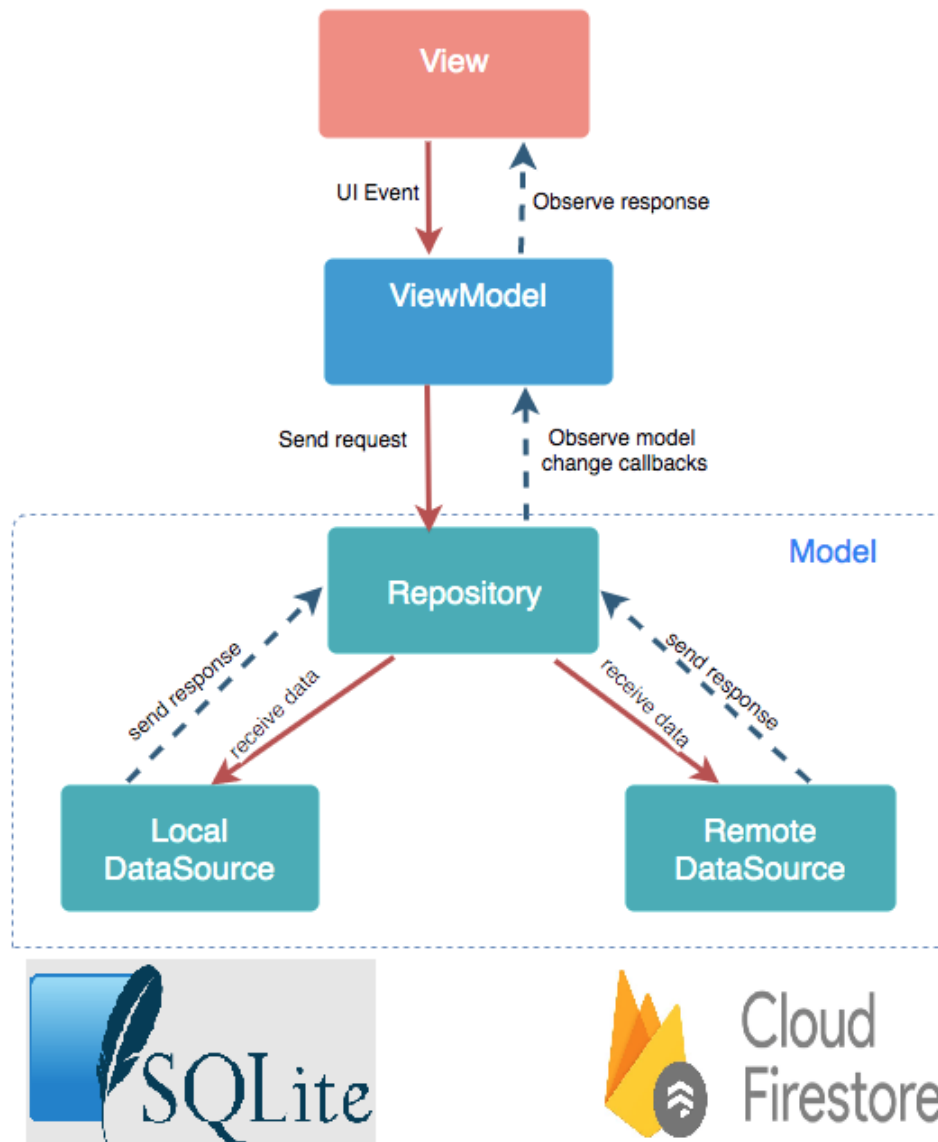


Figura 2.1: Diagrama da arquitectura MVVM.

Para todos os ecrãs, existe uma *Activity*, que é representada como uma *View* neste diagrama, e nela fica definida a aparência e comportamentos dentro do contexto do próprio

ecrã aquando da interacção do utilizador. Grande parte das referidas *Activities* tem associado um *ViewModel*, também representado neste diagrama com o mesmo nome, o qual define um comportamento que sai fora do contexto do próprio ecrã, em que são requisitados dados a um repositório. O *ViewModel* tem também a capacidade de persistir dados no caso de *App* ir temporariamente para segundo plano. Esse repositório pode aceder a uma fonte de dados de natureza local, como é o caso da base de dados *SQLite* utilizada para armazenar as tarefas e localizações, assim como também pode aceder a uma fonte de dados de natureza remota, como é o caso da base de dados *Firestore* para armazenar as tarefas partilhadas.

2.4 Arquitetura de classes

Devido à relativamente extensa dimensão deste projecto, as classes foram separadas e armazenadas em *packages* cujos nomes foram cuidadosamente escolhidos de forma a serem elucidativos relativamente à função das classes em si contidas. Os *packages*, assim como o seu conteúdo e função são os seguintes:

- *adapters* - Os *adapters* são os elementos que compõem uma *RecyclerView*. A *RecyclerView* trata-se de uma lista de elementos baseada em *lazy loading*, com as vantagens citadas no artigo *Imperva (2021)* e seguindo as orientações de *AndroidDevelopers (2022c)*. O papel dos *adapters* é então definir o seu comportamento quando se selecciona um elemento, a aparência da lista, se deverá existir uma escuta activa sobre os elementos, entre outros. Neste *package*, existe então um *adapter* para cada uma das *RecyclerViews* existentes no projecto;
- *checkTasks* - Consiste nas *Activities* e respectivos *ViewModels* da funcionalidade de consulta das tarefas armazenadas;
- *createTasks* - Consiste na *Activity* e respectivo *ViewModel* da funcionalidade de criação de tarefas;

- *database* - Aqui é onde está definida a estrutura da base de dados. As classes cujo nome termina em “*entity*”, representam a estrutura de cada uma das tabelas, e as respectivas anotações (também com o nome “*Entity*”) de cada classe definem as suas restrições de integridade. Existem também aqui classes com a anotação *Data Access Object* (DAO), que permitem estabelecer a ponte entre a base de dados e os repositórios.
- *deleteTasks* - Consiste na *Activity* e respectivo *ViewModel* da funcionalidade de remoção de tarefas armazenadas;
- *editTasks* - Consiste nas *Activities* e respectivos *ViewModels* da funcionalidade de edição das tarefas armazenadas;
- *locationManagement* - Consiste nas *Activities* e respectivos *ViewModels* da gestão de locais, incluindo a interação com o *Google Maps*;
- *locationSelection* - Consiste na *Activity* e respectivo *ViewModel* da funcionalidade de associação de locais a uma tarefa, aquando da sua criação e edição;
- *notifications* - Inclui as classes que estão envolvidas no envio de notificações, assim como o reagendamento das mesmas quando o dispositivo é reiniciado. Tratam-se de classes que estendem de *Broadcast*, e portanto, recebem e reagem às emissões de eventos por parte do sistema operativo, nomeadamente à aproximação de um local, uma data/hora, e ao reinício do dispositivo. É também aqui que ocorre o registo e respectivo cancelamento dos agendamentos de local e data/hora;
- *repository* - Aqui é onde estão definidas classes que contêm as *queries* (convertidas em métodos) para acesso específico a certos dados das respectivas tabelas da base de dados. As classes em causa terminam todas em “*repository*”, e qualquer acesso à base de dados passa forçosamente por elas. Está aqui também definida a classe responsável por toda a comunicação com o *Google Firestore*;

- *settings* - Consiste nas *Activities* e respectivos *ViewModels* da funcionalidade de alteração das cores de fundo e dos botões;
- *sharedTasks* - Consiste nas *Activities* e respectivos *ViewModels* da funcionalidade de partilha de tarefas na *cloud*.
- *utilities* - Aqui ficam definidas classes utilitárias que disponibilizam serviços úteis em diferentes locais do código. Dos quais se destacam conversores, que convertem um tipo de objecto noutra. É também aqui que fica a classe encarregue de verificar e solicitar as permissões do utilizador, quando ocorre o acesso a um serviço que as requeira;

Fora de qualquer *package* ficaram as classes *ToDoTaskReminderApp*, *SplashActivity* e *MainMenuActivity*, pois tratam-se apenas de ecrãs de arranque e instanciação de serviços.

Capítulo 3

Implementação

O presente capítulo descreve a implementação do ToDo Task Reminder, designadamente os componentes de software que o compõem e os recursos usados, tendo em conta as tecnologias e metodologias escolhidas, descritas no capítulo 2.

3.1 Ferramentas utilizadas

Relativamente a ferramentas utilizadas, o *Android Studio* revelou-se como a ferramenta completa e ideal para a realização deste projecto, pois reúne todas as funcionalidades necessárias, dispensando assim o uso de qualquer outra ferramenta adicional.

3.2 Modelo entidade-associação da BD

Todos os dados que necessitam de ser mantidos de forma persistente, são mantidos numa *Base de dados* (BD) relacional da biblioteca *Room*, baseada em *SQLite*. A figura A.1 presente no Apêndice A, é alusiva à estrutura de dados usada na base de dados. É possível identificar todas as tabelas, assim como o relacionamento entre as várias entidades. Nesta figura foram utilizados os nomes em português, opção justificada pelo facto de ser a linguagem utilizada na escrita do relatório. Mas no código, as tabelas e respectivos atributos encontram-se em inglês, de acordo com as boas práticas da programação.

3.3 Permissões

Tendo em conta que esta *App* faz uso de diversos recursos físicos do dispositivo, as regras do *Android* ditam que o acesso a esses recursos apenas é possível se o utilizador assim o autorizar, devendo-se assim seguir as recomendações citadas no site *AndroidDevelopers* (2022d). Neste sentido, foi criada uma classe que lida com esta questão, em que quando o utilizador está em vias de fazer uso de uma funcionalidade que exige uma das referidas permissões, é pedido ao utilizador que conceda essa permissão. Na eventualidade do

o utilizador não autorizar, este é informado e impedido de utilizar a funcionalidade em causa. Ainda assim, mesmo que o utilizador tenha autorizado o acesso a todos os recursos do dispositivo, imediatamente antes da utilização de um dado recurso, é validado se o acesso a este se encontra autorizado. Esta opção foi tomada respeitando as boas práticas recomendadas pela *Google* no site *AndroidDevelopers* (2022a). Estas práticas visam evitar comportamentos imprevisíveis por parte da *App*, dado que, mesmo depois do acesso a todos os recursos ter sido autorizado, a qualquer momento, o utilizador pode remover essa autorização. De acordo com as especificações do *Android*, as autorizações necessárias são:

- *Coarse location* - Permite que a *App* tenha acesso à localização aproximada do dispositivo;
- *Fine location* - Permite que a *App* tenha acesso à localização exacta do dispositivo;
- *Background location* - Na eventualidade de a *App* ter uma execução de código agendada (e que a própria *App* não se encontre em execução), é permitido o acesso à localização do dispositivo;
- *Schedule exact alarm* – Permite agendar a execução de código a uma hora exacta (apenas necessário se versão da API ≥ 31 (*Android* 12 ou superior));
- *Internet* - Permite que a *App* tenha acesso à *Internet*.

A figura 3.1 ilustra um exemplo de um pedido de autorização para acesso à localização do dispositivo. A aparência e abrangência destes pedidos de autorização variam com a versão do *Android* em causa. Este exemplo foi tirado do *Android* 10, e neste pedido está contemplado o acesso à *Coarse location*, *Fine location* e *Background location*.

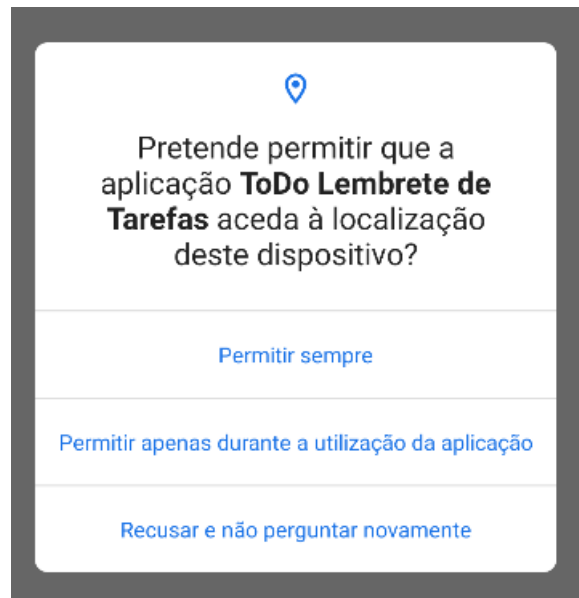


Figura 3.1: Exemplo de um pedido de autorização para acesso à localização.

3.4 Gestão de localizações

3.4.1 Criação de um local

O primeiro passo e potencialmente o mais complexo, é a interacção com o *Google Maps*. O *Android Studio* facilita esta implementação no sentido de disponibilizar a criação de uma *Activity* cujo único objectivo é apenas a interacção com o *Google Maps*, cujas funcionalidades estão descritas no site *GoogleDevelopers* (2022b). É possível observar na figura 3.2 a aparência dessa mesma *Activity*.

Existem então quatro classes que são imediatamente disponibilizadas na criação desta *Activity* que facilitam todo o acesso aos dados pretendidos e manipulação do mapa. São elas:

- *FusedLocationProviderClient* – Cria uma camada de abstracção de todo o processo de comunicação com a *Google Maps API*. Qualquer acesso à *Application Programming Interface* (API) é realizado facilmente com a chamada de um método desta classe (foi usada apenas para adquirir a localização actual do utilizador);



Figura 3.2: Aparência da Activity do *Google Maps*.

- *GoogleMap* - Possibilita a interacção com o mapa. Usada neste projecto para adicionar marcadores ao mapa, assim como deslocar a câmara para um determinado local;
- *Geocoder* – Ter acesso às informações de um determinado local. Usada neste projecto para ter acesso à morada de um determinado local;
- *PlacesClient* – Permite pesquisar um local baseado num nome. Por exemplo, procurar no mapa uma loja, usando como palavras-chaves o seu nome comercial. Não é uma funcionalidade essencial, mas melhora significativamente a experiência do utilizador aquando da pesquisa por um novo local.

Uma vez seleccionada uma localização no *Google Maps*, é dada ao utilizador a hipótese de editar o nome e morada do local, assim como associá-lo a um grupo.

Convertendo agora as coordenadas para uma *String* cujo formato foi definido de forma a facilitar futuras interacções (passar as coordenadas à *App* de navegação), estamos em condições de guardar este novo local na base de dados. Está exemplificada na figura 3.3 a aparência de um local devidamente preenchido e pronto para ser salvo na BD.

The screenshot shows a mobile application interface for creating a new location. The form is filled with the following data:

- Nome:** Instituto Superior de Engenharia de Lis
- Morada:** R. Conselheiro Emídio Navarro 1, 195
- Pertence a um grupo:** ☒ (checked)
- Pertence a um grupo já existente:** ☐ (unchecked)
- Pertence a um novo grupo:** ☒ (checked)
- Grupo:** Escolas do IPL

At the bottom of the form, there are three buttons: "ESCOLHER LOCALIZAÇÃO NO MAPS", "VOLTAR", and "CRIAR".

Figura 3.3: Aparência da *Activity* de criação de novo local, devidamente preenchida.

3.4.2 Edição de um local

A edição de um local segue todo o fluxo de dados da criação de um local, em que até foi possível aproveitar as *Activities* e respectivos *View Models*. A única diferença é que a *Activity* surge com todos os dados relativos à localização seleccionada, preenchidos com valores lidos da base de dados.

3.4.3 Consulta de um local

A consulta de um local consiste apenas no acesso à base de dados para consulta de todos os dados relativos a uma localização, e o preenchimento desses dados na respectiva *Activity*, conforme exemplificado na figura 3.4.

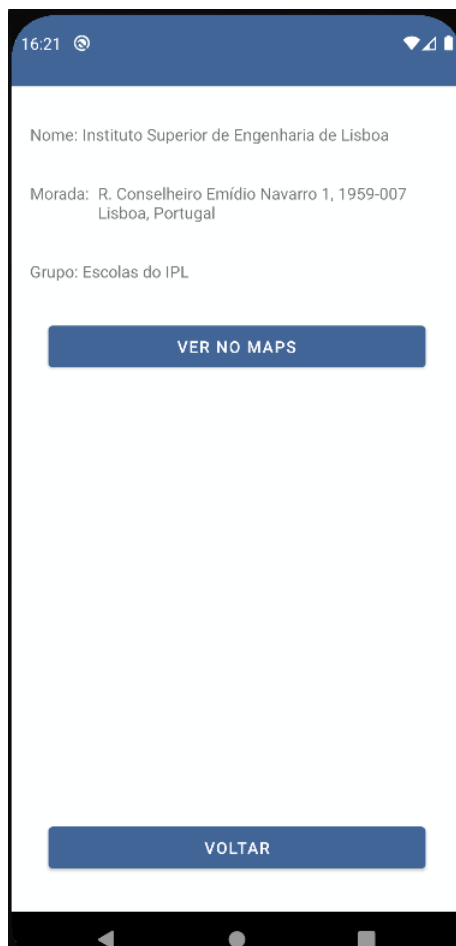


Figura 3.4: Aparência da *Activity* de consulta de detalhes de um novo local.

3.5 Tarefas

3.5.1 Criação de uma tarefa

Na criação de uma nova tarefa, o utilizador começa por preencher o nome e descrição da tarefa. Apesar de uma tarefa ser identificada por um *Identificador* (ID) na base de dados,

ficou decidido que ainda assim não seria permitido a existência de várias tarefas com o mesmo nome. Esta decisão visa apenas facilitar a experiência de utilização do utilizador.

Uma vez preenchidos os referidos campos, o utilizador deve então decidir se pretende que esta tarefa seja recordada com base numa data, uma localização, ambos, ou nenhum.

Por fim, existe uma *checkbox*, que uma vez seleccionada, permite que a tarefa criada seja armazenada também na *cloud* para posterior partilha com outros utilizadores. A selecção desta *checkbox* faz surgir de imediato um *pop-up* onde o utilizador deverá definir as palavras-chave de acesso à tarefa em causa. Existe uma palavra-chave para o utilizador, e existe uma segunda palavra-chave de administrador. Uma tarefa partilhada na *cloud*, terá um *ID online* único gerado aleatoriamente, independente do *ID* que a tarefa tenha localmente. Este *ID online* irá permitir identificar a tarefa na *cloud*. Os detalhes relativos à partilha de tarefas são abordados em maior pormenor na secção intitulada “Partilha de tarefa”.

Ao pressionar o botão “Criar”, são validados todos os dados introduzidos no sentido de não criar uma tarefa de conteúdo inconsistente. As validações são as seguintes:

- o nome da tarefa tem de estar preenchido e não pode ser um nome já existente na base de dados;
- a descrição tem de estar preenchida;
- no caso de seleccionar “Recordar por data”, a data e hora devem estar preenchidos e não podem pertencer ao passado. É também verificado se está autorizada a permissão de criar alarmes;
- no caso de seleccionar “Recordar por local”, deve existir pelo menos um local seleccionado, e a distância deverá ser igual ou superior a 0,1 Km.

Recordar por data

Quando o utilizador decide que deseja ser recordado por uma data, é apresentado um *Layout* onde deve ser preenchida a data e hora a que a notificação deverá ser lançada. Foram utilizadas as classes *DatePickerDialog* e a *TimePickerDialog* do *Android* que dão acesso a uma *User Interface* (UI) que permite ao utilizador de forma fácil escolher uma data e hora respectivamente. Para além de facilitar a experiência de utilização, existe também uma vantagem do lado do programador: o facto de não necessitar fazer validação do formato da data e hora. A única validação que fica assim a precisar de ser feita é se essa data/hora já foi ultrapassada. É possível verificar a aparência deste ecrã na figura 3.5.

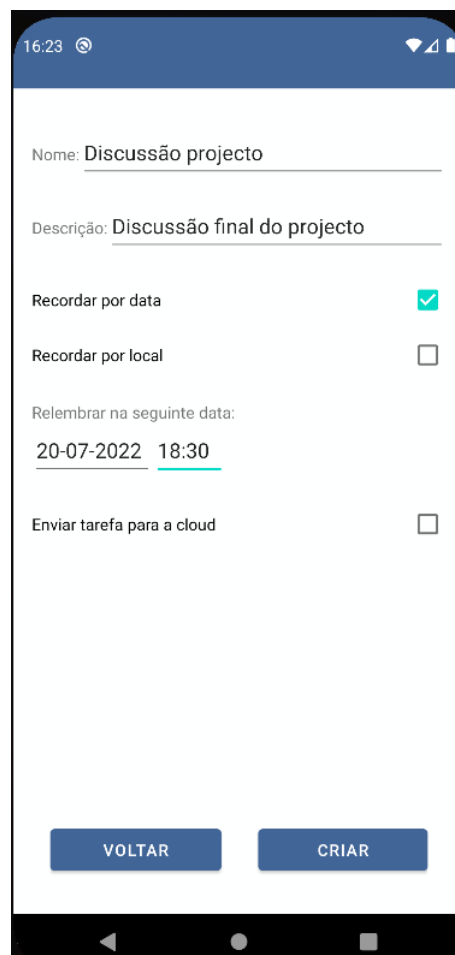


Figura 3.5: Aparência da *Activity* de criação de tarefas para recordar por data.

Recordar por local

Quando o utilizador decide que deseja ser recordado por local, deverá escolher não só os locais, mas também a distância à qual deve ser recordado dos mesmos. Importa referir que a distância escolhida será válida para todos os locais seleccionados desta tarefa. Para escolher as localizações, o utilizador deve pressionar um botão que o transporta para uma *Activity* que lhe apresenta todos os locais adicionados previamente, e agrupados por grupo. Neste menu, foi decidido que não se iria incluir a hipótese de criar um novo local, pois a pouca utilização esperada desta funcionalidade não iria compensar a sua complexidade de implementação. Uma vez seleccionados todos os locais, o utilizador poderá pressionar o botão “Salvar e voltar”, onde retorna ao menu anterior da criação da tarefa, mas com os locais seleccionados já associados à tarefa. Na figura 3.6 pode observar-se a aparência de ambos os ecrãs, o de criação de uma tarefa a recordar por local, assim como o da selecção dos respectivos locais.

3.5.2 Edição de tarefa

No caso de o utilizador pretender editar uma tarefa, é-lhe apresentada uma *Recycler View* com todas as tarefas criadas até ao momento. Ao seleccionar uma tarefa, o utilizador é direccionado para um ecrã muito semelhante ao da criação de tarefas, com a diferença que este já se encontra preenchido, e tem uma opção adicional para marcar uma tarefa como concluída. Apesar das muitas semelhanças entre a *Activity* de criação e edição de tarefas ter levado à ponderação de uma *Activity* partilhada (à semelhança do que aconteceu com a criação e edição de locais), foi o significativo número de diferenças que levou à separação destas. Todo o comportamento da *Activity* de edição de tarefas é idêntico ao da *Activity* de criação de tarefas. A única diferença é na validação de uma data que pertence ao passado, que é apenas permitida caso a tarefa esteja marcada como concluída. Importa referir que caso o utilizador desmarque a opção de partilha de tarefa na *cloud*, a tarefa não será removida da *cloud*. Apenas é removida a associação da tarefa em causa ao *ID online*

The image displays two side-by-side screenshots of a mobile application interface, likely for task management. Both screens show a status bar at the top with the time 16:28 and various icons.

Left Screenshot (Form):

- Nome:** Discussão projecto
- Descrição:** Discussão final do projecto
- Recordar por data:** ☐
- Recordar por local:** ☒
- ESCOLHER LOCALIZAÇÕES** (button)
- Relembrar-me quando a distância for igual ou inferior a:** 1.2 km
- Enviar tarefa para a cloud:** ☐
- VOLTAR** (button) and **CRIAR** (button) at the bottom.

Right Screenshot (Location Selection):

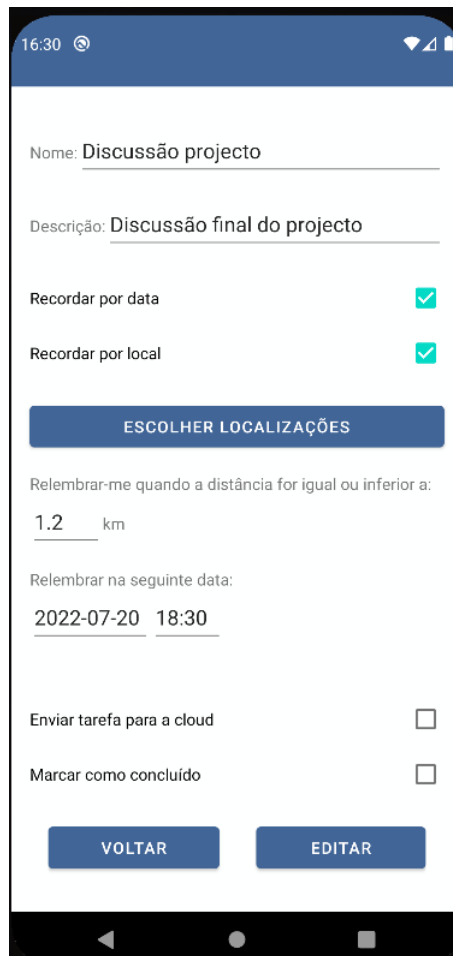
- ☒ Escolas do IPL
- ☒ Instituto Superior de Engenharia de Lisboa
- ☐ Sem grupo
- ☐ Toys"R"Us Telheiras
- SALVAR E VOLTAR** (button) at the bottom.

Figura 3.6: Aparência da *Activity* de criação de tarefas para recordar por local e selecção do mesmo.

da tarefa na *cloud*. É possível verificar a aparência deste ecrã na figura 3.7.

3.5.3 Apagar uma tarefa

Para apagar uma tarefa, é apresentada também neste caso ao utilizador uma *Recycler View* com todas as tarefas criadas até ao momento, onde através de *checkboxes*, pode seleccionar as tarefas que pretende apagar, e pressionar o botão “Apagar”. A aparência deste ecrã pode ser vista na figura 3.8.



The screenshot shows a mobile application interface for editing a task. At the top, the status bar displays the time 16:30 and various icons. The app's header is a solid blue bar. Below the header, the task details are as follows:

- Nome:** Discussão projecto
- Descrição:** Discussão final do projecto
- Recordar por data:** ☒
- Recordar por local:** ☒
- ESCOLHER LOCALIZAÇÕES:** A blue button.
- Relembrar-me quando a distância for igual ou inferior a:** 1.2 km
- Relembrar na seguinte data:** 2022-07-20 18:30
- Enviar tarefa para a cloud:** ☐
- Marcar como concluído:** ☐

At the bottom, there are two blue buttons: **VOLTAR** and **EDITAR**. The bottom of the screen shows the standard Android navigation bar with back, home, and recent apps icons.

Figura 3.7: Aparência da *Activity* de edição de tarefas.

3.5.4 Consultar uma tarefa

Para consultar uma tarefa, é apresentada ao utilizador uma *Recycler View* com todas as tarefas criadas até ao momento. Ao seleccionar a tarefa pretendida, o utilizador é direccionado para um ecrã que apresenta todos os dados relativos à tarefa. Este ecrã destina-se apenas a consulta, não podendo os dados serem modificados. A figura 3.9 ilustra um exemplo deste ecrã .

3.5.5 Partilha de tarefas

Como referido anteriormente, as tarefas armazenadas na *cloud* são identificadas por um *ID online* gerado aleatoriamente. Ao pressionar o botão “Tarefas partilhadas”, surge um

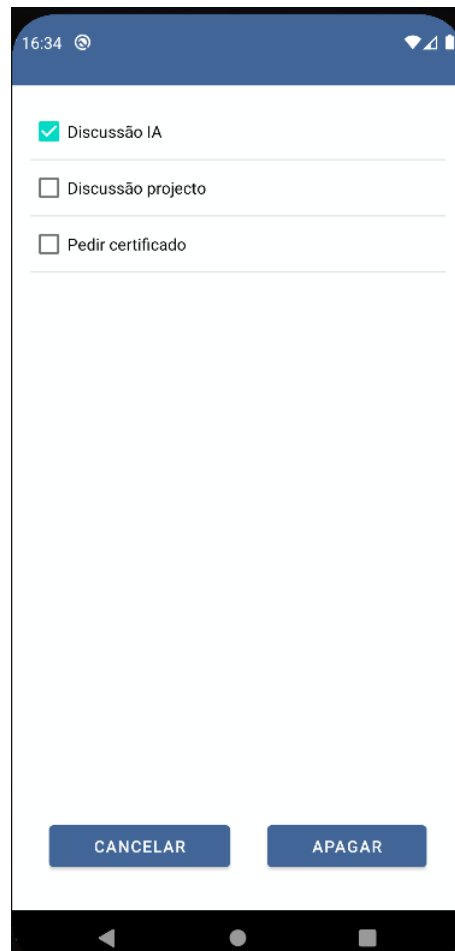


Figura 3.8: Aparência da *RecyclerView* para apagar tarefas.

pop-up que pede então o *ID online* da tarefa, assim como uma palavra-chave. Como foi igualmente referido, aquando da criação de uma tarefa partilhada, é pedido para se definirem 2 palavras-chave: uma de utilizador, a outra de administrador. Ao consultar uma tarefa partilhada, independentemente da palavra-chave introduzida, será carregada e apresentada toda a informação relativa à tarefa partilhada, assim como um botão que permite armazenar a tarefa localmente na base de dados do dispositivo. No entanto, a introdução da palavra-chave de administrador vai originar uma diferença. Se a palavra-chave for a de administrador, irá surgir toda a informação de forma idêntica como se fosse para o utilizador, mas terá também um novo botão, que permite apagar a tarefa da *cloud*. A importação para o dispositivo local apenas é possível se não existir já uma tarefa



Figura 3.9: Aparência da *Activity* para consultar tarefas.

com o mesmo nome. Importa referir que as palavras-chave são armazenadas na *cloud* após serem submetidas a um processo de *Hash*, recorrendo ao protocolo *SHA-256*. A aparência deste ecrã pode ser vista na figura 3.10.

3.6 Notificações

3.6.1 Notificar por data

Quando o utilizador cria uma tarefa para ser recordada por data, é chamado o método *set-DateToRemind* da classe *DateReminderService*, que vai primeiramente validar se existe autorização do utilizador para utilizar este serviço. A não autorização cancela de imediato

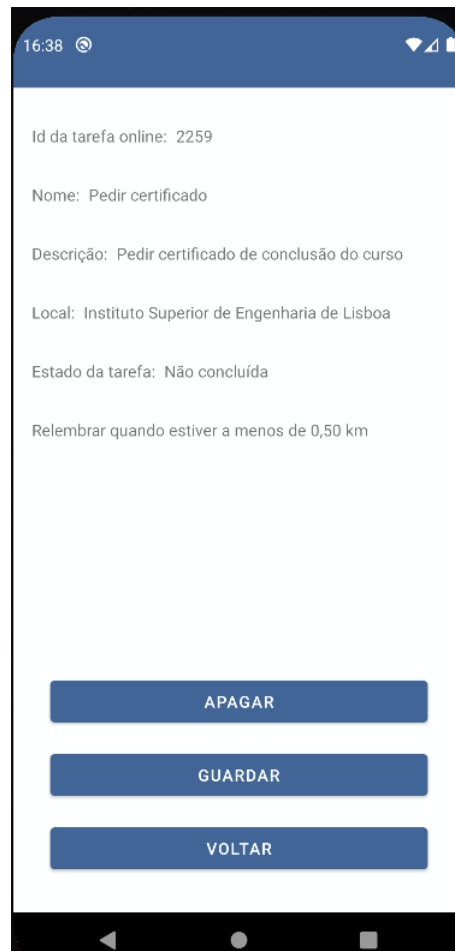


Figura 3.10: Aparência do ecrã aonde é apresentada toda a informação de uma tarefa partilhada, com o botão “Apagar”.

todo o processo. Verificada a autorização, é então criada uma instância de *AlarmManager*, conforme mencionado no site *AndroidDevelopers (2022e)*, sendo passada como parâmetro a data de quando este alarme deve disparar, assim como o *ID* e nome da tarefa, afim de lançar uma notificação com a informação pertinente à respectiva tarefa. À hora definida, é então lançado um evento por parte do sistema operativo, que vai preparar e lançar a notificação com o nome da tarefa, assim como um botão presente na notificação, que direcciona o utilizador para o ecrã de edição de tarefa, pois é expectável que o utilizador deseje alterar algo na tarefa, potencialmente marcá-la como concluída ou agendar para outra hora. A figura 3.11 ilustra uma notificação lançada a recordar uma tarefa baseada numa data.

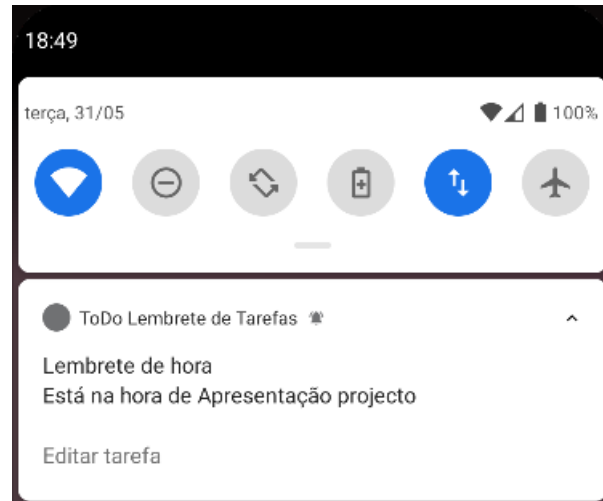


Figura 3.11: Aparência de uma notificação a recordar a tarefa por data.

3.6.2 Notificar por local

A notificação por local recorre ao serviço *Geofence* do *Android*, conforme mencionado no site *AndroidDevelopers* (2022b). Funciona de forma semelhante à notificação por data, em que é solicitado ao sistema operativo que lance um evento numa determinada hora, mas que neste caso, o referido evento será lançado quando o dispositivo se situar geograficamente num local previamente estabelecido. Aqui vai também existir uma validação para verificar se há então autorização do utilizador para utilizar este serviço. A não autorização cancela de imediato todo o processo. Verificada a autorização, é então criada uma instância de *Geofence*, e são passadas a esta como parâmetro, as coordenadas e a distância do local em causa que deve desencadear o evento, assim como um *ID* que irá permitir identificar qual a tarefa que pretende ser notificada deste evento, afim de lançar uma notificação com a informação pertinente ao local e à respectiva tarefa. A notificação apresenta dois botões:

- O botão de editar a tarefa (segue o mesmo princípio e lógica da notificação por data);
- O botão para chamar a função de navegação do *Google Maps*, passando-lhe então

a coordenadas do local de destino;

A figura 3.12 ilustra uma notificação lançada a recordar uma tarefa baseada num local.

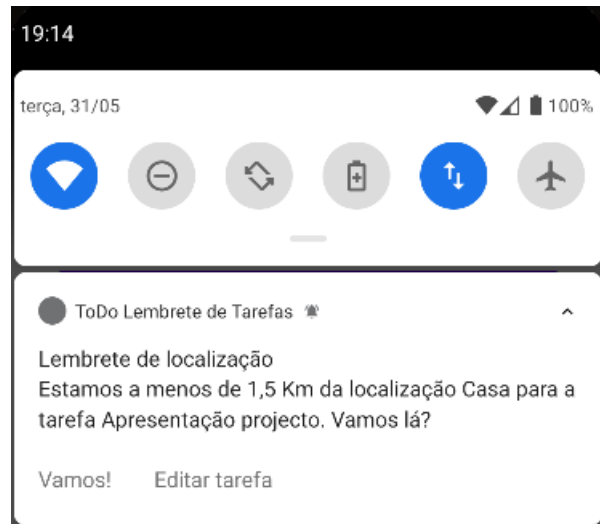


Figura 3.12: Aparência de uma notificação a recordar a tarefa por local.

3.6.3 Reagendamento de tarefas

Quer se trate de uma notificação por local ou por data, em ambos os casos, a *worker thread* que é agendada, não subsiste ao reiniciar do dispositivo. Como tal, todos os agendamentos perder-se-iam. De forma a resolver este problema, é solicitado ao sistema operativo que lance um evento a informar que o dispositivo foi reiniciado. Ao receber este evento, a *App* procede ao reagendamento dos locais e datas de todas as tarefas que não se encontrem marcadas como concluídas.

3.7 Definições

Neste ecrã ficou definido que o utilizador poderá fazer modificações à aparência da *App*. Em particular, modificar a cor de fundo, assim como a cor dos botões. Estas alterações de cor vão aplicar-se a todos os ecrãs.

Para seleccionar a cor pretendida, é disponibilizada ao utilizador uma paleta de cores, bem como três caixas de texto onde é possível introduzir, em alternativa, o código *Red-Green-Blue* (RGB). A figura 3.13 ilustra ambos os ecrãs aonde a escolha da cor é realizada.



Figura 3.13: Aparência dos ecrãs onde é possível escolher qual a nova cor, e aonde aplicá-la.

Capítulo 4

Conclusões e Trabalho Futuro

Neste capítulo enumeram-se algumas conclusões do trabalho efetuado e aspetos de desenvolvimento futuro.

Apesar de na óptica do utilizador, esta *App* aparentar ser de pequena dimensão, mais de 6500 linhas de código foram escritas. E num software desta dimensão, foi possível concluir que é muito difícil implementar logo na primeira tentativa a versão final do código. Seguindo uma certa linha de raciocínio, por vezes conclui-se que a mesma é de difícil interação com outros troços do código já implementados, por melhor que fosse o desenho inicial. Tal levou a redesenhar certas partes de serviços de modo a simplificar a sua implementação.

Da mesma forma, quando se utiliza uma nova biblioteca, por vezes é possível chegar à conclusão que a forma de operação da mesma não é compatível com as funcionalidades inicialmente consideradas. Esta é mais uma situação que leva a que seja pensado novamente um serviço ou funcionalidade que aparentemente já se encontrava perfeitamente definido.

De destacar também a opinião de pessoas mais experientes na área, que podem por vezes dar sugestões de enorme impacto positivo. Como foi o caso de uma sugestão apresentada pelo arguente deste projecto relativamente à notificação por local, que me deu a conhecer o serviço *Geofence*. Este serviço revelou-se significativamente mais adequado do que o serviço previamente definido.

4.1 Trabalho Futuro

Num projecto desta natureza, qualquer trabalho futuro deve assentar no conjunto de apreciações dos utilizadores sobre a aplicação. A *App* ficou a funcionar como era pretendido, com todas as funcionalidades idealizadas.

Está ponderado como trabalho futuro, a criação de uma página *Web* que permite a interação com a *App*. O utilizador poderia criar uma tarefa em qualquer dispositivo que

fizesse uso de um *browser*, para poder criar uma tarefa partilhada e exportá-la para a *App*.

Apêndice **A**

Modelo entidade associação da BD

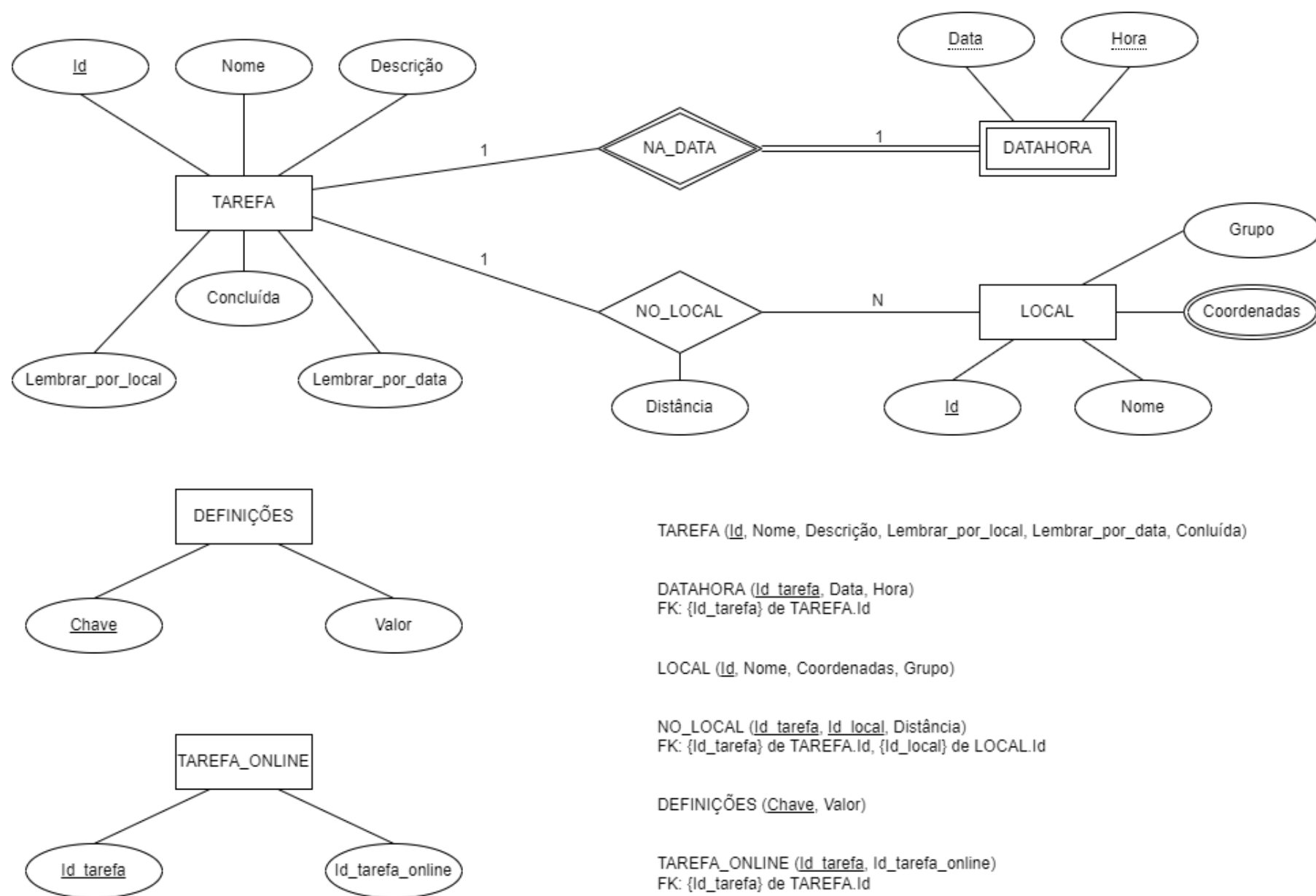


Figura A.1: Modelo entidade associação da BD.

Guia de instalação e utilização

Repositório: ‘<https://github.com/CarlosVazAlves/ToDoTaskReminder>’

Afim de se poder utilizar a *App* correctamente, é necessário possuir um dispositivo com o Android 10 ou superior, e seguir-se os seguintes passos:

- ir ao repositório e fazer o *download* do código para o *Android Studio*;
- obter uma chave para a utilização da *API* do *Google Maps*, de acordo com o tutorial apresentado em *AbhiAndroid (2018)*;
- no ficheiro “local.properties”, deve ser adicionado o seguinte par chave-valor:

`MAPS_API_KEY="chave da API do Google Maps entre aspas";`
- obter uma conta no serviço *Firestore* da *Google*, e seguir todos os passos descritos em *GoogleDevelopers (2022a)* para a sua correcta configuração;
- no *Android Studio*, deve gerar-se o ficheiro *APK* seguindo os seguintes separadores:

Build / Build Bundle(s)/APK(s) / Build APK(s);
- caso o ficheiro *APK* não seja assinado, é necessário dar permissão no dispositivo para poder instalar *APK* não assinados;
- em alternativa, pode ligar-se o dispositivo ao computador, e o *Android Studio* pode instalar directamente a *App* sem ser necessário gerar manualmente o ficheiro *APK*;

- deve-se permitir todas as autorizações solicitadas pela *App* ao longo da sua utilização;
- no caso de pretender criar tarefas a recordar por local, deve-se primeiro criar os locais pretendidos antes de criar uma tarefa;
- ir às “definições” para configurar a *App* ao gosto do utilizador, e está pronto a utilizar.

Referências

- AbhiAndroid (2018). Google maps tutorial with example in android studio. URL: <https://abhiandroid.com/programming/googlemaps> acedido pela última vez a 03/04/2021.
- AndroidDevelopers (2022a). App permissions best practices. URL: <https://developer.android.com/guide/topics/permissions/overview#best-practices> acedido pela última vez a 08/07/2021.
- AndroidDevelopers (2022b). Create and monitor geofences. URL: <https://developer.android.com/training/location/geofencing> acedido pela última vez a 02/07/2021.
- AndroidDevelopers (2022c). Create dynamic lists with recyclerview. URL: <https://developer.android.com/guide/topics/ui/layout/recyclerview> acedido pela última vez a 15/04/2021.
- AndroidDevelopers (2022d). Request app permissions. URL: <https://developer.android.com/training/permissions/requesting> acedido pela última vez a 31/05/2021.
- AndroidDevelopers (2022e). Schedule alarms. URL: <https://developer.android.com/training/scheduling/alarms> acedido pela última vez a 01/05/2021.
- GoogleDevelopers (2022a). Get started with cloud firestore. URL: <https://firebase.google.com/docs/firestore/quickstart> acedido pela última vez

a 08/07/2021.

GoogleDevelopers (2022b). Google maps intents for android. URL: https://developers.google.com/maps/documentation/urls/android-intents#kotlin_5 acedido pela última vez a 03/04/2021.

Imperva (2021). Lazy loading. URL: <https://www.imperva.com/learn/performance/lazy-loading/> acedido pela última vez a 15/04/2021.

nTask (2021). 14 best reminder app(s) for ios and android users. URL: <https://www.ntaskmanager.com/blog/best-reminder-app/> acedido pela última vez a 09/03/2022.

