UNIVERSIDADE LUSÓFONA

Licenciatura Engenharia Informática
2015/2016

Trabalho Final de Curso

Aplicação móvel para Gestão de Processos e Registo de Presenças de uma Clínica de Psicologia

Hermínio Miguel Sobral Tavares
Nº 21304351
Orientador: Prof. Assistente Bruno Cipriano

Índice Geral

Índice de Figuras

Resumo

Este trabalho final de curso teve como objetivo o desenvolvimento de uma aplicação mobile de forma a simplificar os procedimentos de trabalho da Repetição e Diferença Psicologia Clínica LDA. A Repetição e Diferença é uma clínica de psicologia que tem como objetivo garantir o bem-estar psicológico e social dos seus clientes.

A empresa usa as folhas de cálculo do Google Drive para armazenar os dados necessários para a sua atividade laboral. Por exemplo, a empresa tem uma folha onde guarda todos os dados pessoais dos clientes de que necessitam e uma outra onde constam os registos das suas presenças.

Dada a complexidade da interface gráfica das folhas de cálculo utilizadas, este método de trabalho introduz alguma dificuldade no manuseamento destes dados.

De forma a suprimir estas dificuldades, a aplicação foi desenvolvida para o sistema operativo Android (*smartphones* e *tablets*), e permite garantir as mesmas funções até agora realizadas nas folhas de cálculo, mas com maior simplicidade (interface mais amigável) e com maior segurança, devido a se terem implementado validações dos dados que não estavam disponíveis na interface do Google Drive original.

Palavras-Chave: Registo de Presenças, Google Drive, Folhas de cálculo, Android, *Smartphones*, *Tablets*.

Abstract

The aim of this final course work was to develop a mobile application as a way to simplify the working procedures at *Repetição e Diferença Psicologia Clínica LDA*. *Repetição e Diferença* is a Psychology clinic whose aim is to ensure the psychological and social welfare of its clients.

This company uses Google Drive spreadsheets to store the data that are necessary for its labor activity. For example, the company has a spreadsheet where all the clients' necessary data are stored and another one which has the clients' attendance register.

Due to the complexity of the spreadsheets graphical interface, this working method carries some difficulties in handling these data.

In order to overcome these difficulties, the application was developed for the Android operating system (smartphones and tablets) and it guarantees the same functions that have so far been done on the spreadsheets, but in an easier (a more friendly interface) and safer way, because the application performs several data validations which were not implemented by the original Google Drive interface.

Keywords: Attendance Register, Google Drive, Spreadsheets, Android, Smartphones, Tablets.

Agradecimentos

À **minha família,** que sempre estiveram disponíveis em tudo o que precisei. Sempre me apoiaram incondicionalmente em todos os momentos da vida tivessem sido eles bons ou menos bons.

À **minha mulher,** que sempre me apoiou incondicionalmente. Foi a pessoa que mais me apoiou durante estes anos. Sem ti não era possível concluir este marco da minha, nossa vida.

Agradeço principalmente ao professor **Bruno Cipriano**, sem este apoio desta pessoa tanto neste trabalho final de curso como na minha licenciatura nada disto tinha sido possível. Demonstrou sempre uma enorme disponibilidade e um elevado sentido pedagógico nas respostas a todas as minhas questões.
Sempre tive muita vontade que fosse o meu professor orientador no Trabalho de Final de Curso. Para mim foi uma enorme satisfação ver este desejo realizado.

O meu inteiro agradecimento ao **Nuno Margalha**, que demonstrou sempre uma grande disponibilidade nas respostas às questões que se foram levantando à medida que o projeto ia evoluindo.

A todas estas pessoas, muito obrigado.

1. Introdução

Este projeto foi desenvolvido no âmbito da cadeira do Trabalho Final de Curso, na Licenciatura em Engenharia Informática da Universidade Lusófona de Humanidades e Tecnologias para a Repetição e Diferença - Psicologia Clínica, Lda.[1].

Os terapeutas da Repetição e Diferença trabalham com várias folhas de cálculo do Google Drive. Estas folhas são utilizadas para registar diversas informações necessárias para o bom funcionamento da empresa. Por exemplo, guardar os dados pessoais dos clientes, os seus registos de presenças, entre outros aspetos.

Como podemos constatar no anexo A, as folhas de cálculo têm uma interface gráfica pouco amigável. Para além das dificuldades da interface, verificamos também que este sistema carece de certos automatismos. Os dados dos utilizadores não são alvo de nenhuma validação, caso algum terapeuta introduza um NIF com mais ou menos que nove dígitos, as folhas não fazem nenhum alerta sobre o erro.

Assim, o professor Bruno Cipriano contatou-me para elaborar uma aplicação em Android no sentido de melhorar este sistema. Os objetivos deste projeto são construir uma aplicação que fosse substancialmente mais simples e eficaz que as folhas, aumentando assim a probabilidade de ser utilizada pelos terapeutas; e criar automatismos de validação.

A aplicação foi então construída com base nos requisitos acordados e num *design* facultado pelo Nuno Margalha (responsável da empresa a acompanhar o projeto). Esta informação pode ser consultada no anexo B.

Adicionalmente escrevi um documento (anexo C) onde explico o método de acesso às folhas de cálculo do Google Drive. Este guia tem como finalidade ajudar as pessoas a desenvolverem aplicações com base nas folhas de cálculo como eu fiz.

[1] http://www.repeticaoediferenca.com/

2. Enquadramento teórico

2.1 Tecnologias

Nesta secção serão apresentadas as tecnologias envolvidas no desenvolvimento do projeto. Para cada uma delas, é feita a referência também à página *web* da mesma. Esta referência tem como objetivo facultar algumas informações adicionais que em certos casos podem não ser aqui abordadas.

2.1.1 Android

O Android[2] é um sistema operativo baseado em Linux que atualmente é desenvolvido pela Google. Este sistema operativo é desenhado principalmente para terminais móveis sensíveis ao toque, tais como *Smartphones*, *tablets*, entre outros.

Embora o Android tenha sido desenvolvido com vista aos terminais móveis, já é possível denotar a sua presença noutro tipo de equipamentos, tais como por exemplo *SmartTv's* e *SmartPC's*.

Dada a vasta gama de equipamentos onde o Android está presente, este é o sistema operativo para terminais moveis mais utilizado no mundo. Tal como podemos verificar na figura 1, a netmarketshare[3] indica que o Android tem mais de metade da quota de mercado.

[2] https://www.android.com/
[3] https://netmarketshare.com/

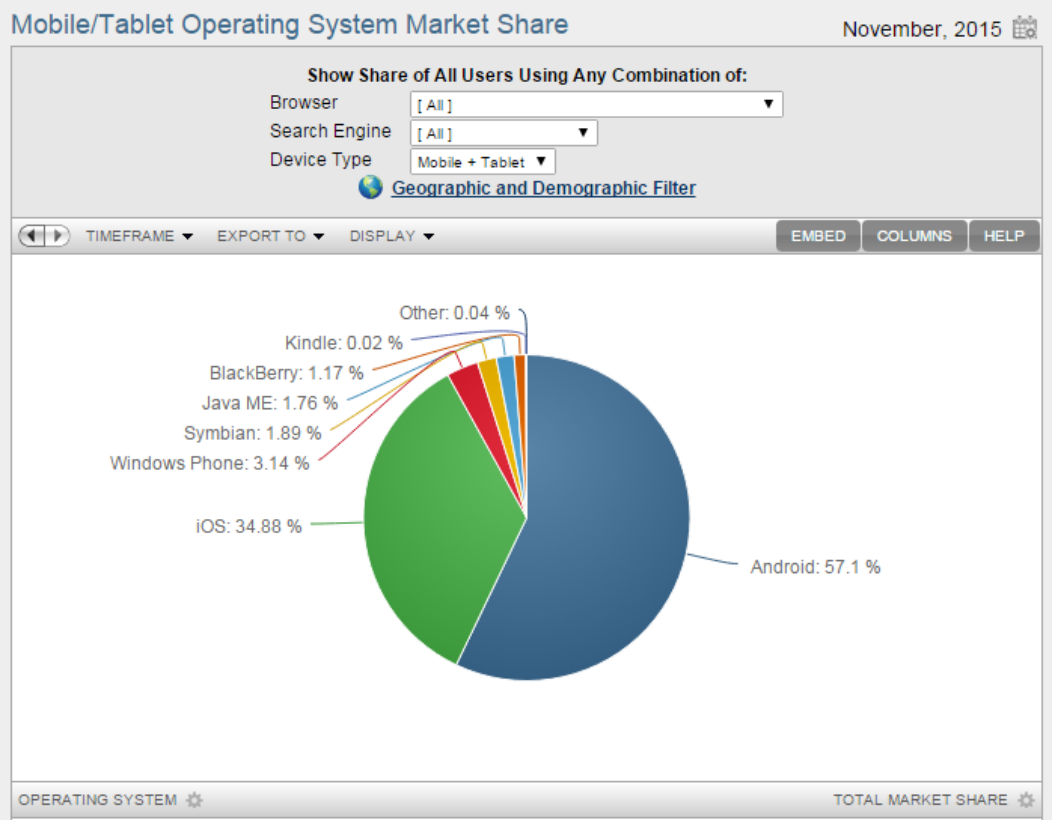
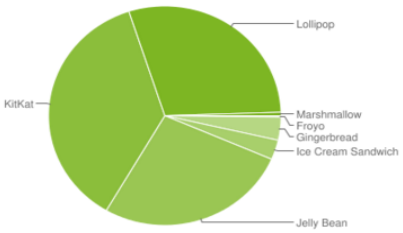


Figura 1 - Quota de mercado

Dada a vasta gama de equipamentos onde o Android se insere e dada a sua evolução natural como sistema operativo, podemos encontrar no mercado diversos equipamentos com diversas versões do Android. A última versão mais recente é denominada por Marshmallow.

De acordo com a Google, o Kitkat ainda é de longe a versão mais utilizada do Android (figura 2). A utilização desta tecnologia foi um requisito da empresa para o projeto.

Version	Codename	API	Distribution
2.2	Froyo	8	0.2%
2.3.3 - 2.3.7	Gingerbread	10	3.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.9%
4.1.x	Jelly Bean	16	10.0%
4.2.x		17	13.0%
4.3		18	3.9%
4.4	KitKat	19	36.6%
5.0	Lollipop	21	16.3%
5.1		22	13.2%
6.0	Marshmallow	23	0.5%



Data collected during a 7-day period ending on December 7, 2015.
Any versions with less than 0.1% distribution are not shown.

Figura 2 - Quotas das diferentes versões do Android

2.1.2 Google Drive

O Google Drive[4] é um serviço de armazenamento em *cloud* apresentado pela Google em 2012. Sendo o Google Drive baseado no conceito de *cloud*, este intrinsecamente faculta a possibilidade dos seus utilizadores terem acesso aos seus ficheiros através de um computador ou de um terminal móvel compatível.

Este serviço, além de disponibilizar um armazenamento de forma segura aos seus utilizadores, tem também um complemento denominado de Google Docs.

O Google Docs oferece aos seus utilizadores a possibilidade de utilizarem em conferência a consulta, escrita e edição de documentos de texto, folhas de cálculo, apresentações, entre outras funcionalidades.

O Google Drive é um serviço disponibilizado gratuitamente a qualquer pessoa que tenha uma conta Google. Embora seja disponibilizado gratuitamente com 15GB de armazenamento, podemos aumentar esta capacidade mediante um determinado custo.

À semelhança da tecnologia Android, a utilização do Google Drive também foi um requisito da empresa para o projeto.

2.1.3 Java

O Java é uma linguagem de Programação Orientada por Objetos (POO) criada na década de 90 por uma equipa cujo responsável era James Gosling. O código Java quando compilado para bytecode não fica automaticamente disponível para ser executado. Este bytecode é posteriormente executado sobre o JVM (*Java Virtual*

[4] https://www.google.com/drive/

Machine). O JVM é uma máquina virtual criada para correr o código Java tornando assim esta tecnologia multiplataforma.

No caso do Android esta máquina virtual chama-se Dalvik e o bytecode com que ela trabalha não é exatamente o mesmo que o JVM. A Dalvik está otimizada para equipamentos com poucos recursos como é o caso dos dispositivos móveis.

O Java é a linguagem de programação com mais presença neste TFC. Esta linguagem foi utilizada para definir todos os comportamentos da aplicação face às ações do utilizador.

De acordo com a TIOBE, a linguagem de programação Java foi considerada mais pesquisada na Internet em 2015[5]. O Java já tinha vencido este prémio há exatamente 10 anos atrás.

2.1.4 XML

O XML (Extensible Markup Language) é uma linguagem de programação que funciona à base de *tags* assim como o HTML, CSS, entre outras. Neste TFC o Java foi utilizado para definir o comportamento da aplicação face às ações do utilizador, o XML foi utilizado para construir o *layout* de toda a aplicação.

2.1.4 Google Sheets API

Esta API, também conhecida como Google Spreadsheets API[6], permite ao programador desenvolver aplicações *client-side* que conseguem ler e modificar folhas

[5] http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html
[6] https://developers.google.com/google-apps/spreadsheets/

de cálculo do Google Drive. Com a Google Sheets API é possível ler, escrever e editar dados a diversos níveis. Estes níveis são os seguintes:

- **SpreadSheets** – Estes ficheiros são aqueles que são criados através da interface do GoogleDrive ou através da própria API. Estes ficheiros são semelhantes aos *books*[7] existentes no MS Excel.

- **WorkSheets** – As worksheets ou folhas no MS Excel são coleções de células ordenadas por linhas e colunas.

- **Cells** – Cells ou células são os pares ordenados (colunas x linhas) onde conseguimos guardar os dados.

2.1.5 Android Studio

O Android Studio[8] é um IDE (*Integrated Development Environment*) baseado no IntelliJ IDEA e é recomendado pela Google para o desenvolvimento de aplicações em Android.

O Android Studio permite ao programador testar a sua aplicação nas diversas distribuições de Android disponíveis no mercado. Com esta funcionalidade o programador fica isento de ter um equipamento com o sistema operativo Android ou diversos equipamentos com diferentes versões do mesmo.

Além do emulador, o Android Studio permite também que possamos testar o *layout* da nossa aplicação em equipamentos de diferentes tamanhos. Assim podemos garantir que a nossa aplicação funciona em qualquer equipamento Android independentemente do tamanho do seu ecrã.

[7] Um *book* é um ficheiro que contém diversas folhas de cálculo.
[8] http://developer.android.com/sdk/index.html

A escolha deste IDE em detrimento do Eclipse deve-se ao facto da Google o recomendar para o desenvolvimento de aplicações.

2.1.6 GitHub

O GitHub[9] é uma implementação web de um sistema de controlo de versões chamado Git. Uma das principais funcionalidades deste serviço é possibilitar a edição de ficheiros por múltiplas pessoas em simultâneo.

Isto é possível porque o Git garante que as alterações efetuadas não colidem. Podemos imaginar o quão difícil seria sem o Git fazer com que duas ou mais pessoas trabalhassem no mesmo ficheiro sem que existissem colisões.

O GitHub está disponível gratuitamente mas apenas com a possibilidade de criarmos repositórios públicos. Existe um pacote dedicado a estudantes[10] e professores que é disponibilizado também gratuitamente com a possibilidade de criar até cinco repositórios privados.

2.1.7 Pinta

O Pinta é uma ferramenta gratuita de desenho e edição de imagens desenvolvida para ambientes multiplataforma (Windows, Mac e Linux). Esta aplicação é muito parecida ao Paint da Microsoft, o que torna bastante simples e intuitiva a sua utilização. Esta ferramenta foi utilizada para criar ou editar todas as imagens presentes na aplicação.

[9] https://github.com/
[10] https://education.github.com/pack

3. Método

3.1 Arquitetura do Sistema

A aplicação desenvolvida no âmbito deste TFC utiliza como repositório de dados o Google Drive. Uma vez que todos os dados chegam à aplicação através da Internet, é importante garantir a qualidade desta comunicação.

O desempenho da aplicação pode ficar comprometido pela velocidade do acesso à Internet. Neste caso, para uma melhor experiencia, é recomendado que esta comunicação tenha a melhor qualidade possível.

Figura 3 - Arquitetura do sistema

3.2. Desenvolvimento das Activities

Neste capítulo serão apresentadas cada uma das activities envolvidas na aplicação, mas antes disso temos fazer uma pequena introdução à classe `Activity`[11]. A classe `Activity` é responsável por gestão da IU (Interface de Utilizador). Sem esta classe não conseguíamos corresponder aos movimentos do utilizador no ecrã do equipamento.

Dito isto, podemos compreender que obrigatoriamente, todas as aplicações em Android começam com uma `Activity`. Na realidade o que vamos ter é uma classe que vai estender a classe `Activity`. De acordo com os mecanismos de herança, esta nova classe vai receber todos os comportamentos da classe `Activity`, para além dos comportamentos específicos implementados por nós.

As activities presentes nesta aplicação são:

- Login
- Notificações (e *E-mails)*
- Calendário e Registo de Presenças
- Processos

3.2.1. Activity de Login

Esta activity é responsável por fazer a autenticação do utilizador através da sua conta Google. Se a conta pretendida não estiver configurada no equipamento, é disponibilizada a funcionalidade de configurar a mesma. Só as contas Google que tenham acesso às folhas do Google Drive da empresa poderão autenticar-se. Às restantes, será exibida uma mensagem de erro com a informação de que não está autorizado a fazer acesso à aplicação.

Caso seja a primeira vez que esta conta Google se esteja a tentar autenticar, é apresentada uma mensagem ao utilizador para que ele possa permitir o acesso às suas

[11] http://developer.android.com/guide/components/activities.html

folhas do Google Drive. Para que a autenticação prossiga, é necessário que o utilizador aceite este pedido.

Assim que este pedido for aceite vamos começar por fazer uma pesquisa a todos os *books* da conta Google selecionada. Esta pesquisa é efetuada com base no identificador único de cada *book*. Este identificador está disponível nos *links* do ficheiro em questão.

```
https://docs.google.com/spreadsheets/d/1QxLxCvhYGuH3vZ-CqRpJzL5ihhyLri8kGvSjwD3VU0c/
```

As funcionalidades da aplicação só estarão disponíveis ao utilizador se este tiver acesso aos *books* dos quais as seguintes folhas fazem parte:

- Processos 15/16
- Folha de Presenças 15/16
- Notificações
- Equipas

O processo de autenticação inicia-se com a construção de uma lista, esta irá conter todos os *books* existentes na conta Google escolhida. Com recurso a esta lista temos de fazer uma nova pesquisa de forma a pesquisarmos as folhas de que necessitamos.

Se todas as folhas forem encontradas, a pesquisa é interrompida e o utilizador terá acesso à aplicação. Se esta pesquisa falhar, será apresentado ao utilizador uma mensagem de erro com essa indicação.

Na figura 4 está esquematizado o fluxograma do processo de autenticação acima descrito.

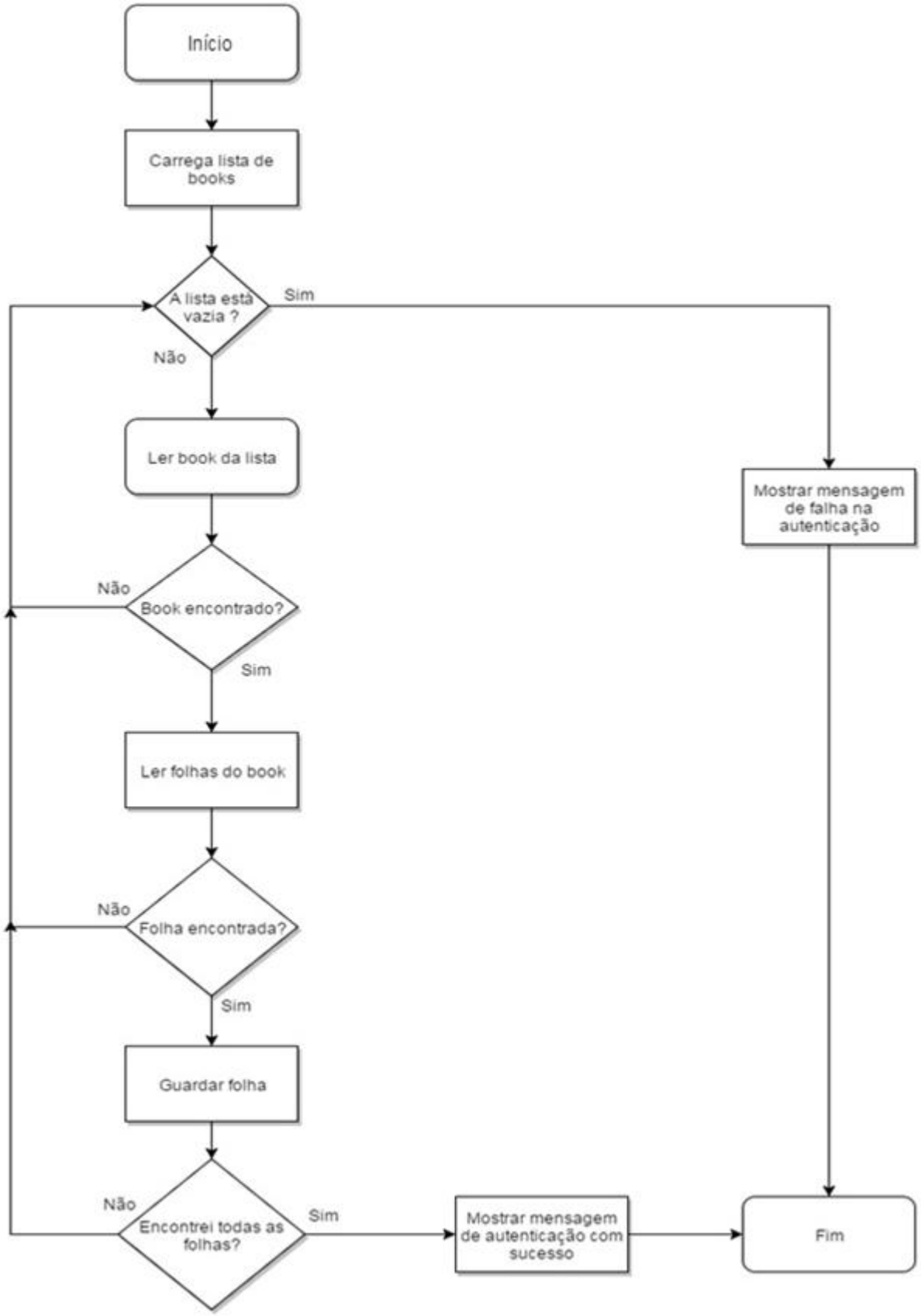


Figura 4 - Fluxograma do processo de autenticação

3.2.2. Activity das Notificações

Assim que o utilizador se autentica, esta activity vai ler os dados existentes na folha das notificações. As notificações são exibidas em forma de tabela constituída por duas linhas e uma coluna. Na primeira linha está o título da notificação e na segunda a mensagem propriamente dita.

Todos os utilizadores desde que autenticados têm acesso a todas as notificações da empresa. A empresa é quem gere o período de visibilidade das notificações assim como o conteúdo. Esta gestão é efetuada através do Google Drive.

Além das notificações, também estão disponíveis dois botões que permitem ao utilizador enviar *e-mails*. Estes endereços já estão pré-definidos na aplicação.

A figura 5 ilustra este modelo em forma de fluxograma.

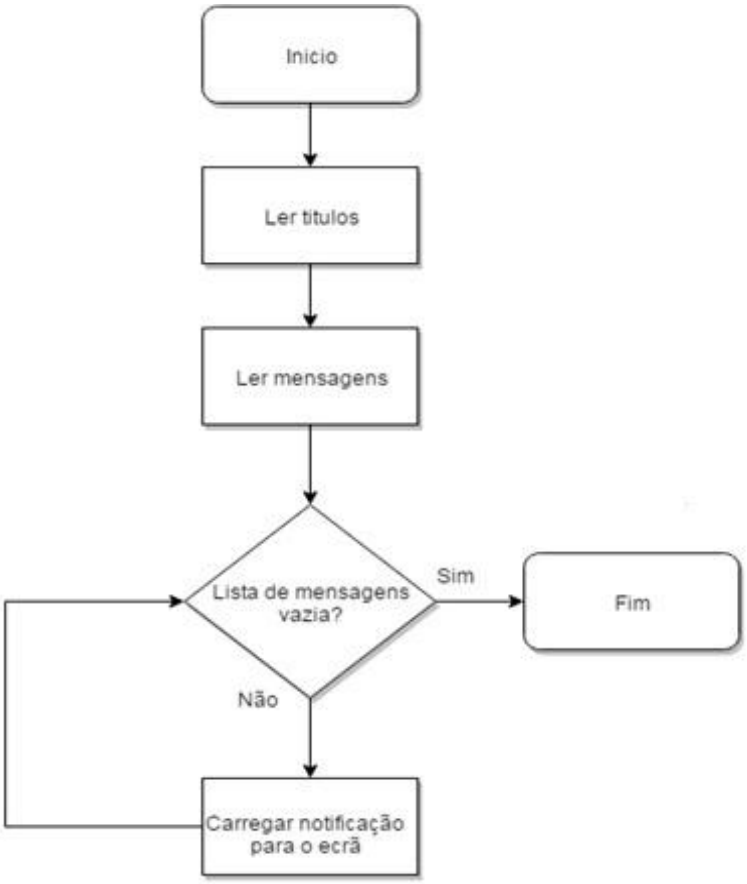


Figura 5 - Fluxograma do processo de leitura das notificações

3.2.3. Activity do Calendário e Registo de Presenças

Existe uma dependência direta entre estas duas activities. É através do calendário que é possível escolher a data que pretendemos ler ou marcar as presenças (registo).

A data escolhida no calendário aparece na primeira linha da tabela. Esta linha é constituída também por outros três botões. Estes botões atribuem a todos os clientes três estados de presença:

- P – Presente
- FC – Falta Cliente
- FT – Falta Terapeuta

Se um terapeuta faltar durante um dia inteiro, podemos atribuir essa falta a todos os clientes se selecionarmos o botão "FT".

O registo de presenças também pode ser definido individualmente para cada cliente. Cada cliente tem à frente do seu nome os três estados de presença existentes, assim, ao pressionar um destes botões o estado da presença selecionado ficará atribuído apenas ao cliente em questão.

Estes botões estão a ser representados através de checkboxes. O *layout* destas foi alterado com o sentido de assumirem um aspeto visual semelhante ao das folhas do registo de presenças do Google Drive.

Os nomes e o registo de presenças dos clientes estão representados numa `GridView`[12]. Esta `GridView` teve de ser reconstruida parcialmente porque o Android não suporta nativamente GridViews cujas linhas têm três checkboxes cada.

Durante o processo de leitura e escrita do registo de presenças é necessário que a associação cliente – presença esteja garantida. Esta associação foi efetuada com o recurso a uma classe.

Nas Figuras 6 e 7 estão esquematizados os fluxogramas dos processos de leitura e escrita do registo de presenças.

[12] Uma GridView é uma tabela cujo número de linhas é dinâmico

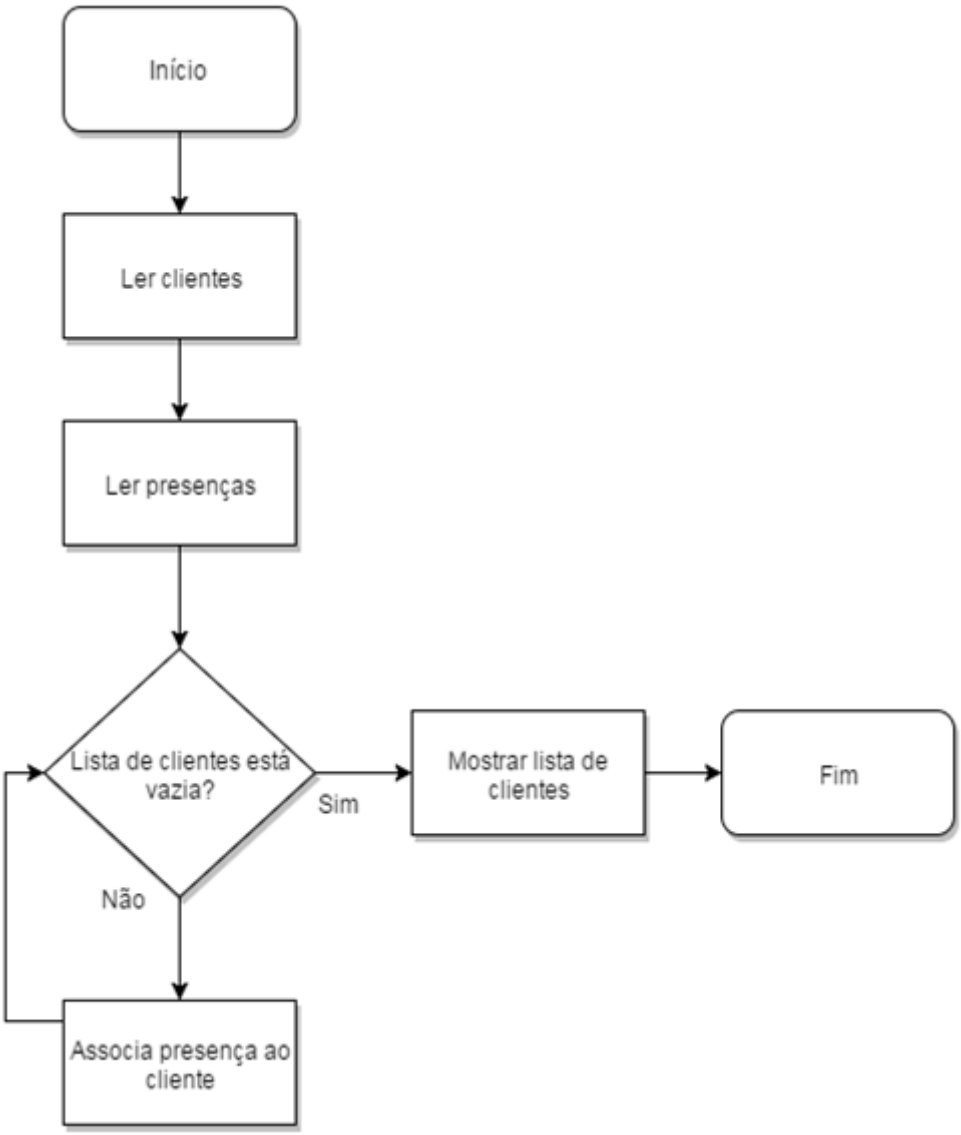


Figura 6 - Fluxograma do processo de leitura do registo de presenças

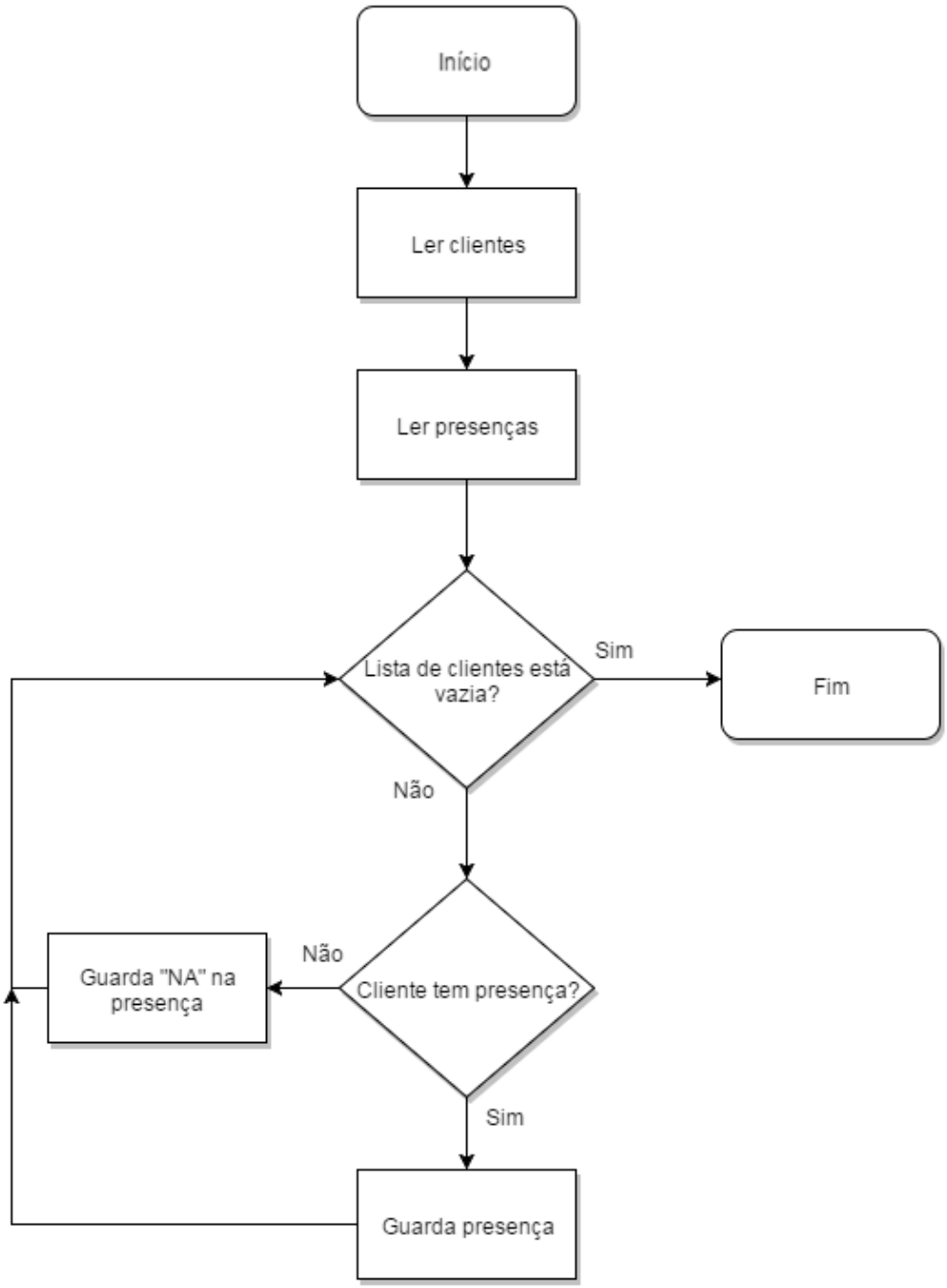


Figura 7 - Fluxograma do processo para guardar os registos de presenças

3.2.4. Activity dos Processos

É através desta activity que os utilizadores conseguirão interagir com a folha dos processos. É possível realizarem inserções, pesquisas e alterações aos processos já existentes.

A pesquisa está disponível de duas formas. Uma delas é através do nome do cliente. Ao começar a escrever esse nome, a aplicação começa a dar sugestões de nomes de clientes já existentes na folha. Para prevenir a hipótese de existirem dois nomes exatamente iguais, as sugestões contêm além do nome a data de nascimento dos clientes.

A segunda forma de efetuar a pesquisa é através do NIF[13]. Este dado é um identificador unívoco de qualquer registo existente na folha dos processos.

A alteração e a inserção são bastante parecidas, contudo a alteração tem uma pequena particularidade. Todos os dados existentes nesta activity podem sofrer alterações menos o NIF. Este não pode ser alterado de forma a salvaguardar a integridade da pesquisa. Por este motivo, este dado deverá ser alterado diretamente na folha de cálculo do Google Drive.

De forma a salvaguardar a integridade dos dados durante uma escrita na folha, antes dos dados serem guardados é efetuada uma validação. O objetivo é verificar, mesmo antes da escrita, se existiu alguma alteração na linha que irá receber os novos dados. Se não existiu nenhuma alteração, os dados são automaticamente guardados na folha. Caso tenha existido alguma alteração, será apresentada uma mensagem ao utilizador a informar que os dados existentes na linha em questão sofreram uma alteração. O utilizador pode assim optar por fazer ou não a escrita dos dados[14].

Os fluxogramas da pesquisa, inserção e alteração estão disponíveis nas Figuras 8, 9 e 10.

[13] NIF – Número de Identificação Fiscal
[14] Para mais detalhes consultar “3.3.3. Gestão de Concorrência da escrita” na página 30

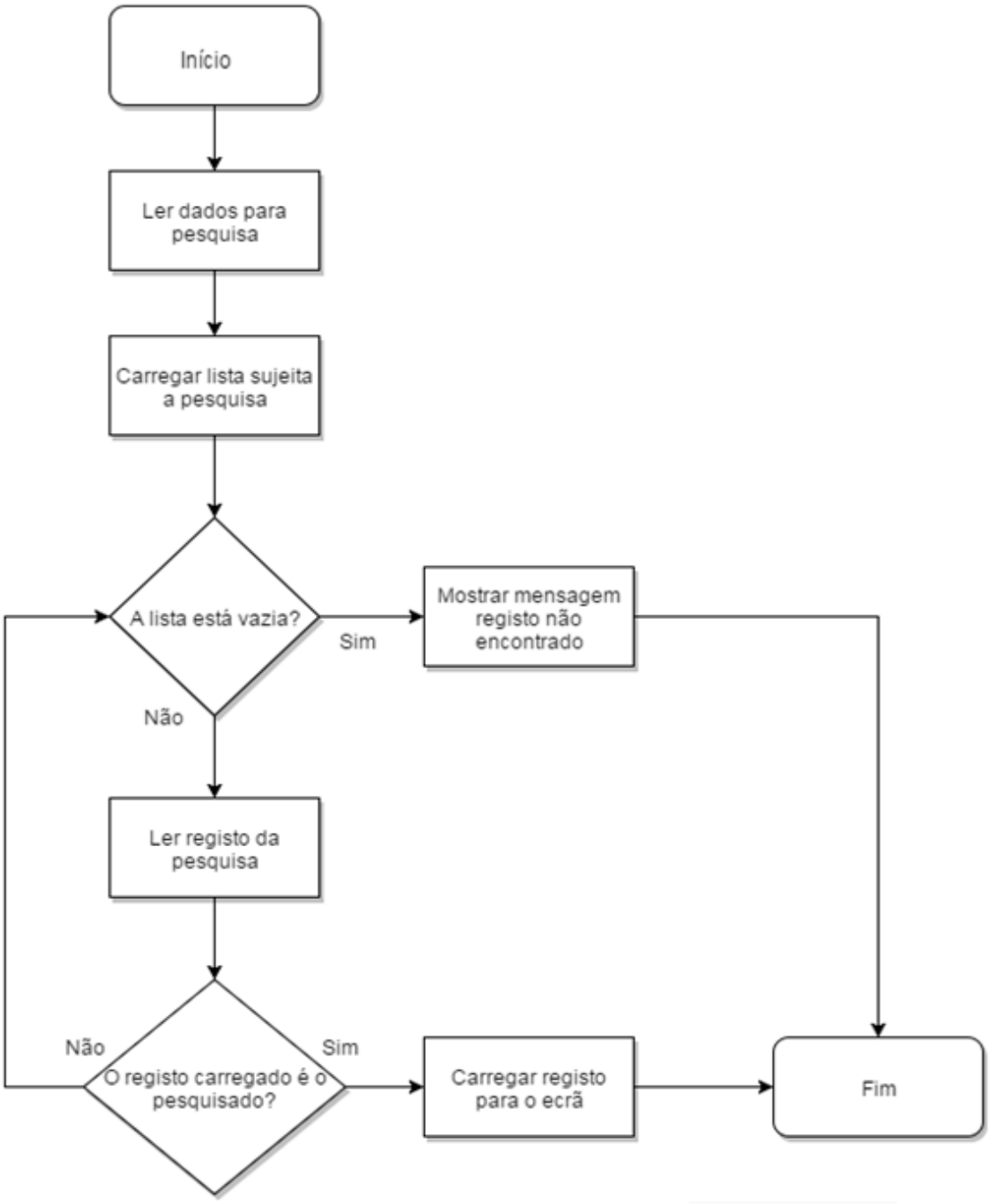


Figura 8 - Fluxograma do processo de pesquisa

Se repararmos nas duas figuras abaixo (8 e 9), podemos verificar que a estrutura do algoritmo da alteração é igual à da inserção de um novo caso. Isto quer dizer que podemos reutilizar o código da inserção e implementar a alteração.

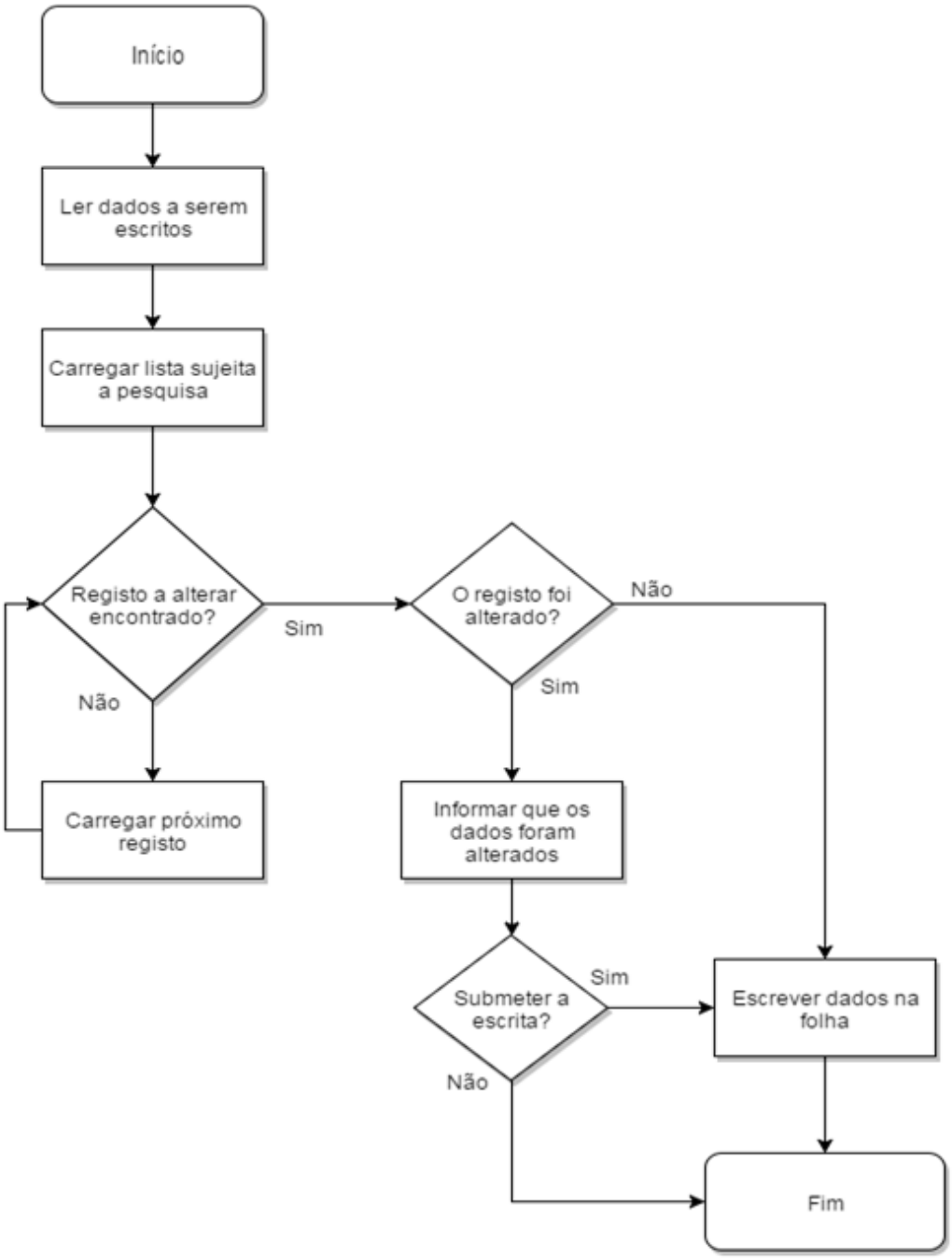


Figura 9 - Fluxograma do processo de alteração

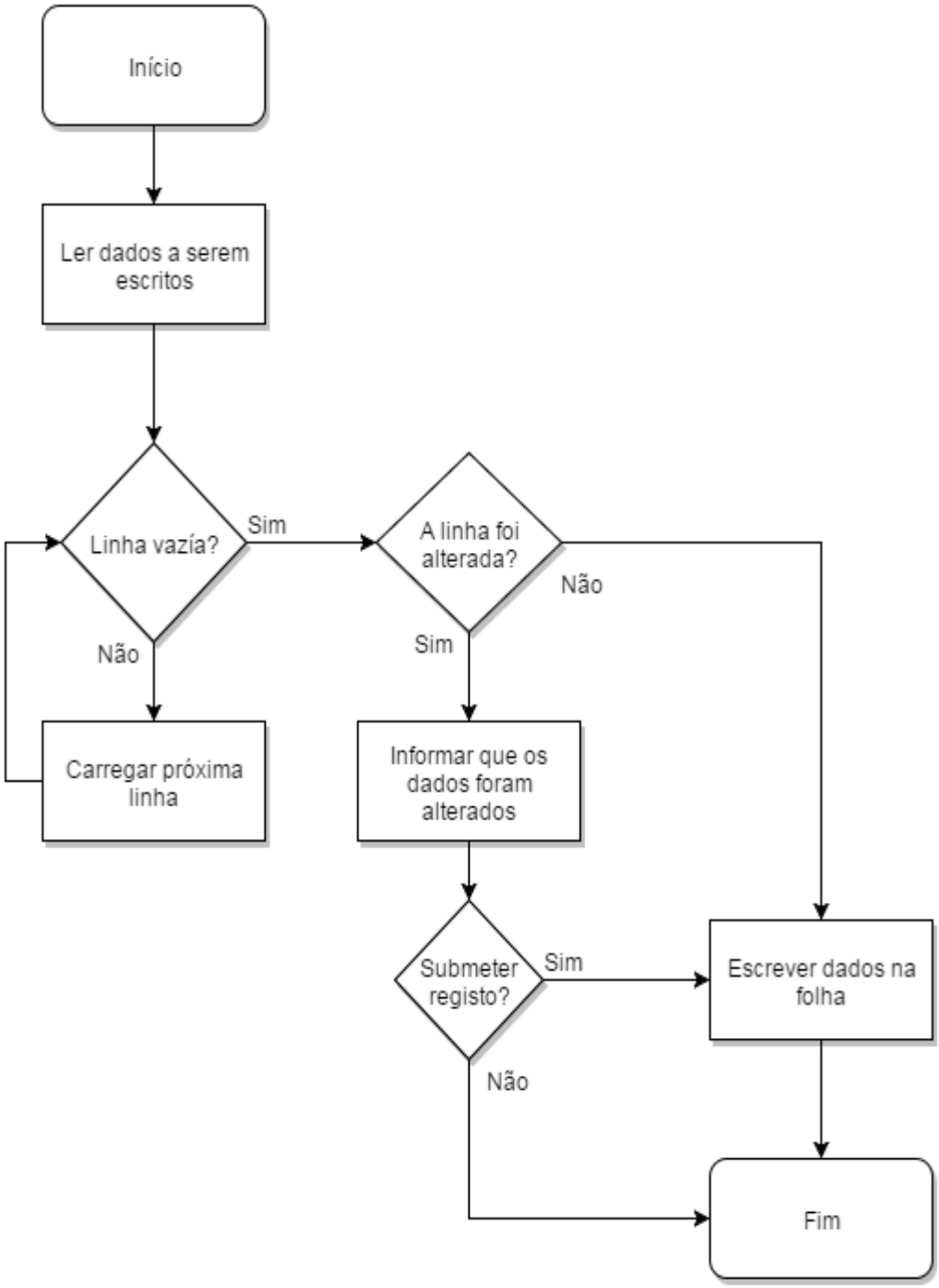


Figura 10 - Fluxograma do processo de inserção

3.3. Técnicas e Algoritmos

Aqui vão ser retratados com mais algum rigor as técnicas e os algoritmos utilizados na aplicação.

3.3.1. AsyncTask

O Android tem uma *thread* responsável pela gestão da IU (Interface do Utilizador). Uma das recomendações de boas práticas no desenvolvimento de aplicações para Android, é não sobrecarregar esta *thread*. Se isto acontecer, a aplicação terá um comportamento bastante instável.

Como tipicamente é complicado gerir código concorrente, o Android possui uma classe que tem como objetivo colmatar esta dificuldade.

A `AsyncTask`[15] é uma classe abstrata que tem como objetivo facilitar a implementação de código concorrente.

Como a comunicação com Google Drive é exigente a nível do processamento, senti a necessidade de utilizar os métodos desta classe sempre que precisava de comunicar com as folhas.

Dos vários métodos que a classe `AsyncTask` tem, fiz `Override` destes três:

`onPreExecute`	O código é executado na *thread* IU, é executado imediatamente antes do método `doInBackGround`. Utilizei este método para mostrar um ecrã de *loading* ao utilizador.
`doInBackground`	Este método cria a *thread* onde o nosso código concorrente vai ser executado. Este método foi utilizado para fazer a comunicação com as folhas, quer seja de escrita quer de leitura.
`onPostExecute`	À semelhança do método `onPreExecute`, este também é executado na *thread* de UI. Este código só será executado após o `doInBackground`. A maior parte das vezes este método foi utilizado para desativar o ecrã de *loading* inicialmente mostrado ao utilizador.

[15] http://developer.android.com/reference/android/os/AsyncTask.html

3.3.2. Otimização da leitura

Quando comecei a fazer o primeiro processo de leitura das folhas, apercebi-me de que ia ter de resolver um problema. Esse problema estava relacionado com o uso excessivo de memória durante a leitura das folhas. Desta forma concluí que ler a folha na sua íntegra não era a melhor solução.

Na Figura 11 temos o gráfico relativo à alocação de memória durante a leitura da folha inteira. Este gráfico foi gerado pelo Android Studio.

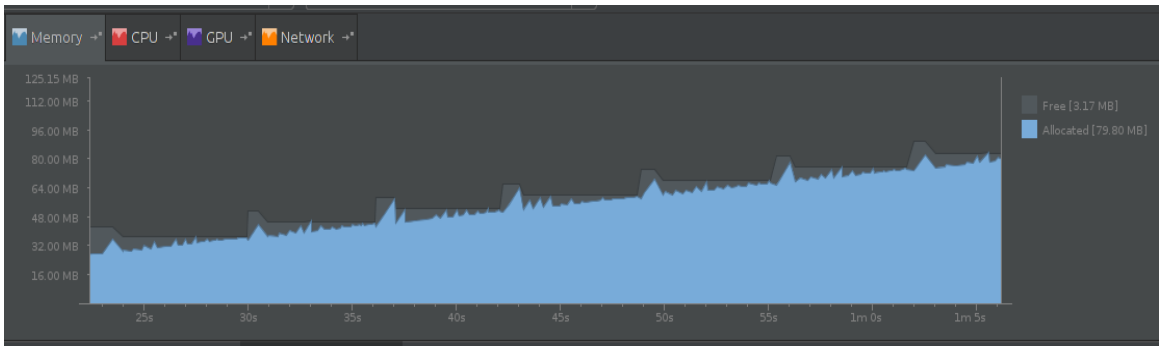


Figura 11 - Gráfico da memória sem otimização

Foi então que resolvi verificar se existia alguma forma de ler apenas as células que necessitava a cada pedido do utilizador. Verifiquei então que na documentação da Google existia a possibilidade de restringir a leitura das células através de uma String.

Na Figura 12 está o gráfico da alocação de memória, este gráfico mostra novamente a alocação de memória após a otimização do processo de leitura.

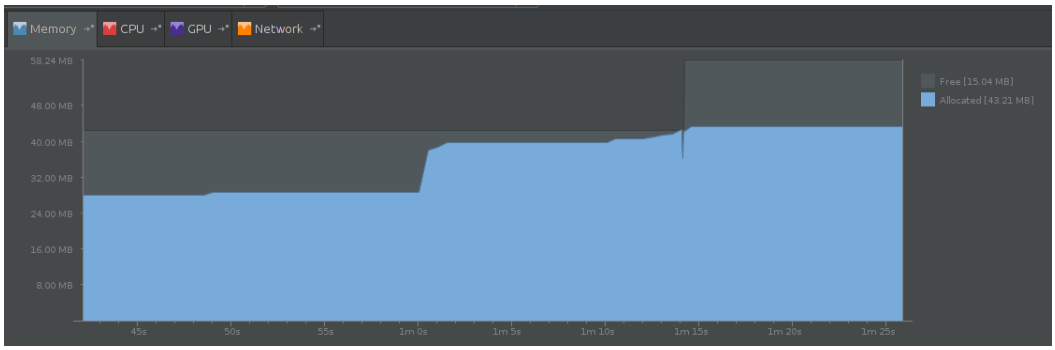


Figura 12 - Gráfico da memória com otimização

Assim que comecei a fazer restrições ao nível da linha e da coluna, verifiquei que a aplicação começou a consumir muito menos memória. Além desta vantagem,

ficou também substancialmente mais rápida. Este acréscimo deve-se ao facto de ter deixado de ler as folhas na sua totalidade. Passei agora a ler apenas as partes que necessitava face aos pedidos do utilizador.

3.3.3. Gestão de Concorrência da escrita

Uma vez que as folhas podem ser escritas através da aplicação e mesmo diretamente no Google Drive, era muito importante garantir que durante a escrita não íamos apagar dados já inseridos ou alterados sem o nosso conhecimento. Se dois utilizadores, um na aplicação e outro no Google Drive, estiverem a concorrer pelo mesmo processo, é importante que na aplicação o utilizador tenha essa informação.

Criei um mecanismo que avisa o utilizador caso exista alguma alteração aos dados que não seja do conhecimento de quem está a utilizar a aplicação. Por exemplo, se um utilizador estiver a alterar um processo na aplicação, e um outro fizer uma alteração ao mesmo processo diretamente no Google Drive, após fazer o pedido de escrita na aplicação, o utilizador recebe uma mensagem a indicar que os dados deste processo foram alterados. Nesta fase, o utilizador tem a possibilidade de avançar com a sua alteração ou cancelá-la.

Esta validação é efetuada através da comparação de duas datas. Quando o utilizador faz uma pesquisa, a aplicação guarda a data da última alteração que ocorreu ao conjunto de células lido. Quando o utilizador faz a alteração e pretende guardar a mesma, é efetuada uma nova leitura da data da última alteração desse conjunto de células afetado.

Se as datas forem iguais, a escrita é efetuada automaticamente. Se as datas forem diferentes, é então apresentada a mensagem ao utilizador com a indicação de que os dados sofreram uma alteração desde que ocorreu a última leitura.

3.3.4. Menu de navegação

O menu de navegação está presente em todas as activities, exceto na de login. Verifiquei então que a solução ideal seria construir este menu numa activity em separado. Assim as outras activities só tinham de o referenciar de alguma forma.

Após algumas pesquisas, verifiquei que é possível fazer a inclusão do código XML através de outros ficheiros. Assim, construi o *layout* do menu de navegação e inclui-o em todos os ficheiros XML das restantes activities.

Com a solução ao nível do *layout* implementada, tinha de fazer o mesmo para o código Java, de forma que os métodos da barra de navegação funcionassem. Decidi então implementar um esquema de herança. Estendi todas as activities e assim já passei a ter a barra de navegação a funcionar de forma uniforme com o mínimo de código possível.

No diagrama de classes da figura 15 podemos consultar este modelo.

3.3.5. Suporte *multiscreen*

O Android é um sistema operativo que está presente em equipamentos com os mais diversos tamanhos de ecrã. O fato de existirem tamanhos diferentes vai exigir que a aplicação funcione de igual forma em todos os equipamentos, independentemente do tamanho do ecrã. Para que a aplicação se adapte aos vários ecrãs, para além do seu tamanho, há que levar também em consideração as diferentes densidades de pixéis dos equipamentos.

A Google classifica os ecrãs e as densidades em quatro tipos:

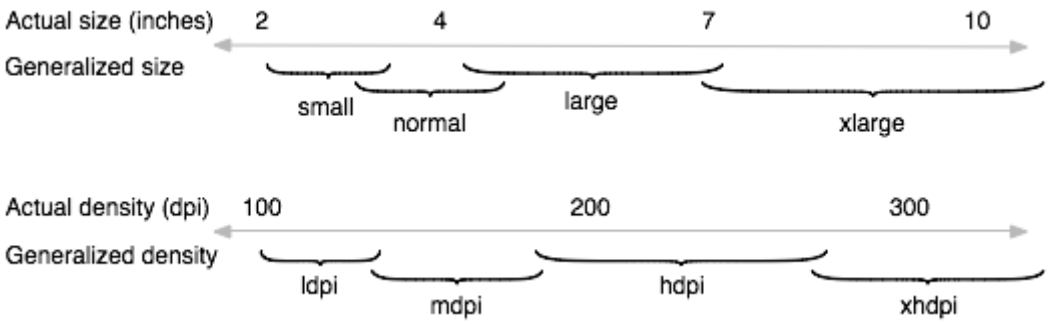


Figura 13 - Tamanhos e densidades

Para termos uma aplicação que se adapte a qualquer ecrã, temos de relacionar todos os tamanhos com as diversas densidades de pixéis. De acordo com as boas práticas estas dimensões têm de ser definidas num ficheiro .xml para cada relação ecrã x densidade.

Para fazer esta implementação recorri a um repositório do GitHub[16]. Este projeto denominado de "sdp" tem como propósito criar uma medida universal a todos os equipamentos. Em vez de definirmos as dimensões em dp (*density-independent pixel*) fazemos em sdp (*scalable dp*).

3.3.6. Interfaces Serializable e Parcelable

Tipicamente as aplicações Android são constituídas por duas ou mais ativities. Isto leva-nos ao seguinte problema: como fazemos para transferir objetos entre activities? O Android dá-nos a possibilidade de fazer esta transferência através da classe `Intent`[17].

Uma `Intent` é uma classe que nos permite despoletar determinadas ações recorrendo ao sistema operativo. Por exemplo, se quisermos iniciar uma `Activity`, ligar o Bluetooth ou abrir o cliente de *e-mail*, temos de recorrer a uma `Intent`.

A classe `Intent` não permite a transferência das referências dos objetos entre activities se estas classes forem criadas pelo programador, o que acontecia nesta aplicação (ver secção 3.3.9. Modelo de Dados). Para estas classes temos que implementar a interface `Serializable`[18] ou a `Parcelable`[19]. Ambas as interfaces cumprem a mesma função, contudo é recomendado a utilização da `Parcelable`. Embora a sua implementação seja mais complexa (pois é necessário implementar 5 métodos), como podemos verificar na figura 14, a diferença do desempenho é considerável.

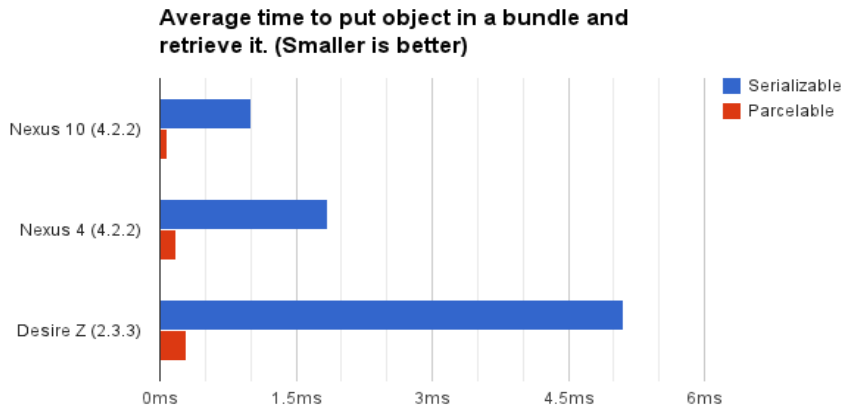


Figura 14 - Gráfico de desempenho Serializable vs Parcelable

[16] https://github.com/intuit/sdp
[17] http://developer.android.com/reference/android/content/Intent.html
[18] http://developer.android.com/reference/java/io/Serializable.html
[19] http://developer.android.com/reference/android/os/Parcelable.html

Existem ferramentas que nos permitem lidar com essa complexidade extra, por exemplo, o Parcelaber[20] que é capaz de gerar o código do `Parcelable` tendo como referência uma classe defina pelo programador.

3.3.8. Classe Folha

De forma a que o código fique mais organizado, criei a classe Folha para que tenhamos tudo o que diz respeito à comunicação entre a aplicação e as folhas numa só classe.

Como esta classe serve apenas como utilitário, implementei-a com o construtor privado. Uma vez que a classe folha apenas faz leituras e escritas não faz sentido que esta seja instanciável.

Esta classe tem por exemplo o mapeamento entre colunas e os dados, leituras, escritas, gestão da concorrência da escrita, entre outros métodos necessários para o funcionamento da aplicação.

O mapeamento entre as colunas e os dados, isto é, a correspondência entre número da coluna e o tipo de dados que essa coluna guarda, está feita nos atributos da classe. Por exemplo, na folha dos processos, a coluna 32 faz corresponder ao NIF dos clientes.

```
private static final int COLUNA_NIF_CLIENTE = 32;
```

A classe folha implementa também uma espécie de cache para melhorar o desempenho da aplicação. Esta funcionalidade goza da localidade temporal dos dados na folha da equipa, isto é, como o nome dos terapeutas, das escolas e estados não sofrem alterações constantes, estes dados ficam guardados em memória. Estes são armazenados em ArrayLists e são carregados com dados das folhas se estiverem vazios.

```
 private static ArrayList<String> escolas = new ArrayList<>();
```

[20] http://www.parcelabler.com/

Além da aplicação ficar mais rápida, como não vamos ler tantas vezes aqueles dados das folhas, poupamos o tráfego de dados ao utilizador. Dado que a aplicação vai funcionar em *smartphones* e *tablets*, é importante que consuma o mínimo possível de dados.

3.3.9. Modelo de Dados

Neste tópico será apresentado o modelo de dados que suporta a aplicação (figura 15). A figura contém um diagrama UML que descreve a forma como as classes implementadas se relacionam.

Neste modelo estão presentes dois tipos de classes, as nativas Android e aquelas que criei tendo em vista a estrutura das folhas da empresa e a realidade da mesma (estas últimas aparecem a bold no diagrama). Das classes que constituem este modelo de dados, só a `Activity` e a `AsyncTask` é que não foram criadas por mim.

A classe Folha foi criada para uma melhor organização do código. É aqui que a aplicação faz a leitura, a escrita e a gestão da concorrência da escrita. As funcionalidades desta classe estão detalhadas com maior rigor na secção 3.3.8.

A `MainMenuActivity` é a classe que implementa o menu de navegação. Esta classe foi criada de forma a reutilizar este código. É possível consultar mais detalhes na secção 3.3.4.

As classes `Notificacao`, `DadosDeCaso` e `Presenca` foram construídas porque é através delas que conseguimos modelar a aplicação à realidade da empresa.

A classe `Presenca` implementa a interface `Parceable` de forma a possibilitar a transferência de dados entre activities tal como explicado na secção 3.3.6.

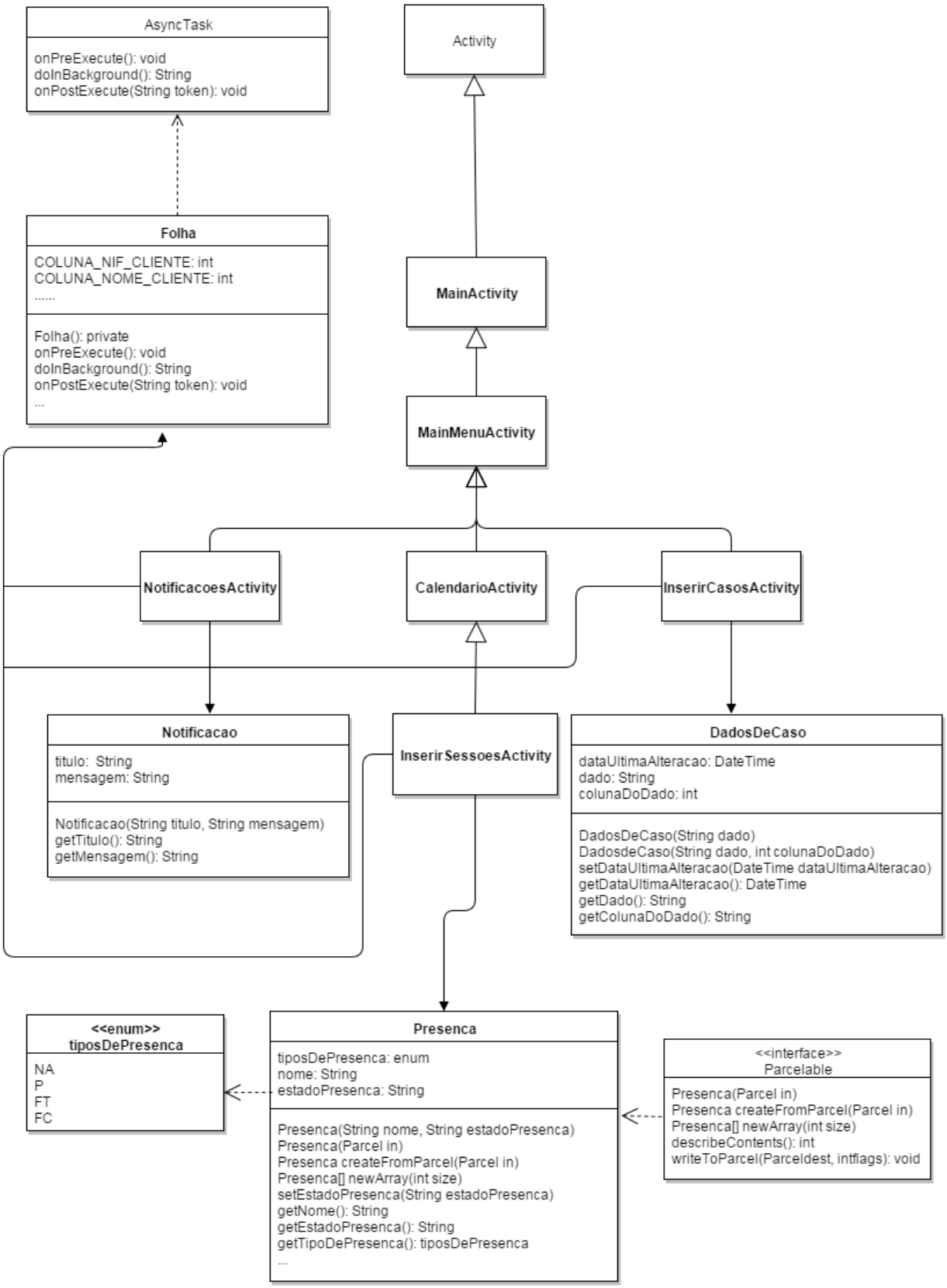


Figura 15 - Diagrama UML do modelo de dados da aplicação

4. Resultados

Este capítulo irá demonstrar algumas imagens da aplicação em funcionamento.

4.1. Activity de Login

Na Figura 16 temos um *screenshot* da activity de login. Esta tem um botão que permite ao utilizador autenticar-se na aplicação.

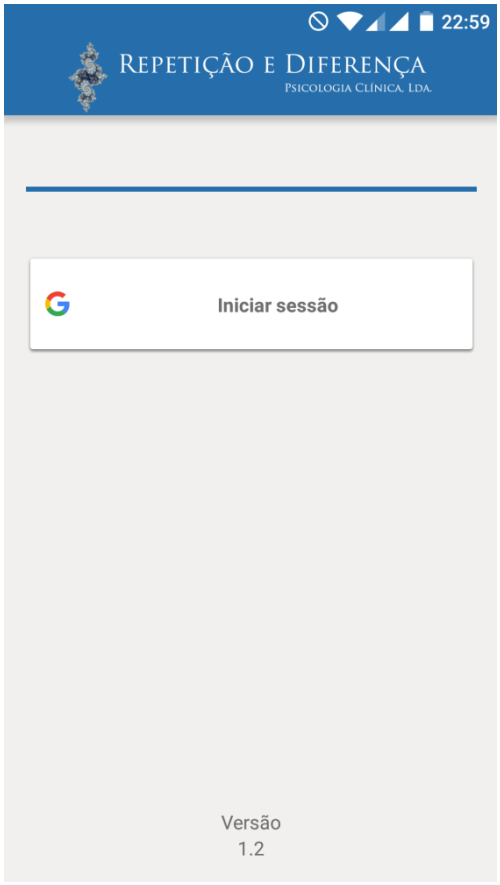


Figura 16 - Activity de Login

4.2. Activity das Notificações

No menu superior, além de existirem os botões que permitem ao utilizador navegar na aplicação, também aparece uma mensagem de boas vindas. O nome que aqui aparece é aquele que está associado à conta Google utilizado na autenticação.

A Figura 17 apresenta um *screenshot* da activity das notificações.

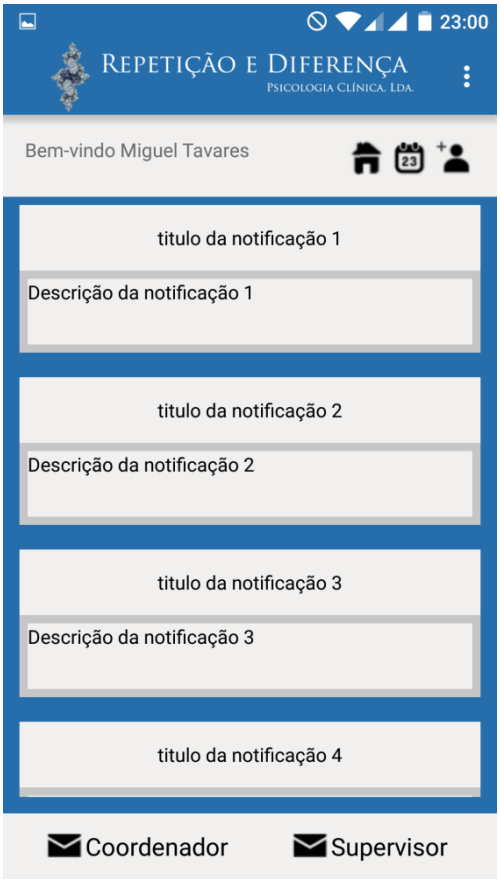


Figura 17 - Activity das Notificações

4.3. Activity do Calendário

Esta activity é constituída por um calendário onde o utilizador pode escolher uma data. Ao escolher a data, o utilizador terá acesso ao registo de presenças desse mesmo dia.

Na Figura 18 está representada a activity do calendário.

Figura 18 - Activity do Calendário

4.4. Activity do Registo de Presenças

Aqui o utilizador terá a possibilidade de consultar, alterar e registar as presenças dos seus clientes.

Nesta activity aparece um botão adicional no menu de navegação. O botão da disquete permite ao utilizador gravar as alterações efetuadas. Estes dados vão ser escritos na folha do registo de presenças existente no Google Drive.

Abaixo, na Figura 19 temos uma ilustração da activity das presenças.

Figura 19 - Activity das Presenças

4.5. Activity dos Processos

Aqui o utilizador poderá consultar, alterar e inserir novos processos. À semelhança da activity das presenças, aqui também aparece uma disquete que permitirá gravar os dados no Google Drive.

Para que os dados fiquem mais organizados, estes foram divididos em várias seções. Ao fazer *slide* o utilizador terá acesso às diversas seções. Para tornar esta navegação mais fácil e intuitiva, todas estas têm um título associado.

Na Figura 20 temos um *screenshot* da activity dos processos.

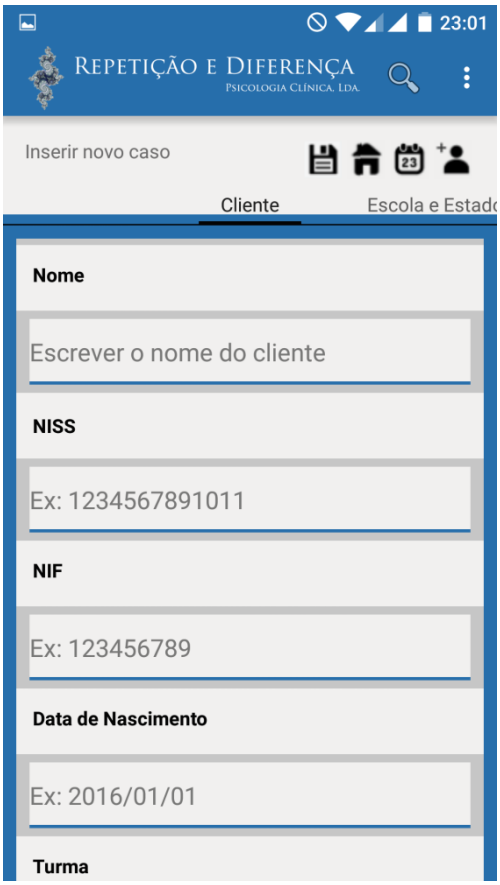


Figura 20 - Activity dos Processos

5. Conclusões

Fiquei bastante satisfeito com este TFC por diversos motivos. Um deles deve-se ao fato de ter trabalhado com diversas tecnologias de que gosto. Como sempre usei *smartphones* com sistema operativo Android, tinha interesse em saber mais coisas relacionadas com este sistema operativo. Senti que este trabalho foi muito vantajoso neste sentido.

Aprendi também que devo ser mais exigente a construir algoritmos no desenvolvimento de aplicações móveis, visto que estes equipamentos têm tipicamente mais limitações tanto ao nível da performance como ao consumo de tráfego de dados. Senti estas dificuldades por exemplo quando li uma folha de cálculo inteira.

Senti algumas dificuldades em tornar a aplicação *responsive*. Embora o Android Studio dê alguma ajuda (faz um *preview* com equipamentos de várias medidas), verifiquei que não é fácil adaptar uma aplicação a ecrãs de diversos tamanhos.

Outra situação que me agradou bastante neste projeto foi o facto de o problema se tratar de um caso real. Penso que este TFC me vai permitir ir mais (e melhor) preparado para o mercado de trabalho.

6. Trabalho futuro

Futuramente penso que seria interessante acrescentar uma funcionalidade de estatísticas à aplicação. Aqui os utilizadores teriam acesso à soma mensal das presenças com as faltas dos clientes. Penso que com esta funcionalidade adicional, o método de faturação da empresa iria ficar mais facilitado.

Julgo também que poderia ser interessante para fins estatísticos a possibilidade dos utilizadores terem acesso ao número de presenças e faltas de cada mês.

Na Figura 21 temos uma possível ilustração da activity das estatísticas.

Figura 21 - Possível activity das estatísticas (Maquete)

7. Bibliografia

Queirós, R. (2013). Android – Introdução ao Desenvolvimento de Aplicações. FCA

Referências eletrónicas

Market Share Statistics for Internet Technologies. (05/01/2016). Market Share Statistics for Internet Technologies.
https://netmarketshare.com/

Google. (06/01/2016). Sheets API.
https://developers.google.com/google-apps/spreadsheets/

Google. (06/01/2016). AsyncTask.
http://developer.android.com/reference/android/os/AsyncTask.html

Google. (06/01/2016). Supporting Multiple Screens
http://developer.android.com/guide/practices/screens_support.html

GitHub Education. (06/01/2016). Student Developer Pack - The best developer tools, free for students.
https://education.github.com/pack

TIOBE Software. (16/01/2016). TIOBE Index for January 2016.
http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Stackoverflow. (20/11/2015). using AsyncTask this way: “new AsyncTask(){” is giving me errors.
http://stackoverflow.com/questions/18706764/using-asynctask-this-way-new-asynctask-is-giving-me-errors

Developer Phil. (20/01/2016). Parcelable vs Serializable.
http://www.developerphil.com/parcelable-vs-serializable/

Anexos

Anexo A

Ficheiros Google Docs

Introdução

Uma vez que as figuras que compõem este anexo são as folhas de cálculo que a empresa usa, os dados contidos nas mesmas foram alterados de forma a salvaguardar a sua privacidade. As Figuras 1, 2 e 3 são screenshots das folhas existentes no Google Drive.

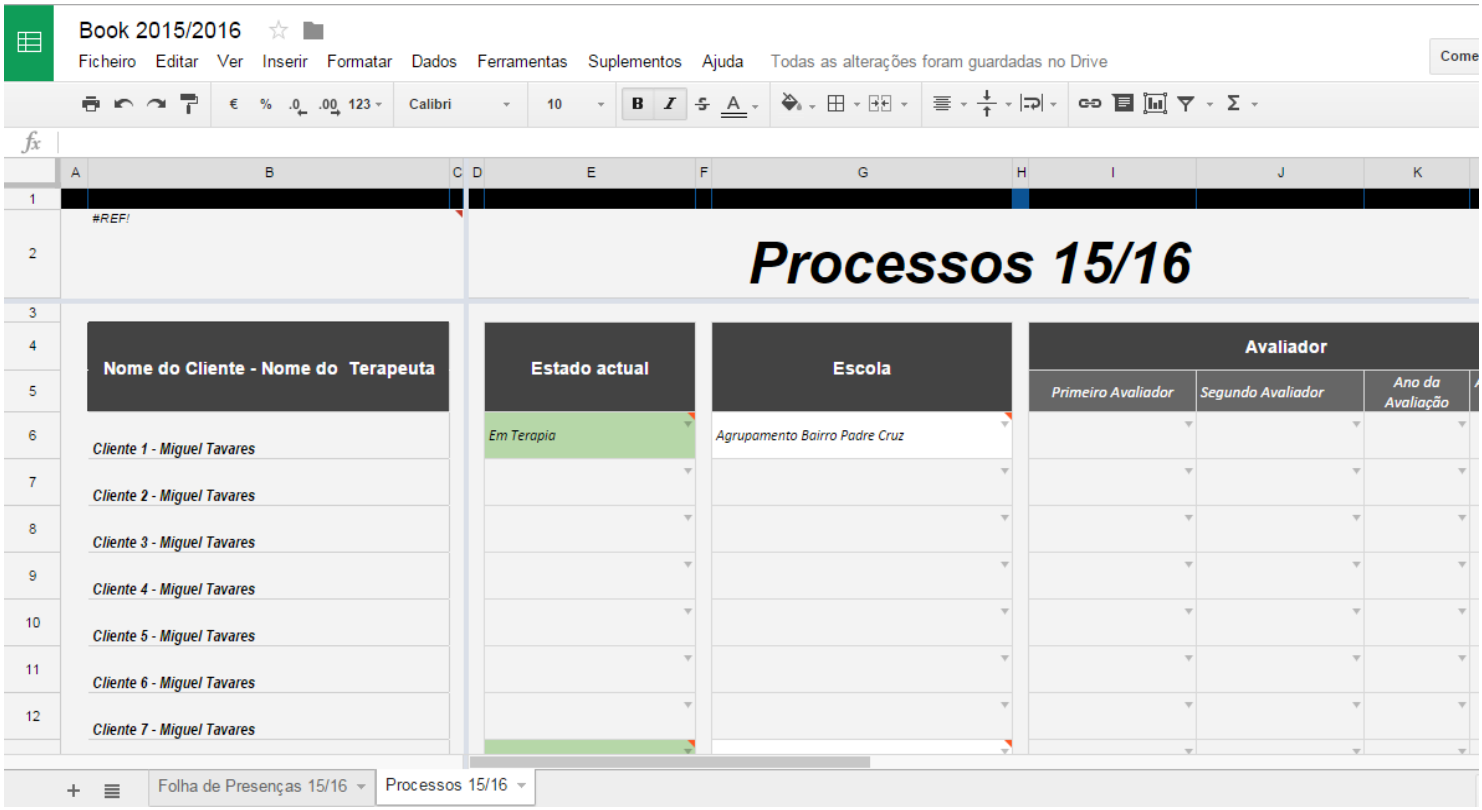


Figura 1 - Folha de cálculo dos Processos I

Book 2015/2016

Ficheiro Editar Ver Inserir Formatar Dados Ferramentas Suplementos Ajuda Todas as alterações foram guardadas no Drive

Comentários

#REF!

Nome do Cliente - Nome do Terapeuta	do Requerente			Dados do Cliente					Notas
	Cartão do Cidadão	NIF	Morada	Niss	NIF	Data de Nascimento	Turma	Ano	
Cliente 1 - Miguel Tavares					111111111	2016/05/05			
Cliente 2 - Miguel Tavares					222222222	888			
Cliente 3 - Miguel Tavares					333333333	777			
Cliente 4 - Miguel Tavares					444444444	666			
Cliente 5 - Miguel Tavares					555555555	111			
Cliente 6 - Miguel Tavares					666666666	13231			
Cliente 7 - Miguel Tavares					777777777	333			

Folha de Presenças 15/16 | Processos 15/16

Figura 2 - Folha de cálculo dos Processos II

Notificações Escolas

Ficheiro Editar Ver Inserir Formatar Dados Ferramentas Suplementos Ajuda Todas as alterações foram guardadas no Drive

✉ Notificações ✉

Folhas de Presença:	Folha de Presenças - Segurança Social
titulo da notificação 1	Descrição da notificação 1
titulo da notificação 2	Descrição da notificação 2
titulo da notificação 3	Descrição da notificação 3
titulo da notificação 4	Descrição da notificação 4
titulo da notificação 5	Descrição da notificação 5
titulo da notificação 6	Descrição da notificação 6
titulo da notificação 7	Descrição da notificação 7
titulo da notificação 8	Descrição da notificação 8
titulo da notificação 9	Descrição da notificação 9
titulo da notificação 10	Descrição da notificação 10

Figura 3 - Folha das Notificações

Anexo B

Requisitos

Requisitos Funcionais

Neste documento são apresentados os requisitos funcionais e técnicos do projeto, assim como as maquetes inicialmente propostas pela empresa.

Em algumas das activities acabaram por ser feitas pequenas alterações em relação à maquete respetiva, pois chegou-se à conclusão que isso seria vantajoso.

Autenticação

- O utilizador ao fazer *login* pode optar por uma de duas situações. Uma delas é autenticar-se através de uma conta Google já existente. A outra seria adicionar uma nova conta a seu gosto e fazer login através da mesma.
- Caso o utilizador não tenha acesso às folhas necessárias para o funcionamento da aplicação, este não se conseguirá autenticar.

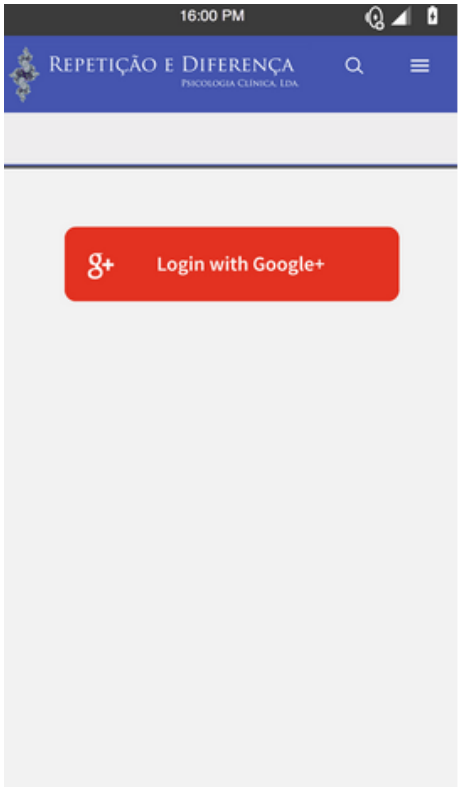


Figura 4 - Activity de Login (Maquete)

Notificações

- Assim que o utilizador se autenticar na aplicação terá acesso às notificações da empresa.
- O utilizador terá também a possibilidade de enviar *e-mails* para a coordenação e a supervisão da empresa.

Figura 5 - Activity das Notificações (Maquete)

Registo das presenças

- É necessário que os utilizadores tenham acesso a um menu onde possam registar as presenças dos seus clientes.
- O terapeuta só terá visibilidade sobre os processos que lhe estão associados.

Figura 6 - Activity do Calendário Maquete

Figura 7 - Activity das Presenças Maquete

Gestão de processos

- Os utilizadores terão acesso a um formulário onde poderão consultar, alterar e inserir processos.
- Todos os processos inseridos por um dado utilizador ficarão associados a esse mesmo utilizador.
- Os utilizadores só têm visibilidade sobre os processos que lhes estão associados.
- A aplicação deverá validar o preenchimento dos seguintes campos como sendo obrigatórios:
    - Nome do Cliente
    - NIF
    - Estado
    - Escola

- A aplicação deverá validar o preenchimento dos seguintes campos como sendo numéricos:
    - NIF
    - NISS
    - Data de Nascimento

- A aplicação deverá validar o preenchimento dos seguintes campos como tendo determinado tamanho
    - NIF (9 dígitos)
    - NISS (11 dígitos)

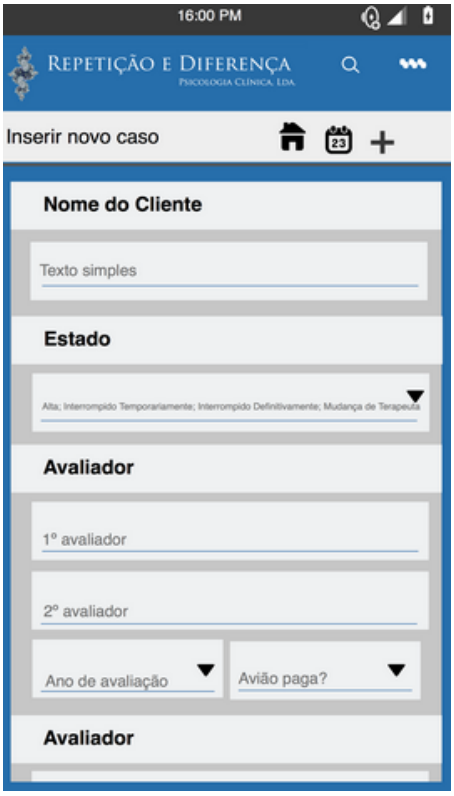


Figura 8 - Activity dos Processos (Maquete)

Requisitos Técnicos

- A aplicação deverá ser desenvolvida para o Sistema Operativo Android.
- A aplicação tem de interagir com as folhas de cálculo do Google Drive já existentes.

- A aplicação terá de implementar regras de concorrência. Durante o processo de gravação dos dados, o utilizador deverá ser alertado caso estes se alterem após a última leitura.

Anexo C

Como usar a Sheets API

Introdução

Este documento vai explicar o processo que temos de cumprir para termos acesso às folhas de cálculo no Google Drive.
Este manual vai ser organizado em três partes distintas:

1. Criar um projeto no Google Developers Console
2. Programação para termos acesso às folhas
   2.1. Leitura das folhas
   2.2. Escrita nas folhas

Criar um projeto no Google Developers Console

Para ter acesso às folhas de cálculo do Google Drive temos em primeiro lugar de criar uma conta no *Google Developers Console*[21]. Depois de criarmos o registo vamos criar um projeto. É através daqui que fazemos toda a gestão do nosso projeto, isto é, podemos ativar e desativar APIs, consultar os níveis de utilização das mesmas, entre outros aspetos.

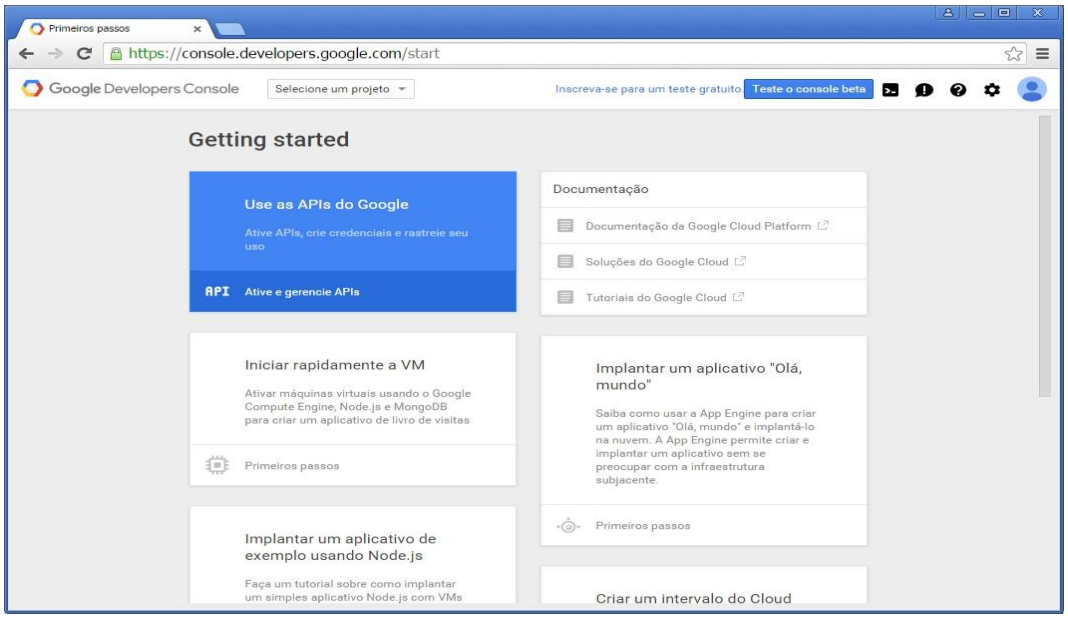


Figura 9 - Google Developers início

[21] https://console.developers.google.com

Depois de fazer o registo temos de criar um projeto.

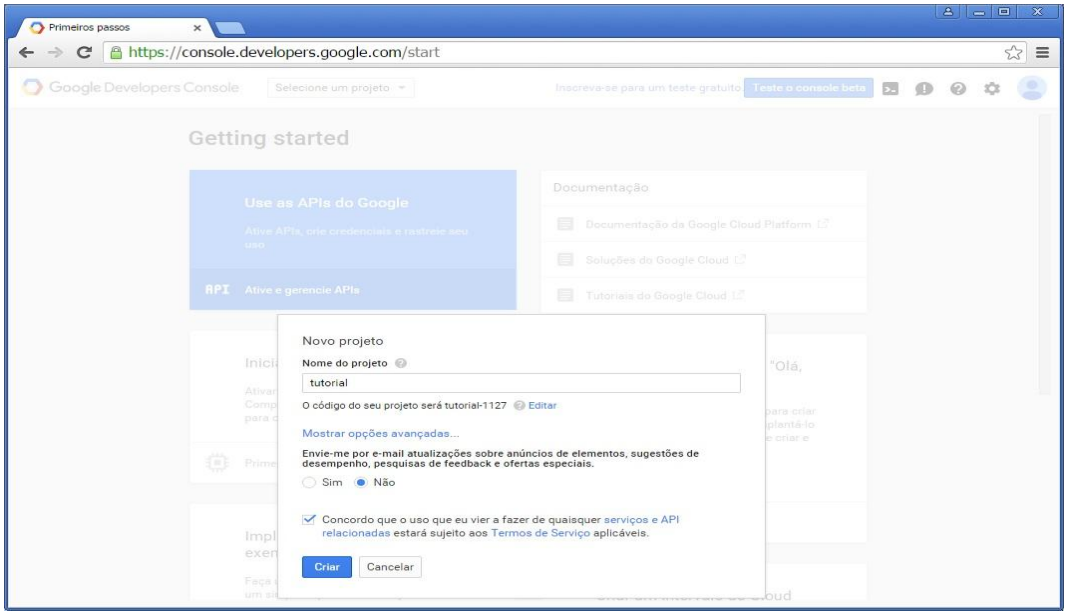


Figura 10 - Criar um projeto

Agora, selecionamos o projeto em que estamos a trabalhar. Podemos selecioná-lo no canto superior esquerdo.

Depois da seleção, vamos ativar as APIs que necessitamos no nosso projeto. Para isso, selecionamos à esquerda ***APIs e autenticação*** -> ***APIs***

Dentro deste menu vamos ter diversas API's. Aquela que queremos ativar é a ***Drive API***

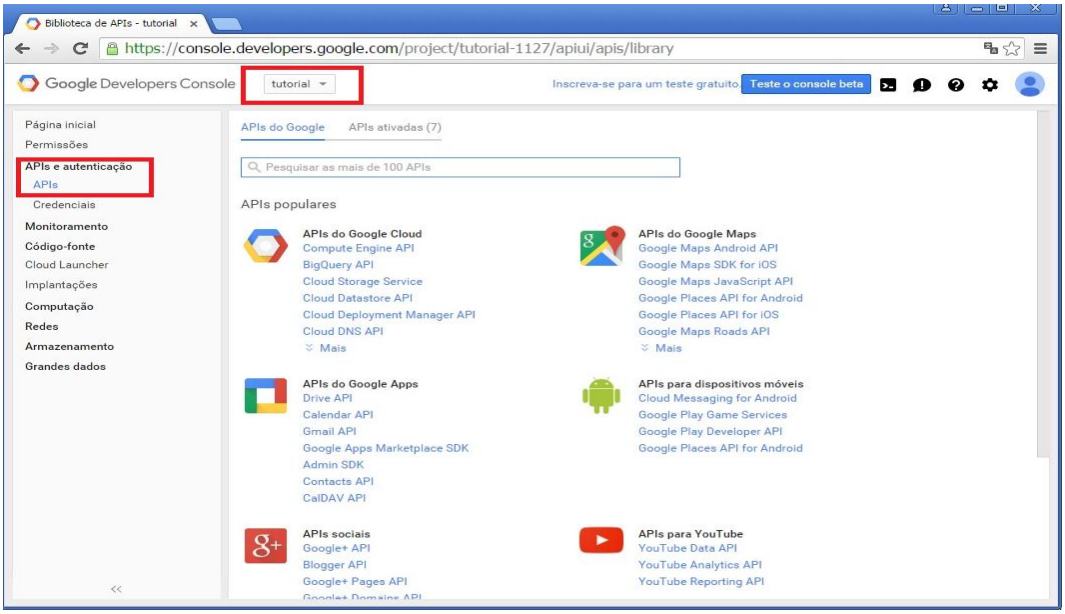


Figura 11 – Ativar a API

Depois de ativarmos a Drive API, podemos consultar a mesma na lista das APIs ativas.

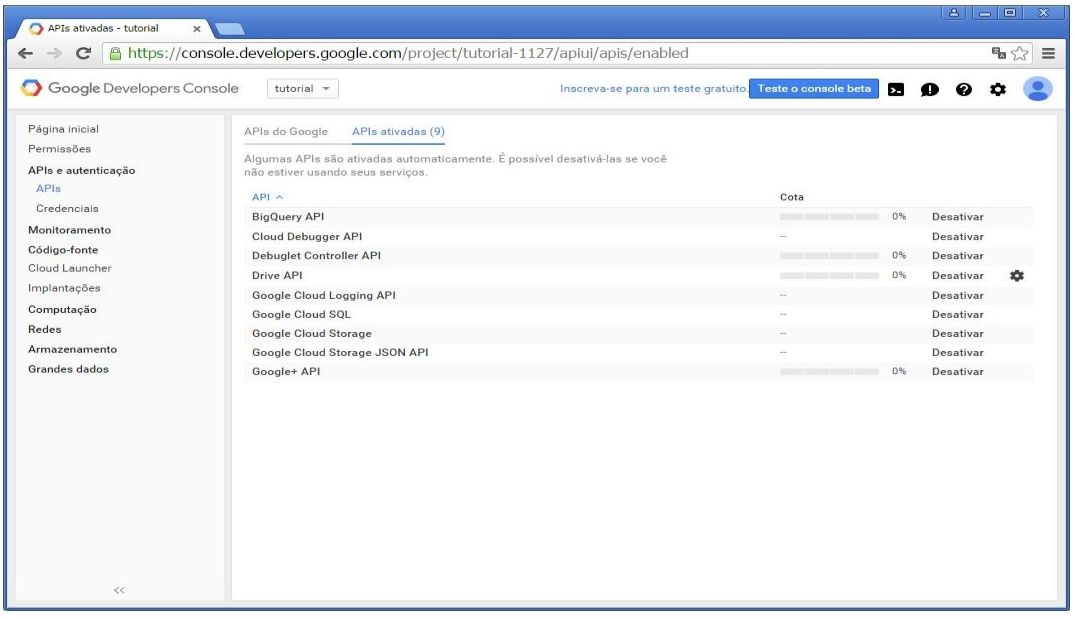


Figura 12 - Lista de APIs ativas

Assim que confirmarmos que a API pretendida está ativa, selecionamos à esquerda ***APIs e autenticação*** -> ***Credenciais***.

Dentro do menu das ***Credenciais*** selecionamos ***Adicionar credenciais*** -> ***ID do cliente OAuth 2.0***.

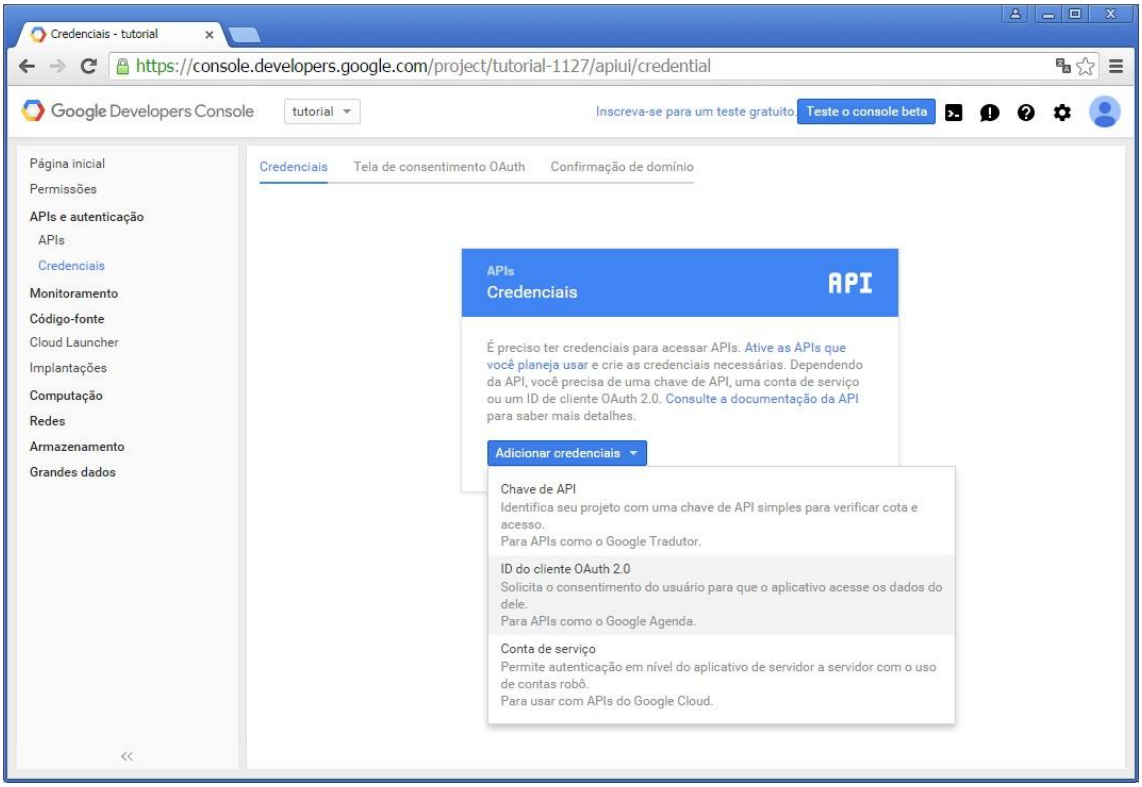


Figura 13 - Adicionar credenciais

No menu ***consentimento*** inserimos os dados relativos ao projeto que estamos a criar. Podemos adicionar o nome, um logotipo, entre outros detalhes.

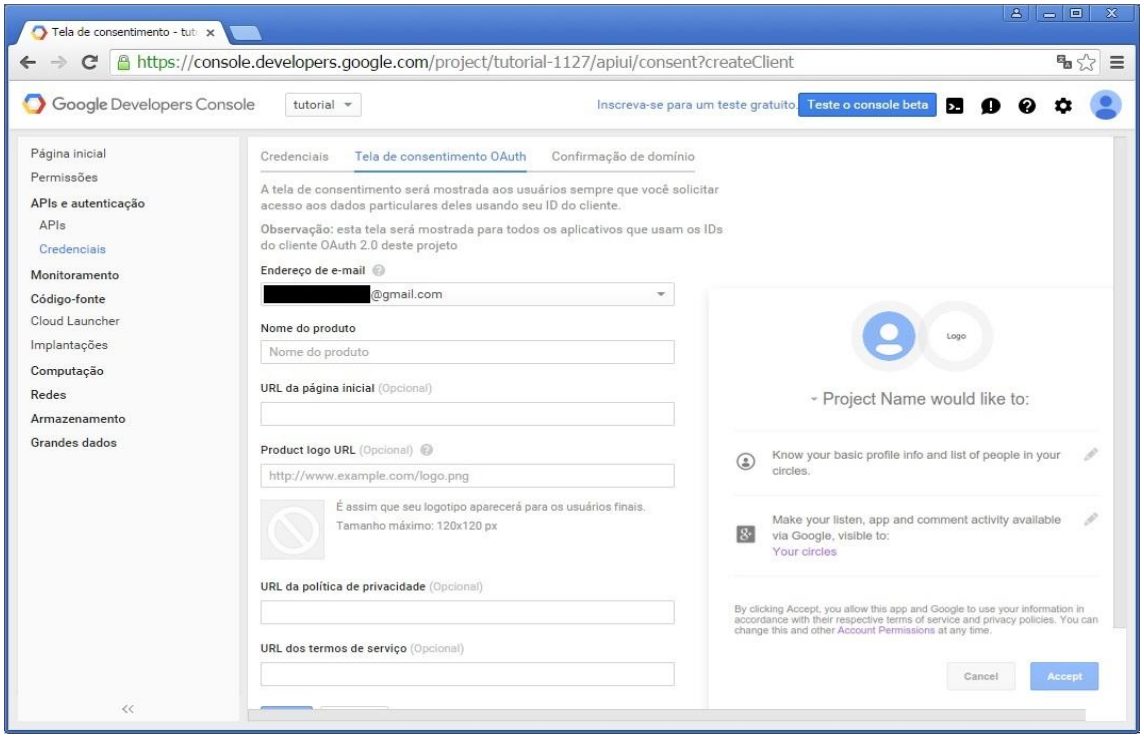


Figura 14 - Formulário das credenciais

Agora, na opção das ***credenciais*** vamos adicionar uma tecnologia onde a API vai ser utilizada. Um projeto pode ter várias credenciais, tais como Android, Aplicativo da Web, entre outros. Neste caso em concreto vamos selecionar Android.

No ***nome*** damos uma descrição relativa à credencial em questão, tal como foi dito, podemos ter diversas credenciais ao nível da API. Para o caso do Android temos de ***gerar um certificado de autorização.*** Para isso é necessário executarmos um comando que está presente na página atual.

keytool -exportcert -alias androiddebugkey -keystore path-to-debug-or-production-keystore -list –v

Exemplo para Linux

keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -keypass android

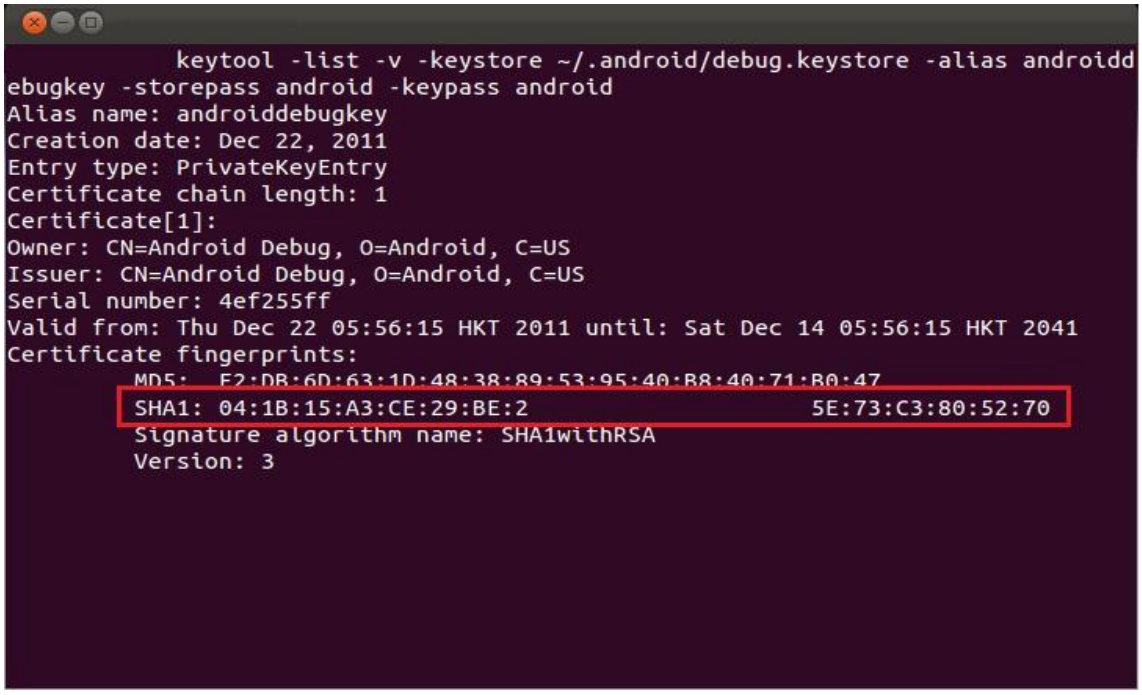


Figura 15 - Gerar chave SHA1

Ao executarmos este comando, vamos precisar da chave ***SHA1***. Vamos copiá-la e preenchemos o espaço alusivo à chave.

No ***Nome do Pacote*** temos de preencher com o nome do pacote Java que criámos. Neste caso em concreto poderia ser **com.tutorial**.

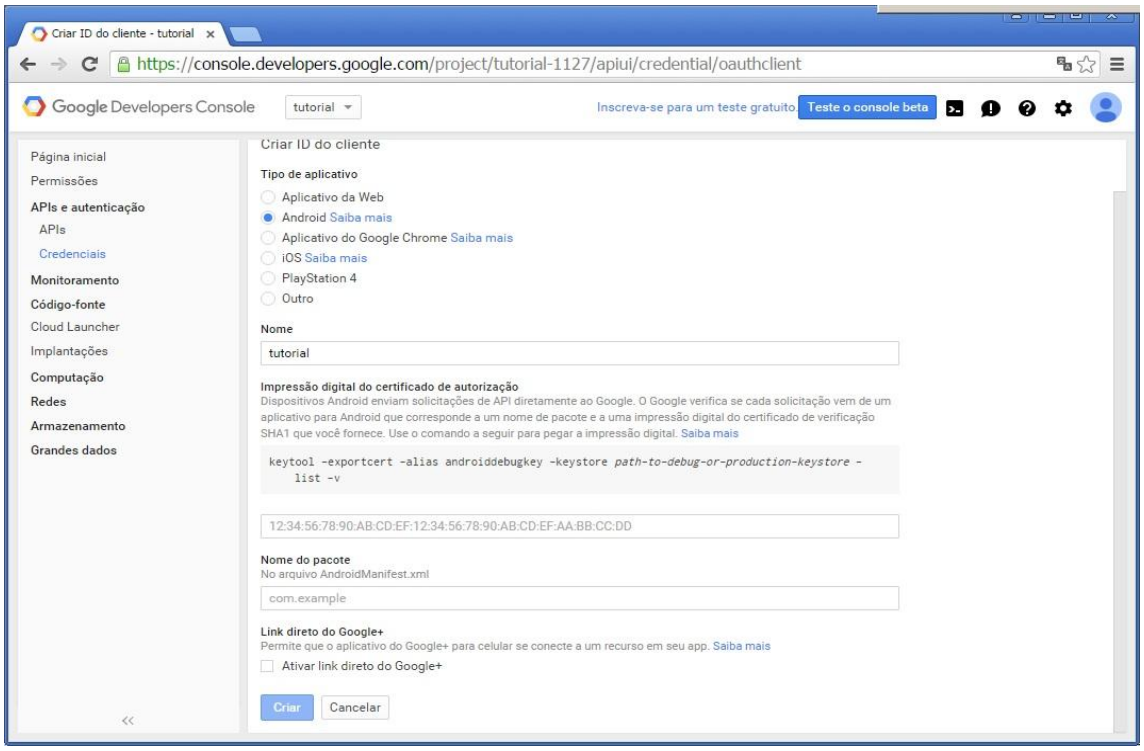


Figura 16 - Criar credencial

Depois de preenchermos todos estes campos pressionamos o botão ***Criar.*** Somos encaminhados novamente para o menu das credenciais. Aqui, podemos consultar a nossa lista de credenciais já criada.

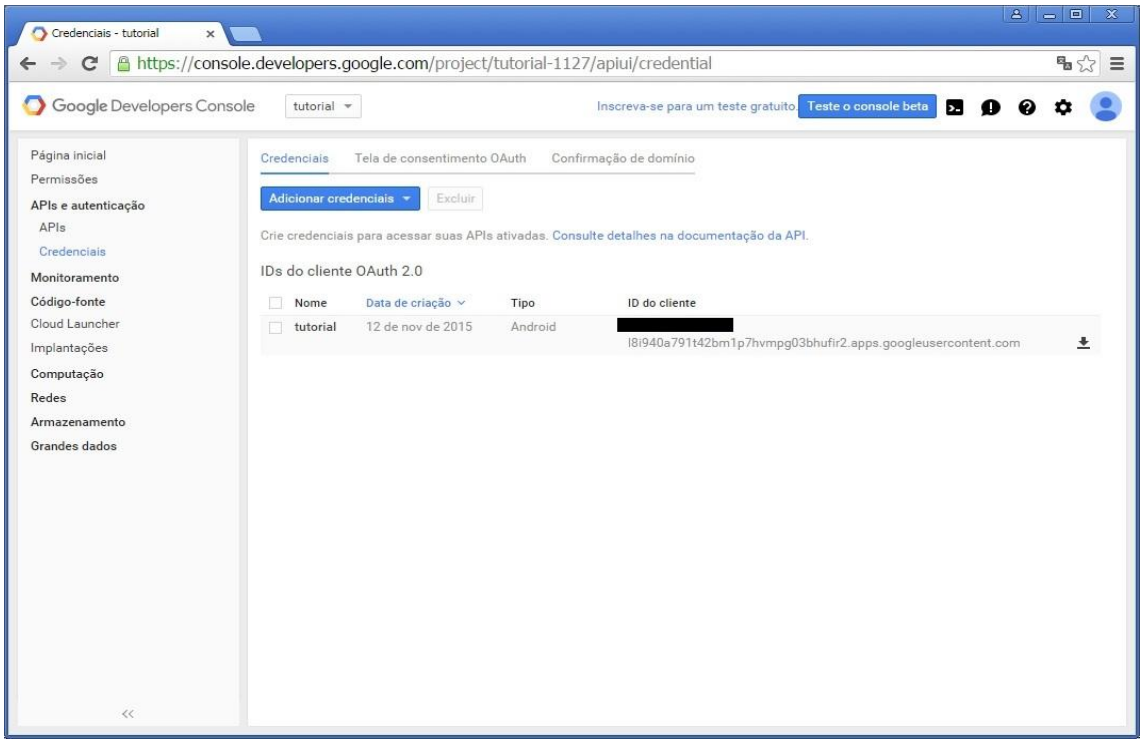


Figura 17 - Lista de credenciais

Programar o acesso às folhas do Google Drive

Para termos acesso às folhas temos em primeiro lugar de fazer *download* de algumas bibliotecas. Estas são disponibilizadas pela Google em https://github.com/google/gdata-java-client. Destas bibliotecas vamos apenas precisar de quatro:

- `gdata-core-1.0.jar`
- `gdata-spreadsheet-3.0.jar`
- `gdata-spreadsheet-meta-3.0.jar`
- `guava-11.0.2.jar`

Depois de criarmos o projeto da nossa App, temos de inserir estas bibliotecas na diretoria *~/app/libs*.

O Android Studio tem uma ferramenta que nos facilita o processo de *building* assim como a gestão das dependências do nosso projeto. Esta ferramenta chama-se Gradle e cumpre as mesmas funcionalidades que o Ant e o Maven juntos. Uma das vantagens do Gradle é ter uma linguagem de suporte bastante simples que se chama Groovy. O Groovy é uma linguagem OO com uma sintaxe relativamente parecida ao Java.

Agora que já conhecemos um pouco melhor o Gradle, vamos adicionar as dependências necessárias para o funcionamento da nossa aplicação.

build.gradle (project)

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:1.3.0'
        classpath 'com.google.gms:google-services:1.5.0'
        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}
```

build.gradle (app)

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.1.0'
    compile 'com.google.android.gms:play-services-auth:8.3.0'
    compile 'com.google.android.gms:play-services-plus:8.3.0'
}
```

Agora temos de dar algumas permissões no Manifest.xml. Estas permissões são necessárias porque sem elas o Android não vai permitir que façamos a ligação à Google.

```
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.NETWORK" />
<uses-permission android:name="android.permission.USE_CREDENTIALS" />
<uses-permission android:name="android.permission.INTERNET"/>
```

Este código XML tem de ser colocado entre as tags `manifest` e `aplication`.

Agora que temos tudo configurado a nível de permissões na nossa aplicação, podemos começar a programar o acesso às folhas de cálculo.

As folhas de cálculo têm dois tipos de visibilidade, privadas e públicas. As privadas são aquelas que apenas são vistas pelas contas que as criam e pelas contas com as quais estão partilhadas. As públicas são aquelas que estão visíveis a qualquer pessoa, quer tenha ou não uma conta Google.

Para explicar melhor as interações com as folhas criei uma folha pública de testes que está acessível através do seguinte endereço:

```
https://docs.google.com/spreadsheets/d/1QxLxCvhYGuH3vZ-CqRpJzL5ihhyLri8kGvSjwD3VU0c/
```

O que está a negrito e a sublinhado no endereço é uma referência unívoca que identifica o ficheiro do Google Drive. Este identificador é fulcral para o acesso às folhas públicas, é através dele que conseguimos diferenciar esta folha de cálculo das outras folhas de cálculo públicas.

Os exemplos de código que se seguem estão disponíveis num repositório público do GitHub[22] criado por mim.

```
// Usar este para folhas privadas
SPREADSHEET_FEED_URL = new
URL("https://spreadsheets.google.com/feeds/spreadsheets/private/full");
```

[22] https://github.com/mstavares/SheetsAPI_TFC

```
// Usar este para folhas publicas
URL SPREADSHEET_FEED_URL = new
URL("https://spreadsheets.google.com/feeds/worksheets/"+ KEY_FOLHA
+"/private/full");
```

A primeira linha de código[23] vai servir para dizermos que queremos fazer a pesquisa das folhas existentes no nosso Google Drive. As folhas que vamos encontrar serão as que nós criámos e aquelas que foram partilhadas com a nossa conta Google. Com a segunda linha[24] a folha que vamos encontrar é a que está associada ao identificador que é igual a `KEY_FOLHA`.

O método que vamos escrever para a leitura das folhas tem de ser assíncrono. Isto porque o Android utiliza a *thread* principal para a gestão da IU (Interface do Utilizador). Se executarmos as próximas instruções sem que as mesmas estejam num método assíncrono, vamos ter uma *exception.* Isto vai acontecer porque estamos a comprometer a performance do sistema operativo ao sobrecarregar a *thread* principal, e isto o Android não permite que aconteça.

Este método assíncrono é criado através da classe abstracta `AsyncTask`[25]. No sistema operativo Android não é recomendado que façamos a criação de *threads* diretamente. A classe `AsyncTask` faz a gestão das *threads* sem que tenhamos de as definir.

Para a leitura das folhas é necessário fazer `Override` de dois métodos, `doInBackground` e `onPostExecute`. O segundo método é executado assim que o `doInBackground` acabar. Por isso, no `doInBackground` vamos programar o acesso às folhas e no `onPostExecute` programamos o que queremos fazer depois de encontrarmos ou não as folhas.

```
String scope = "oauth2:https://spreadsheets.google.com/feeds";
token = GoogleAuthUtil.getToken(MainActivity.this, conta.getEmail(), scope);
service = new SpreadsheetService("MySpreadsheetIntegration-v3");
service.setHeader("Authorization", "Bearer " + token);
```

[23] Ficheiro MainActivity.java linha 195
[24] Ficheiro MainActivity.java linha 198
[25] http://developer.android.com/reference/android/os/AsyncTask.html

A `String scope` vai definir a nível do Google Drive onde vamos fazer a pesquisa das folhas[26]. O Google Drive tem vários scopes e aquele que vamos utilizar é o que está afeto às folhas de cálculo.

O `token` é uma `String` que vamos receber para de futuro realizarmos a nossa autenticação. Se o `token` for devidamente criado o retorno deste método será diferente de `null`. O método `GoogleAuthUtil.getToken` recebe como parâmetros o contexto da activity onde estamos a trabalhar, o *e-mail* associado à conta Google e o nosso scope.

Se a autenticação decorrer sem dificuldades vamos receber uma `String` que será diferente de `null`.

A linha de código seguinte vai definir a versão da API que vamos utilizar. De acordo com a Google não faz sentido utilizar as versões anteriores porque já não há folhas de cálculo a funcionar sobre as versões anteriores.

O método `service.setHeader` vai agora fazer a autenticação no Google Drive através do `token` gerado anteriormente.

Se todos estes passos forem executados com sucesso, a partir deste momento já vamos ter acesso às folhas. Tal como já foi indicado anteriormente, o acesso às folhas é efetuado de maneiras distintas consoante a visibilidade da folha em questão.

Para termos acesso às folhas públicas fazemos da seguinte forma:

```
WorksheetFeed feed = service.getFeed(SPREADSHEET_FEED_URL,
WorksheetFeed.class);
	List<WorksheetEntry> worksheets = feed.getEntries();
	for (WorksheetEntry worksheet : worksheets) {
		if(worksheet.getTitle().getPlainText().equals(nomeDaFolha)) {
			folha = worksheet;
			autenticado = true;
			break;
		}
	}
```

[26] Ficheiro MainActivity.java linha 200

Através do objeto `feed` vamos receber uma referência da qual vamos conseguir extrair uma lista que irá conter todas as folhas. Neste caso em concreto, as folhas públicas que vamos receber são as que estão associadas à chave que definimos na `String SPREADSHEET_FEED_URL`[27].

Agora que temos uma lista com todas as folhas, podemos efetuar uma pesquisa nessa lista através do título da folha. O título da folha é o nome que aparece na aba inferior da folha de cálculo.

Para o caso das folhas privadas o procedimento é relativamente semelhante[28].

```
SpreadsheetFeed feed = service.getFeed(SPREADSHEET_FEED_URL,
SpreadsheetFeed.class);
	List<SpreadsheetEntry> spreadsheets = feed.getEntries();
	for (SpreadsheetEntry spreadsheet : spreadsheets) {
		if (spreadsheet.getKey().equals("1vgSF2baarOk17Ox-
pMmjDmsWtHFqdEUxYCzfY9KMcPg")) {
			folhas = spreadsheet;
			autenticado = true;
			break;
		}
	}
```

Enquanto que no caso das folhas públicas fazíamos a pesquisa ao nível da folha, para as folhas privadas temos, em primeiro lugar, de fazer uma pesquisa ao nível do *book*. Um *book* pode conter ou não diversas folhas. O identificador unívoco indicado no endereço da folha pública é o identificador daquele *book*, uma vez que já estamos a referenciar aquele ficheiro, não é necessário efetuar uma pesquisa pelo mesmo.

Em ambos os casos assim que encontramos o *book* ou a folha que pretendemos guardamos a mesma. Quer a nossa pesquisa tenha ou não sucesso, guardamos o resultado desta num booleano.

Este código obrigatoriamente tem de estar envolvido num `try / catch`. Uma das exceções que este código retorna é devido às permissões do Google Drive[29].

[27] Ficheiro MainActivity.java linha 210
[28] Ficheiro MainActivity.java linha 229
[29] Ficheiro MainActivity.java linha 224

O parâmetro `AUTH_CODE_REQUEST_CODE` é uma constante do tipo `int` que temos de definir. Esta constante pode assumir um valor arbitrário.

```
} catch (UserRecoverableAuthException transientEx) {
      startActivityForResult(transientEx.getIntent(),AUTH_CODE_REQUEST_CODE);
}
```

Se o utilizador ainda não deu permissões de acesso ao Google Drive vamos lançar um ecrã para que ele o possa fazer[30].

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
      if (requestCode == AUTH_CODE_REQUEST_CODE) {
            GoogleSignInResult result =
Auth.GoogleSignInApi.getSignInResultFromIntent(data);
            if (result != null && result.isSuccess()) {
                  conta = result.getSignInAccount();
                  nomeDaConta = conta.getDisplayName();
                  getGoogleOAuthTokenAndLogin();
            } else {
                  signInWithGplus();
            }
      }
}
```

Esta autorização fica associada à conta Google do utilizador, assim de futuro não será necessário voltar a fazê-lo.

Agora que já temos as folhas que pretendemos ler, será agora possível executarmos operações de escrita e leitura sobre as mesmas.

[30] Ficheiro MainActivity.java linha 135

Leitura das folhas

Para lermos e escrevermos nas folhas vamos necessitar novamente de métodos assíncronos. Estes métodos são necessários pelos mesmos motivos que foram indicados na pesquisa das folhas[31].

```
// folhas privadas
for (WorksheetEntry worksheet : MainActivity.getFolhas().getWorksheets())
	if (worksheet.getTitle().getPlainText().equals("Folha")) {
		folhaEscolhida = worksheet;
		break;
	}
```

```
// folhas públicas
folhaEscolhida = MainActivity.getFolha();
```

Como no caso da pesquisa, os processos de leitura e escrita em folhas públicas e privadas é também bastante semelhante. A única diferença entre estes processos é que no caso das folhas públicas já temos o acesso à folha em si. No caso das folhas privadas, como temos acesso ao *book*, temos de fazer uma nova pesquisa dentro desse *book* pela folha que pretendemos usar. Esta parte do código tem de estar dentro do método `doInBackground`.

Para fazermos a leitura das folhas temos que saber de antemão as células que pretendemos ler. Essa restrição pode ser efetuada ao nível da linha e ou ao nível da coluna[32].

```
private static String restringeLinhasEColunas(int linhaMin, int linhaMax, int
ColunaMin, int ColunaMax) {
	return "?min-row=" + linhaMin + "&max-row=" + linhaMax + "&min-col=" +
ColunaMin + "&max-col=" + ColunaMax;
}
```

[31] Ficheiro Main2Activity.java linha 85
[32] Ficheiro Main2Activity.java linha 55

O método referido acima é aquele que faz a restrição das células que vamos ler ou escrever. Como podemos verificar, este método recebe quatro argumentos, dois deles ao nível da linha e outros dois ao nível da coluna. Como vamos selecionar uma área de células dentro da folha temos de delimitar essa área com início e fim. Os argumentos `linha` / `coluna` definem o início da seleção, os `linhaMax` / `colunaMax` definem o fim da seleção.

Agora que temos o intervalo de células que pretendemos ler, já podemos receber os dados das células[33].

```
folhaEscolhida = MainActivity.getFolha();
feedURL = new URI(folhaEscolhida.getCellFeedUrl().toString() +
restringeLinhasEColunas(LINHA_DA_CELULA_A1, LINHA_DA_CELULA_A1,
      COLUNA_DA_CELULA_A1, COLUNA_DA_CELULA_A1)).toURL();
feedCell = MainActivity.service.getFeed(feedURL, CellFeed.class);
celulasLidas = feedCell.getEntries();
```

O método `getEntries()` vai devolver uma lista do tipo `CellEntry`. Cada elemento dessa lista vai conter os dados das células escolhidas, entre outras informações.

Agora que temos uma lista com o conteúdo de todas as células, só temos que fazer um ciclo `for` para imprimirmos esses dados[34].

```
private static  void lerDaCelula (List<CellEntry> celulasLidas) {
      for(CellEntry celula : celulasLidas)
            edtxtDados.setText(celula.getCell().getValue());
}
```

[33] Ficheiro Main2Activity.java linha 102
[34] Ficheiro Main2Activity.java linha 141

Escrita nas folhas

O processo de escrita é exatamente igual à leitura, a única diferença é nas instruções utilizadas para a escrita em si. Após definirmos a linha ou coluna onde vamos escrever, só temos de passar esses dados ao método que vai tratar da escrita dos dados[35].

```
private static void escreverNaCelula(URL feedURL, int linha, int coluna) {
	try {
		MainActivity.getService().insert(feedURL, new CellEntry(linha, coluna, aEscreverNaCelula));
	} catch (IOException e) {
		e.printStackTrace();
	} catch (ServiceException e) {
		e.printStackTrace();
	}
}
```

[35] Ficheiro Main2Activity.java linha 146