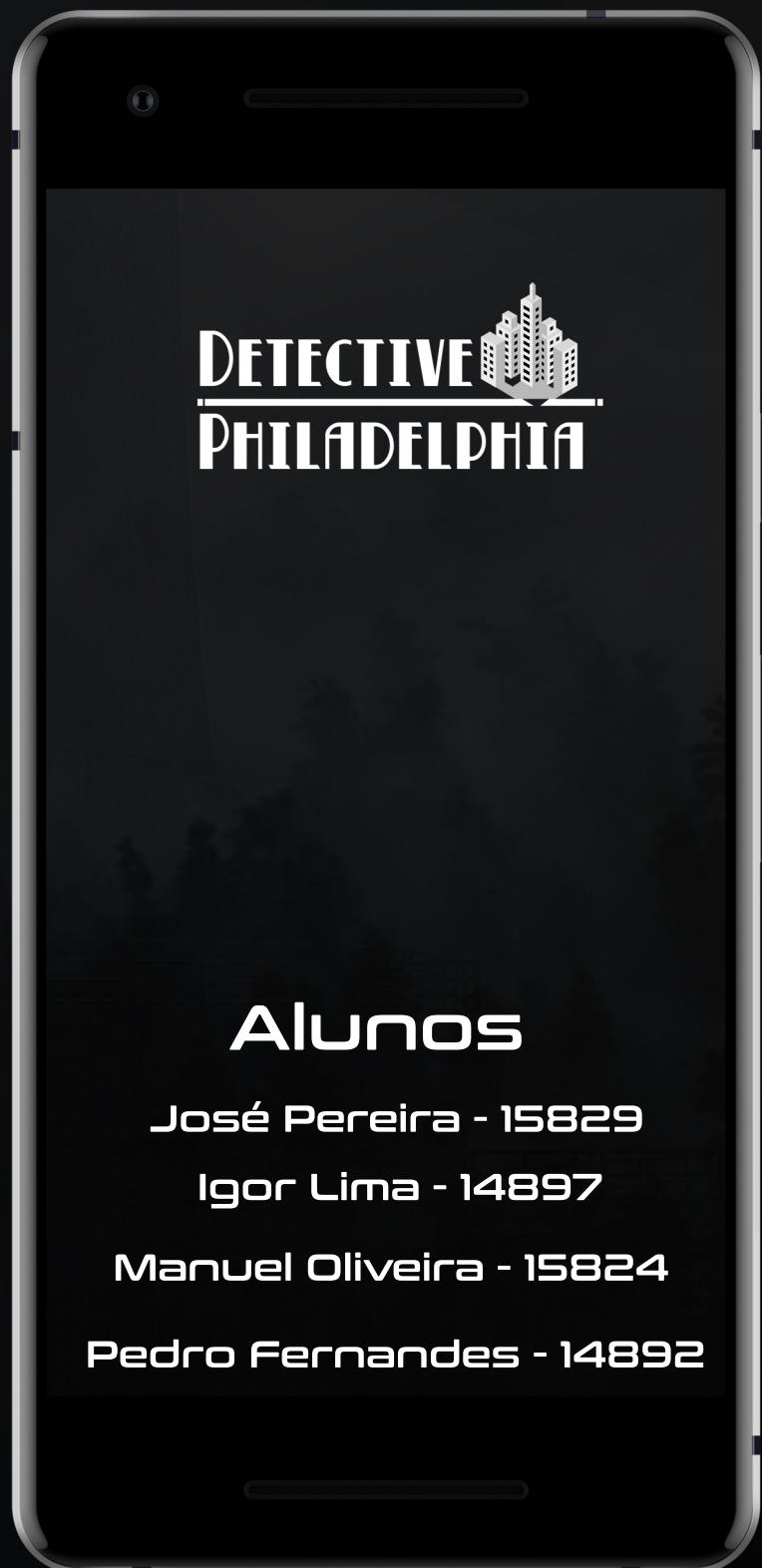


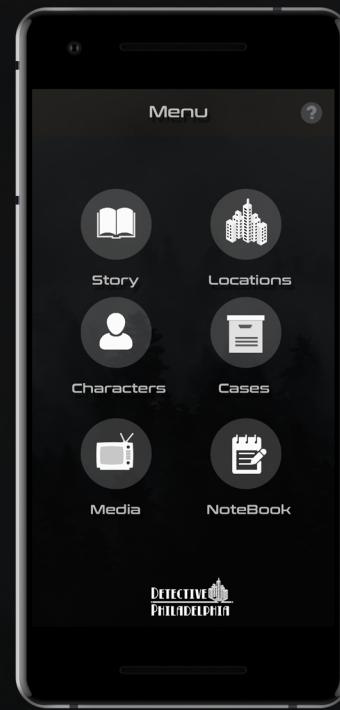
# Estrutura do Trabalho



# Estrutura do Trabalho



Ecrã principal

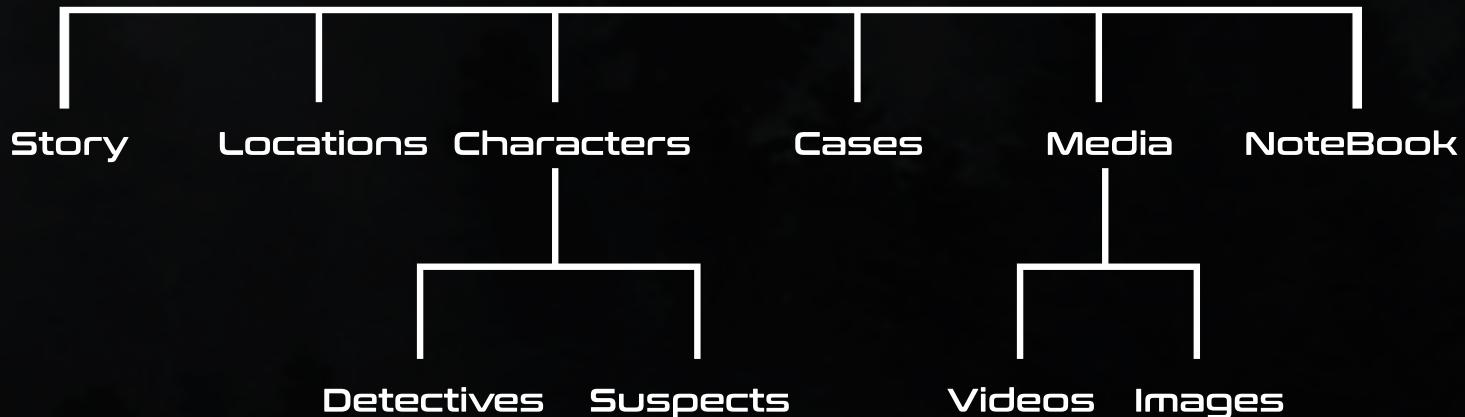


Ecrã de navegação

## Ramificação de atividades

Ecrã principal

Ecrã de navegação



# Estrutura do Trabalho

## O que é a APP?

Esta app irá fazer de acompanhamento ao nosso jogo “Detective philadelphia”. O utilizador desta App irá beneficiar dela, pois ela contém tanto a história, as personagens, as localizações, os casos e os ficheiros multimédia organizados por categorias e também de um notebook que se irá tornar uma fiel ferramenta para o jogador.

## Design

O design desta App será direcionado a partir da paleta de cores que se encontram no nosso jogo, essa paleta de cores é escura e de tons semi-frios.

O design será minimalista, mas com algum detalhe nos ícones para melhorar a navegação do utilizador na App.



## Ecrã Principal

O ecrã principal contém um campo de inserção de texto que permite ao utilizador colocar o seu nome na App, para que algum dos textos apresentados a seguir sejam personalizados.

Depois de inserir, tanto o nome como o ano de nascimento, para confirmar que o utilizador tem mais de 16 anos, então o botão de Start será ativado e permitirá ao utilizador passar do ecrã principal para o ecrã de navegação.

Se o utilizador inserir um ano que nos indica que tem menos de 16 anos, aparecerá um toast a indicar que não possui idade suficiente para utilizar a App.

# Estrutura do Trabalho

## Ecrã de navegação

O ecrã de navegação desta App, será o menu pela qual o utilizador poderá aceder tanto a informação categorizada, mas também a ferramentas que o irão ajudar.

Neste ecrã encontraremos 6 botões que nos irão direcionar a diferentes atividades.

Cada botão apresenta um ícone representativo do que cada atividade faz, como também um pequeno título na parte inferior de cada ícone.

O botão de informação na parte superior direita apresentará a informação dos developers do jogo e da App.

Este menu terá a possibilidade de recuar para o ecrã anterior para ser possível a mudança de nome ou idade



# Estrutura do Trabalho

## Story (sinopse)

O ecrã da categoria “Story” irá ser constituído sómente por um TextView que irá conter toda a informação geral sobre o jogo.

## Locations

Este ecrã já será constituído por uma ListView que em cada elemento contém o seu nome e uma imagem, após escolher qualquer dos elementos o utilizador será levado para outro ecrã que mostrará a imagem com maior definição e os detalhes sobre o determinado local e a sua importância no jogo.

## Characters

Constituído por 2 botões extra (Detectives e Suspects) sendo que cada um é constituído por uma ListView com todas as personagens. Os elementos da lista dos detectives irá ter os elementos com maior dimensão pois apenas serão 2 personagens ao contrário dos 5 suspeitos.

Cada elemento da lista também será constituído pelo nome da personagem tal como a sua imagem representativa.

Após pressionar em qualquer elemento o utilizador será levado para um ecrã que mostra toda a informação da personagem (estilo ficheiro detetive).

# Estrutura do Trabalho

## Cases

Este ecrã apresentará a informação dos casos de detective do nosso jogo.

Será caracterizado por uma lista que mostra os casos do jogo.

Depois de clicar num dos elementos o utilizador será redirecionado para uma outra atividade que irá conter dados provenientes da rede, tanto texto e imagens que caracterizam cada caso.

O http disponibilizado à App será proveniente de um servidor FTP que conterá como texto um ficheiro em formato de Json, no qual iremos retirar as informações e também o url das imagens de cada caso.

## Media

Este ecrã estará dividido com dois botões a indicar a existência de 2 tipos de media disponíveis na nossa App.

Um desses formatos é o de imagem.

Na atividade responsável pela apresentação das imagens haverá uma grid para apresentar as imagens marcantes do nosso jogo.

Outra atividade será responsável pela apresentação de vídeos que estarão disponibilizados na App.

A utilização da VideoViewer será essencial para a funcionalidade desta App.

Estes vídeos estarão disponíveis na memória do dispositivo ou por um URL (isto ainda se encontra por definir).

# Estrutura do Trabalho

## NoteBook

**Nesta atividade o utilizador poderá criar as suas próprias notas de consulta, no estilo do próprio detetive do jogo.**

**Na principal atividade desta categoria será possível ver uma lista com o título e data de cada nota que o utilizador escrevou e também um action button que irá iniciar uma atividade que permitirá ao utilizador adicionar uma nota, pedindo um título e conteúdo, sendo que a data provém do sistema.**

**Ao carregar num elemento da lista iremos apresentar uma atividade que tem todo o conteúdo da nota apresentada.**

**Haverá possibilidade de apagar cada nota utilizando um botão dentro da nota.**

# Estrutura de Código

## Dados de Internet

Para buscar dados à internet utilizamos 2 tipos de acessos e mais um auxiliar.

Primeiro criamos um alojamento (servidor FTP), e no seu index/( dado que é publico) armazenamos as imagens, vídeos e também um JSON.

Para a conexão ao servidor FTP, utilizamos uma classe chamada `HttpFetchData`, esta classe trata dos processos `Async`, pois modos de procura na internet requerem outra threads sem ser a principal, pois iriam bloquear elementos de UI da nossa aplicação e outras funções até que fossem libertadas.

Para fazermos load das imagens numa outra classe também utilizamos um JSON disponível no servidor.

Também utilizamos o firebase(Google) ,com as suas implementações em Auth(Login) e Database(Armazenamento de Notas/Ficheiros).

Também acedemos à internet para autenticar o user e/ou fazer reset da password.

## Video Player

Nesta classe temos um vídeo player em que reproduzimos um vídeo a partir de um url do nosso servidor FTP disponível em <https://detectivephiladelphia.000webhostapp.com/>

Fazemos set do `VideoPath` com o url do ficheiro mp4 no nosso servidor FTP

Temos um botão que nos inicia o vídeo e outro que pausa o vídeo, o botão de pausa funciona como um botão de resume, quando o vídeo já se encontra em pausa este funciona como um botão de resume.

Definimos um `OnItemClickListener()` para os botões, que é adquirido a partir da implementação de `View.OnItemClickListener()` na Classe.

Quando o vídeo está a reproduzir e carregamos no botão, então este irá pausar e guardar a sua posição, para que o vídeo possa continuar a partir dessa posição quando o botão para resumir o vídeo for tocado.

O botão de restart reinicia o vídeo.

# Estrutura de Código

## Utilities

Esta Classe Java contém todas as funções comuns e necessárias para a organização e conversão de alguns tipos de ficheiro, como também a definição de paths e bitmaps.

Algumas das funções guardam bitmaps , outras fazem load desses bitmaps, outras retornam o caminho de storage do dispositivo do utilizador.

## Story

Esta actividade contém unicamente a definição de um layout que contém uma ImageView contendo a história do nosso jogo.

## Cases

Esta actividade contém unicamente a definição de um layout que contém uma ImageView contendo os casos do nosso jogo.

## ResetPassword

Esta actividade serve para que se o utilizador se esquecer da palavra passe quando do seu login, a app manda um email de mudança de palavra passe de para o email do utilizador, desde que ele esteja na base de dados do firebase.

Nela temos um botão, e um campo de texto editável, em que o utilizador coloca o seu email, e se o email for nulo, existe um display de um Toast que nos indica isso mesmo.

Nela criamos uma instancia de firebaseAuth, em que depois acedemos à função sendPasswrodResetEmail, que recebe o email, e adicionamos um OnCompleteListener, que contém uma task que nos indica se a task respectiva foi completada com sucesso ou não através de um campo booleano que nos indica isso mesmo(task.isSuccessful()), aparecendo de seguida um Toast que indica o estado do processo.

# Estrutura de Código

## Personas

Esta actividade contém unicamente a definição de um layout que contém uma ImageView contendo +as personagens do jogo, e os seus dados principais.

## Menu

A actividade do menu contem os botões de direcionamento para todas as funções da nossa APP, sendo que podem ser elas acedidas através de ImageButtons , que apresentam um onClickListener, em que no método OnClick(), verificamos o seu id , e consoante o seu ID , o Intent para começar uma nova actividade é inicializado e essa actividade é aberta.

O botão de interrogação presente no canto superior direito contem conexão a um Toast que apresenta os developers da App.

## Media Menu

A actividade do menu contem os botões de direcionamento para todas as funções da Media ,imagens e trailer(vídeo) sendo que podem ser elas acedidas através de ImageButtons , que apresentam um onClickListener, em que no método OnClick(), verificamos o seu id , e consoante o seu ID , o Intent para começar uma nova actividade é inicializado e essa actividade é aberta.

# Estrutura de Código

## LoginActivity

A LoginActivity contém 2 campos de texto editável, como também dois botões:

um de SignUp e outro que redireciona o utilizador para a actividade ResetPassword, onde poderá iniciar o processo de mudança de palavra passe.

No método OnCreate(), se o user(fireBase Auth) não for igual a nulo, ou seja, já existir, então automaticamente passamos o utilizador à actividade seguinte (Menu) e mostramos um Toast a dar a indicação que o utilizador já se encontra Logged In.

Também temos uma função RegistreUser, que é acionada quando o utilizador clica no botão de signup.

Este método, retorna os valores de email e password que estão contidos nas respectivas PlainText e verifica se o email ou password se encontram vazios.

Para dar feedback ao utilizador, temos um progresso Dialog que fica aberto quando o utilizador se encontra a registrar.

Para registar o utilizador na plataforma de FireBase utilizamos a função firebaseAuth.createUserWithEmailAndPassword(email, password), que cria um utilizador na nossa base de Auth(-FireBase), e nessa função adicionamos um OnCompleteListener, onde temos um booleano na task que nos indica o sucesso ou não do processo, se positivo, passamos ao menu, senão, mostramos um Toast a indicar o insucesso da operação ao utilizador.

# Estrutura de Código

## Locations

Nesta actividade temos como objectivo apresentar uma ScrollView que apresenta as localizações disponíveis e que ao clicarmos nelas, outra actividade vai aparecer, onde as informações disponíveis na classe LocationModel, que contem:

```
public String getLocationName()
```

```
public String getDescription()
```

Estas informações são passadas a uma nova actividade através de um putExtra do Intent, onde na outra actividade transformamos essas keys em texto informativo.

Nesta classe existe uma lista de LocationModel, onde são adicionadas localizações

Dentro da ScrollView possuímos um OnClickListener() para verificarmos cliques.

## LocationModel

LocationModel é uma classe java que contém as informações necessárias para o display na actividade de Location-Details.

## Images

A actividade de images contem uma ListView que contém as imagens vindas do servidor FTP.

O adaptor da lista é formado através do XML. List\_item, que contém referencia a uma imageView.

Esta classe contém uma lista de urls, onde os links URL das imagens vão ser armazenados.

Através de uma instancia da classe HttpFetchData, chamamos um processo numa outra threads, para fazer o retrieve de um ficheiro JSON disponível no servidor, e a partir daí accedemos ao array disponível nesse ficheiro JSON, com a key de "urls" e ao longo da iteração desse array(leitura), adicionamos esses urls à lista já mencionada.

# Estrutura de Código

## Images-continuação

Nesse mesmo momento notificamos o adapter que houve mudanças através de `adapter.notifyDataSetChanged();`

No final do método de `OnCreate()`, inicializamos o adapter e fazemos set do Adapter da ListView;

No Adaptador, e na sua função `View getView()`, fazemos load das imagens a partir dos urls recebidos da lista.

Este load de imagens para cada ImageViewer foi realizado através de uma libraria chamada :

Picasso <http://square.github.io/picasso/>

Esta libraria ajudou a fazer load das imagens sem precisarmos de realizar nós próprios processos em http ou processos outros threads (Async tasks).

## HttpFetchData

Esta classe contém todos os mecanismos para o funcionamento e retrieve de dos de internet em protocolo http(s).

Este processo contem uma Async task dado que se fosse realizado na Main threads iria bloquear os elementos de UI, e bloquear a aplicação, pois os dados da internet não são instantâneos e demoram tempo a serem buscados.

Esta classe também possui um Lista de HttpsListeners e um construtor que recebe unicamente a String do URL e realiza o processamento do método `execute()`, que por sua vez executa o método `fireSuccessResponseEvent` que por todos os elementos da lista mencionada realiza o método `onHttpResponseEvent`, que por sua vez adiciona o elemento à lista, sendo este um processo quase que recursivo.

Na função `dolnBackground()` estabelecemos uma conexão com um URL, para isso primeiro inicializamos um URL, sendo inicializado com uma String.

Declaramos um tempo de Timeout, tanto de connect, como de read,e também do método a utilizar(neste caso "Get"), de seguida incializamos um InputStream através da nossa conexão de URL, e depois também um BufferedReader, que irá armazenar em buffer os dados(Strings), que irão ser construídos a partir de uma StringBuilder , que irá fazer append de cada linha que irá ler da conexão estabelecida.

# Estrutura de Código

## Notebook

O notebook permite ao utilizador criar notas que vai averiguar durante o jogo, por isso, ele pode criar, apagar e até editar essas mesmas notas.

A actividade principal do notebook faz display da lista de notas que o utilizador criou, e apresenta um image button que lhe permite adicionar uma nota à lista.

Todos estes processos são realizados na base do Firebase, dado que lá é feita a escrita das notas como também a sua leitura. Decidimos utilizar a realtime DataBase devido à facilidade do seu uso.

Quando um dos items da listView é clicada, existe a inicialização de uma actividade que permite ver em detalhe o conteúdo da nota, enquanto que também permite a edição dos espaços de texto editável através do clique no botão de edição presente no canto superior direito.

A utilização da método `void SetTextIsEditable(boolean condition)`, permite-nos desativar a edição do texto ou ativá-la, sendo este método activado por um botão de edição presente no menu da actividade.

Para permitir a gestão das notas e a sua identificação a classe Notes uma string `keyID`, que é modulada através do método `UUID.randomUUID()`, dando um ID único a cada nota, isto permite que quando o utilizador edite a nota o programa saiba em que Note irá fazer override na base de dados ( se o ID fosse algo como o título, se houvesse 2 notes com títulos iguais então não saberíamos em qual fazer override).

Para acedermos às notas que estão na DataBase, percorremos com a ajuda de um DataSnapshot todas as notas do utilizador e adicionamos-las ao array de Notes que irá ser usado na listView, com a ajuda do Adapter.

Quando uma das notas é clicada ou o botão de adicionar nota é accionado, é enviado por Bundle o seu `keyID`, se este for nulo, é indicado à segunda actividade que esta nota é nova, e daí é atribuido um `keyID`, como já mencionado, se não for nulo, então sabemos que nos encontramos numa nota antiga e assim não lhe atribuimos um `UUID`, pois esta já o apresenta.

Quando nos encontramos na actividade de editar a nota, possuímos uma AlertDialog, a seguir ao clique no botão de eliminar nota, que nos permite ter uma confirmação do utilizador se real-