App Install & Go

Roberto Filipe Manso Barreto
(nrº 21123, regime diurno)

Orientação de
Luís Gonzaga Martins Ferreira

Licenciatura em Engenharia em Sistemas Informáticos
Escola Superior de Tecnologia
Instituto Politécnico do Cávado e do Ave

Identificação do aluno

Roberto Filipe Manso Barreto
Aluno número 21123, regime diurno
Licenciatura em Engenharia em Sistemas Informáticos

Orientação

Luís Gonzaga Martins Ferreira

Informação sobre o Estágio

Motorline Eletrocelos S.A
Travessa do Sobreiro, 29 Rio Côvo (Sta. Eugénia) 4755-474 Barcelos
Eng. Helder Remelhe

Resumo

Resumo do trabalho realizado. Deve ser sucinto, e cobrir todo o relatório: uma introdução ao problema que se pretendeu resolver, um pequeno resumo da abordagem realizada, e algumas conclusões do trabalho atingido.

Poderão ser criados vários parágrafos, até para que cada um corresponda às três fases de introdução, desenvolvimento e conclusão.

Não é relevante colocar no resumo o local de estágio ou a referência ao curso. Essa informação já consta da capa.

Abstract

This is the translation of the previous text. It should say the exact same thing. Please do not use directly Google Translator.

Agradecimentos

[A secção de agradecimentos é a parte pessoal do documento, e o único sítio onde o aluno pode escrever de forma menos formal, usando o tipo de linguagem que lhe parecer adequado para as pessoas a quem agradece.]

Conteúdo

1. Trabalho desenvolvido

1.1 Frontend

1.1.1 Organização do projeto

Assim como o backend o modelo a seguir para o frontend foi o MVC.

Como recomendado de boas práticas de código limpo da framework as cores do tema da aplicação foram todas colocadas em um ficheiro separado de forma a garantir que a troca de tema da aplicação é facilitada. Outras aplicações de boas práticas de código limpo foram, sempre que possível particionar o código das páginas em vários widgets de forma a ser de fácil navegação e também a criação de widgets reutilizaveis de forma a evitar a repetição de código e também para agilizar o desenvolvimento de código. Sendo assim a estrutura do projeto foi organizada da seguinte forma:

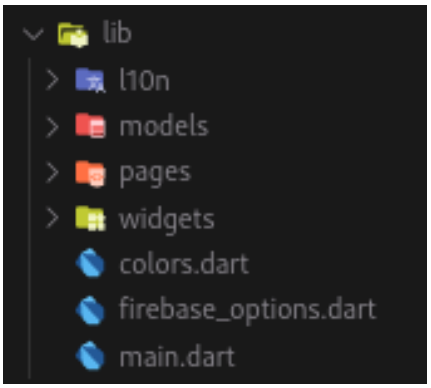


Figura 1.1: Organização do projeto

- **l10n** - Traduções da aplicação;
- **models** - Modelos de classes como handlers, helpers, providers, entre outros;
- **pages** - Páginas da aplicação;
- **widgets** - Widgets referentes às páginas;

1.1.2 Extensions

A linguagem de programação dart, assim como outras linguagens de programação orientadas as objetos permite alterar e adicionar métodos aos objetos base da linguagem, para realizar isso são criadas extensões do objeto, neste caso foi criada uma extensão para o objeto string de forma a facilmente capitalizar um texto.

1.1.3 Handlers

Os handlers são porções de código que permitem a execução de código perante um evento como por exemplo realizar uma ação perante um clique no ecrã, neste caso os handlers foram utilizados para detetar o estado da aplicação e realizar ações perante estes estados. Os estados da aplicação referem-se a se a aplicação se encontra em primeiro plano, segundo plano, a resumir ou então desligada. Com estes handlers é possível alterar o funcionamento da aplicação perante estes estados, como por exemplo tratar de notificações da aplicação.

1.1.4 Providers

Os providers são classes criadas para ajudar com comunicações externas, neste caso chamdas à API, estes providers foram criados de acordo com os diversos modelos de dados que se recebem, como por exemplo, uma publicação do fórum. Um provider oferece perante um modelo um conjunto de métodos para as diferentes chamadas necessárias à API, utilizando o exemplo anterior, para uma publicação existem métodos para eliminar, adicionar, editar e obter dados, sendo que acada método terá os seus requisitos do serviço que invocam.

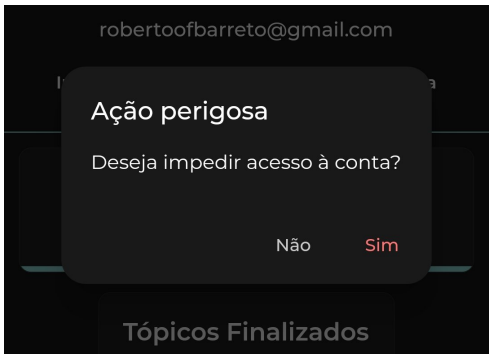
Estes providers automaticamente detetam a linguagem da aplicação e realizam o pedido ao serviço utilizando essa linguagem.

1.1.5 Helpers

Os helpers como o próprio nome indica são classes que ajudam com a realização de uma tarefa, neste caso os helpers, foram utilizados no auxílio de mensagens de notificações, estas mensagens deveriam conter o nome do utilizador e também a ação do mesmo traduzida na linguagem da aplicação, sendo assim foi criada a classe de helper de notificação que contém um método estático para obter a mensagem de uma notificação de acordo com a ação da mesma.

1.1.6 Gestão de utilizadores

Um requisito da aplicação e que apenas as empresas têm a possibilidade de se resgistarem, pelo que apenas estas poderão registar os seus técnicos. Para isso foi criada a página de gestão de utilizadores apenas acessível às empresas, nestas páginas estas poderão pesquisar pelos seus técnicos, ou registar novos técnicos, sendo que tem se ser obrigatóriamente indicado o nome, email e tipo de utilizador. Outras ações que este utilizador poderá realizar é a visualizar um técnico onde poderá bloquear o acesso ou até apagar a conta do mesmo, sendo que esta ação e irreversível.



(a) Aviso de impedir acesso à conta

(b) Aviso de remover conta

Sempre que um utilizador sem acesso ou com a conta apagada efetua o login, este receberá uma mensagem de erro mencionando que a sua este não possui acesso à conta impedindo assim o acesso.

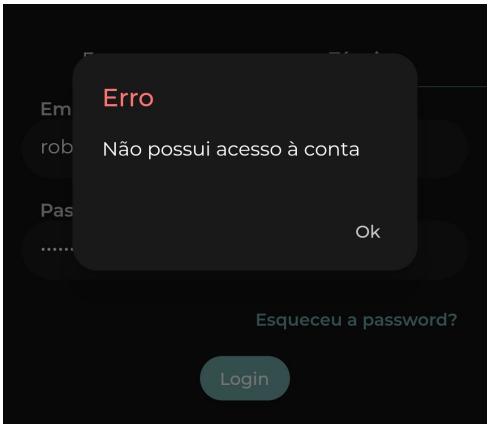
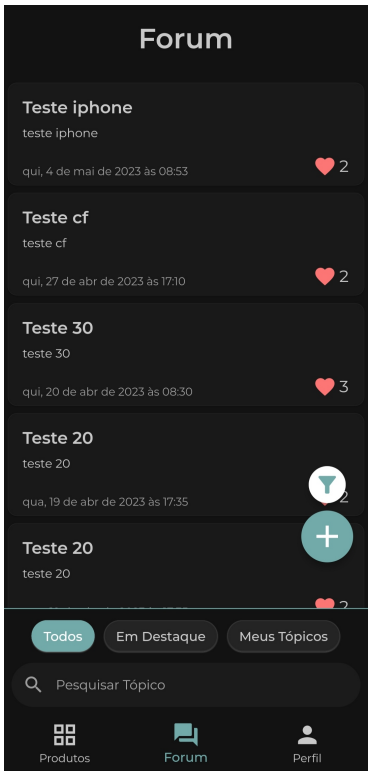


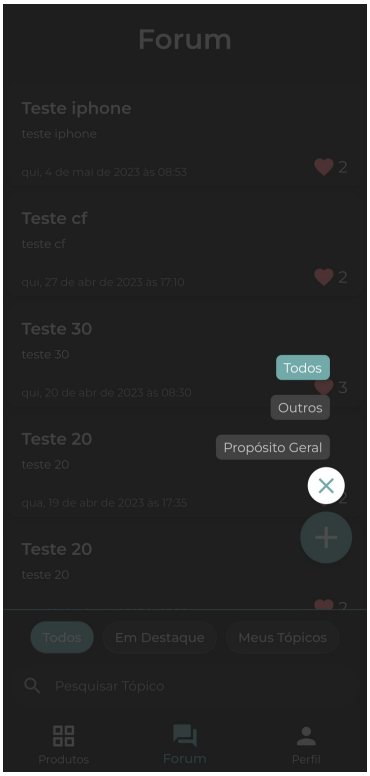
Figura 1.3: Aviso de login a conta sem acesso

1.1.7 Fórum

O desenvolvimento da página de fórum trouxe diversas dificuldades entre elas a gestão de filtros e pesquisas e também a mostragem de publicações e atualização das mesmas.



(a) Página de forum



(b) Filtragem de tipo

1.1.7.1 Filtragem de tópicos

O grande problema com a filtragem dos tópicos é que existem 3 tipos de filtros, o filtro de categoria de tópico, o filtro de tipo de tópico e o filtro de pesquisa.

Uma vez que algum filtro seja alterado, os filtros seguintes deverão ser novamente executados de forma a garantir que todos estes se encontram aplicados, inicialmente este tipo de filtragem não era executado, o que levava a problemas como por exemplo sempre que se efetua uma pesquisa, esta não era efetuada sobre as publicações filtradas o que levava a que a pesquisa fosse efetuada por todas as publicações.

Outro problema encontrado era na troca de categoria, por vezes acontecia que os filtros de categoria adicionavam-se o que levava a que estes não mostrassem publicações.

Sendo assim foram criados métodos para auxilio na filtragem, assim como também uma prioridade, sendo que a cada método invoca o método seguinte em forma de encadeação. Primeiramente aplica-se o filtro de categoria, de seguida este envia o resultado da filtragem para o método de filtragem por tipo e por fim se existir algum tipo de pesquisa os tópicos serão filtrados pela mesma.

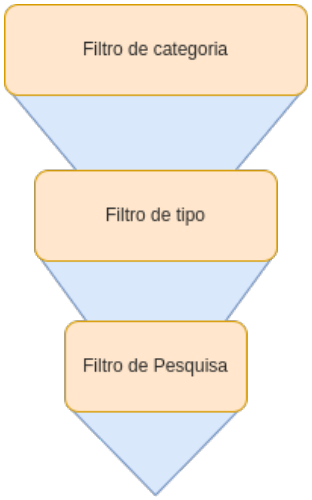


Figura 1.5: Filtragem do forum

1.1.7.2 Carregamento de tópicos

Inicialmente o carregamento de tópicos fazia-se por inteiro, desde carregamento de todos os tópicos, até ao carregamento de todos os dados dos mesmos, pois visto que não existiam muitos tópicos este não seria um problema para a API, mas conforme os testes foram sendo realizados a quantidade de tópicos existentes foi sendo incrementada, sendo possivel visualizar o tempo de demora de resposta do servidor a aumentar, assim como também a performance da aplicação no fórum a piorar.

De forma a resolver este problema primeiramente pensou-se em uma técnica de sliding window no total, o problema é que esta solução iria levar a que se o utilizador desejar voltar para trás nos tópicos a partir de uma quantidade escolhida de tópicos estes teriam de ser carregados novamente, o que não levaria a uma boa experiência de utilização.

Para resolver este problema foi utilizada a ideia de sliding window, mas os tópicos iriam se manter carregados, sendo que a própria framework consegue através da lista retirar de renderização os tópicos que o utilizador não consegue ver.

Esta solução foi implementada utilizando 3 valores, quantidade de tópicos a obter, índice inicial e data do primeiro tópico. O valor de quantidade de tópicos a obter, inicialmente 10, permite limitar a quantidade de tópicos que a API irá processar reduzindo o tempo de resposta, o índice inicial, permite indicar qual o índice do primeiro elemento que se deseja obter da lista e a data do primeiro tópico permite manter uma referência temporal para obter tópicos, garantindo assim que a lista que se está a visualizar é sempre a mesma.

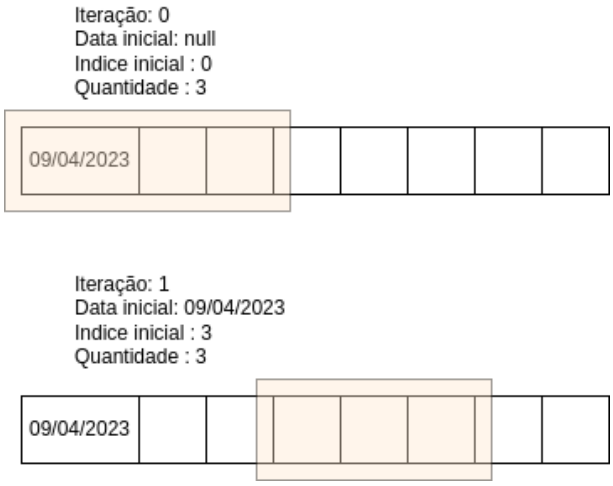


Figura 1.6: Carregamento de tópicos

Foi também reduzido a quantidade de dados carregados por cada tópico, sendo assim apenas os comentários diretos à publicação são carregados não carregando as respostas a estes, o que também contribuiu para a melhoria de performance.

Por fim sempre que o utilizador alcança o fim da lista de tópicos este poderá deslizar para carregar mais tópicos.

1.1.7.3 Detalhes de tópico

A página de detalhes de tópico sofreu os mesmos problemas que a página anterior sendo necessário aplicar a mesma solução sobre os comentários de tópico e também sobre as respostas ao mesmo. Sendo assim são carregados os primeros 10 comentários e por fim mostrado ao utilizador quantos mais comentários existem que poderá carregar sendo o limite 10 comentários.

Estes comentários podem também conter respostas sendo que estas podem ser carregadas também 10 de cada vez conseguindo o utilizador esconder ou mostrar estas.

Um problema que surgiu no desenvolvimento da página de detalhes de tópico foi também o destaque de uma mensagem para por exemplo destacar alguma resposta de um notificação. Este foi um grande problema pois com a nova implementação as mensagens não se encontram carregadas no momento de destacar a mensagem, pelo que e necessário procurar a mesma dentro das mensagens carregadas, expandindo assim as respostas do comentário que contêm a mensagem a destacar.

O destacamento de mensagens também continha um erro no qual sempre que algo no ecrã e atualizado, este recarregava a animação pelo que este código teve de ser movido de forma a apenas ser executado no momento de inicialização do ecrã após todos os elementos se encontrarem devidamente carregados.

Figura 1.7: Destaque de mensagens

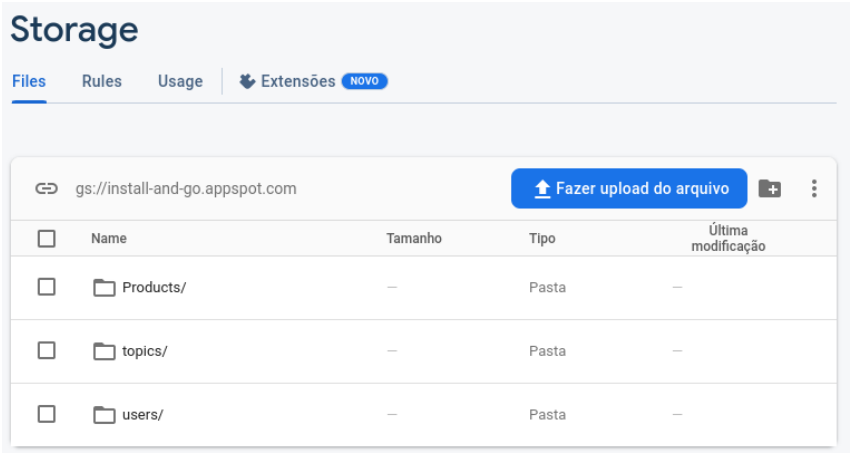
1.1.8 Firestorage

Visto que a desenvolvedora do Flutter e do Firebase é a mesma, esta disponibilizou recursos que facilitam a utilização desta ferramenta pelo Flutter, sendo assim todas as imagens e videos de utilizadores, publicações e comentários são guardadas diretamente da aplicação para o firestorage, assim como o acesso às mesmas é realizado diretamente.

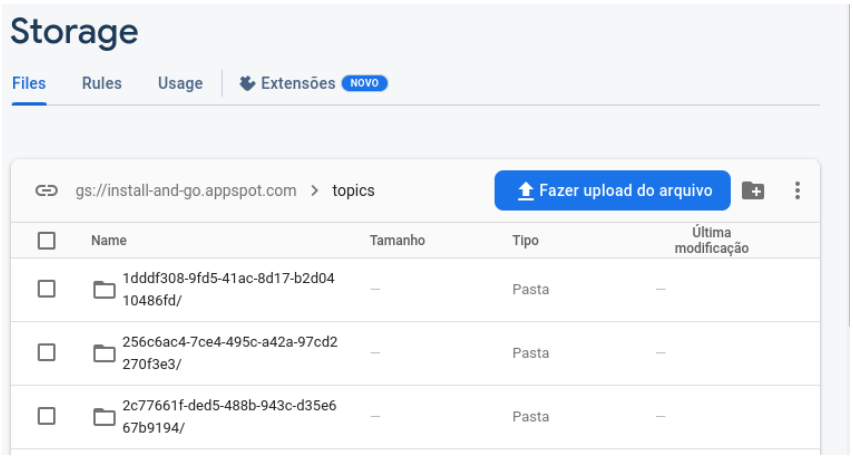
Para permitir este tipo de acesso o Firebase disponibiliza de um configurador de projeto que permite através do terminal configurar a ligação entre o projeto e o servidor do firebase, sendo no final apenas necessário importar a biblioteca do serviço do firebase que se deseja invocar a classe do mesmo sempre que se deseja realizar alguma ação.

Os ficheiros ficam então organizados de acordo com o seu contexto, no caso de utilizadores existe a pasta utilizadores, no caso de tópicos existe a paste tópicos e no caso de comentários existe a pasta comentários.

Dentro dos utilizadores como cada utilizador apenas contém uma imagem então estas são guardas com o nome do id do utilizador substituindo a imagem anterior se existir. No caso de tópicos e comentários como podem conter várias imagens e videos, então estes são guardados em pastas com ids dos mesmos contendo dentro destas os ficheiros referentes.



(a) Raiz do firestorage



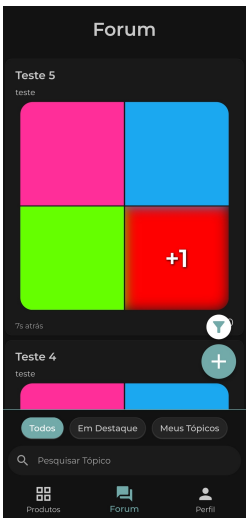
(b) Pasta topics do firestorage

1.1.9 Apresentação de Imagens

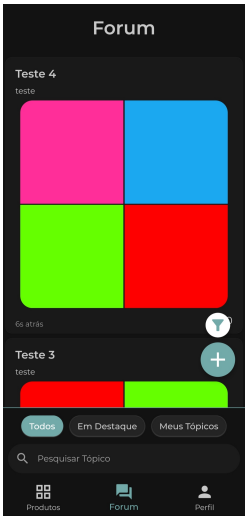
A apresentação de imagens é definida em 2 niveis, a nivel de pré-visualização, por exemplo ver a miniatura da imagem de uma publicação no forum e a nivel de detalhe, onde é possivel ver a imagem em ponto grande e realizar zoom na mesma.

Para a apresentação de miniatura da imagem foi decidido mostrar até 4 imagens, sendo que acima de 4 imagens seria mostrado apenas 3 imagens significando assim que a quarta imagem indicaria quantas mais imagens existem para mostrar.

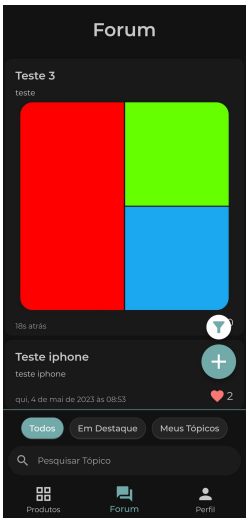
Para a implementação da apresentação das miniaturas de imagens, foi utilizada a biblioteca staggered_grid_view, esta biblioteca permite organizar imagens em grelha. Neste contexto desejava-se organizar estas imagens em diferentes aspetos, e disposições, pelo que esta biblioteca permite indicar quantas colunas e linhas existem na grelha ao criar o agrupamento de imagens, sendo assim foi decidido que quando são duas imagens estas dividem a grelha, quando são três imagens a primeira divide metade da grelha e as outras duas dividem a outra metade, quando são 4 ou mais então as 4 imagens dividem a grelha por igual. Quando se encontram existentes mais do que 4 imagens foi então decidido colocar um filtro de desfoque sobre a ultima imagem e colocar por cima desta quantas mais imagens existem para mostrar.



(a) Publicação com 5 imagens



(b) Publicação com 4 imagens



(c) Publicação com 3 imagens

1.1.10 Apresentação de Imagens em carrosel

A apresentação de imagens deverá permitir que o utilizador as visualize em ponto grande conseguindo também realizar diversas ações sobre estas, para isso foram experimentadas diversas bibliotecas, mas nunca se conseguia o comportamento desejado, sendo assim foi decidido criar o próprio carrosel de imagens, sendo que o próprio flutter já disponibiliza um widget para tal.

O ponto de maior dificuldade para este processo foi a implementação de zoom, visto que o flutter não dispõe de widgets para tal, sendo assim foi necessário primeiramente detetar gestos com o detetor de gestos da ferramenta e aplicado um zoom sobre o centro do gesto, os gestos aceites foram o de pinça e o gesto de duplo clique.

O grande problema com esta solução é que visto que é permitido um scroll horizontal de imagens os gestos por vezes poderão não funcionar corretamente, principalmente o gesto de pinça que se efetuado na horizontal poderá resultar em scroll horizontal. Para resolver tal problema foi decidido que quando dois dedos são detetados no ecrã a navegação horizontal fica bloqueada e assim que estes são levantados, a navegação horizontal é ativada novamente.

1.1.11 Carregamento de Imagens

O carregamento de imagens do dispositivo poderá ser realizado por meio de seleção de imagens da galeria, para realizar este tipo de seleção primeiramente foi testada a biblioteca image_picker, mas esta biblioteca utiliza o seletor de ficheiro do dispositivo, o problema é que este tipo de seleção permite também a seleção de ficheiros também o que faz com que seja necessário um conjunto de outras verificações para garantir que apenas as imagens são selecionadas o que levaria a uma possível perda de performance e perda de qualidade na experiencia de utilização do utilizador.

Sendo assim foi de seguida testada a biblioteca advance_image_picker, mas o problema desta biblioteca é que não permite selecionar vídeos, pelo que foi decidido experimentar a biblioteca wechat_asset_picker, esta biblioteca cria uma página de seleção de imagens própria para seleção de imagens e vídeos diretamente da galeria, esta permite indicar o limite máximo de seleção e os tipos de ficheiros que o utilizador poderá selecionar de forma a que apenas os tipos aceites são mostrados, acrescentando também que esta biblioteca permite visualizar a tipagem de cada ficheiro no próprio objeto, o único ponto desvantajoso é que não é possível traduzir o botão de confirmação de seleção.

Após a carregar os ficheiros para memória, estes são então enviados para o firestorage como mencionado anteriormente.

1.1.12 Videos

A grande dificuldade encontrada com os vídeos foi o facto de ser necessário um comportamento diferente nestes quando se encontram no ecrã de visualização de imagens e vídeos, pois ao contrário de imagens, os vídeos necessitam de um player, sendo sempre necessário verificar qual o tipo de ficheiro antes de carregar o widget do mesmo.

Para resolver o problema de utilizar um player foi primeiramente testada uma biblioteca que permite a utilização do player nativo do dispositivo, ou seja, android utilizaria o player do android e ios utilizaria o player de ios. O grande problema com esta solução é que o player de android tem o botões completamente brancos, sem nenhum tipo de fundo para os destacar significando isto que se um video branco fosse visualizado, o utilizador não conseguiria visualizar os botões do player.

Após o problema anterior foi decidido desenvolver o player próprio, após a implementação de funções como, pausar, resumir video, avançar 5 segundos e voltar 5 segundos, assim como também esconder e mostrar os botões, existiam 2 grandes problemas, sendo estes mostrar o video em ponto grande e retornar para o mesmo tempo de video em ponto pequeno e também o comportamento do player não era completamente fluido.

Após alguma pesquisa foi percebido que o player de ios resolvia os problemas do player de android através da colocação de um fundo nos botões do player. Através da biblioteca appinio, é possível utilizar o player nativo de ios em android, sendo também possível configurar este. Sendo assim foi decidido utilizar o player de ios em ambos os sistemas operativos resolvendo assim todo o problema. Este player também permitiu a resolução de um problema menor que era a visualização de vídeos em ponto grande sendo que dependendo a orientação do vídeo o player alterava a orientação da aplicação voltando à orientação vertical uma vez que se termina a visualização do vídeo em ponto grande.

1.1.13 Links

Uma funcionalidade da aplicação necessária é também a utilização de links, para isto programação mobile oferece duas soluções, deep links e dynamic links. Como mencionado na secção de tecnologias foi decidido implementar a solução de dynamic links da firebase.

Para implementar esta solução primeiramente a nível de backend foi necessário gerar os links, para isto, existem 2 soluções, implementação do firebase no próprio backend ou então uma chamada ao firebase utilizando uma chave de pedido. Primeiramente foi testado a implementação do firebase no próprio backend, mas esta implementação surge com alguns problemas pois existem diversas configurações especificas necessárias, sendo então recomendado pelos colegas de trabalho a chamada ao firebase devido à sua simplicidade.

Sendo assim para a realização de chamadas ao firebase foi utilizado o axios, realizando assim um método post para o serviço de dynamic links do firebase, indicando o o prefixo do projeto. De forma a permitir a reutilização deste código, este foi então colocado em um método onde este é chamado indicando o link desejado e os dados a enviar. O link é utilizado para por exemplo como em uma página web, indicar qual página se deseja direcionar o utilizador, já os parametros, assim como em um url web, são enviados através do próprio link, sendo assim estes dados são colocados na string do link permitindo a configuração do mesmo perante diversas situações.

Para a implementação de frontend foi necessário primeiramente importar a biblioteca de dynamic links do firebase, sendo de seguinda no código de inicialização da aplicação colocado um método para em caso de a aplicação ser aberta a partir de um link, este o ler. Quando este método lê o link primeiramente este extrai a página indicada pelo link e de seguida extrai a lista de parametros recebidos, sendo então o utilizador redirecionado para a página do link indicando como parametros os dados recebidos no link.

Aquando a testagem da implementação diversas tentativas de abertura de links foram realizadas mas sem sucesso, a grande dificuldade desta implementação é que os links dinamicos não permitem realizar debug, sendo que ou funcionará na totalidade ou então não funcionará lenvando que seja complicado identificar bugs, pelo que a documentação do serviço foi de grande auxilio pois estava em falta a indicação do nome do pacote da aplicação para android e ios, após a colocação destes dados, tudo seguiu em completo funcionamento.

1.1.14 Links

Uma funcionalidade da aplicação necessária é também a utilização de links, para isto programação mobile oferece duas soluções, deep links e dynamic links. Como mencionado na secção de tecnologias foi decidido implementar a solução de dynamic links da firebase.

Para implementar esta solução primeiramente a nível de backend foi necessário gerar os links, para isto, existem 2 soluções, implementação do firebase no próprio backend ou então uma chamada ao firebase utilizando uma chave de pedido. Primeiramente foi testado a implementação do firebase no próprio backend, mas esta implementação surge com alguns problemas pois existem diversas configurações especificas necessárias, sendo então recomendado pelos colegas de trabalho a chamada ao firebase devido à sua simplicidade.

Sendo assim para a realização de chamadas ao firebase foi utilizado o axios, realizando

assim um método post para o serviço de dynamic links do firebase, indicando o o prefixo do projeto. De forma a permitir a reutilização deste código, este foi então colocado em um método onde este é chamado indicando o link desejado e os dados a enviar. O link é utilizado para por exemplo como em uma página web, indicar qual página se deseja direcionar o utilizador, já os parametros, assim como em um url web, são enviados através do próprio link, sendo assim estes dados são colocados na string do link permitindo a configuração do mesmo perante diversas situações.

Para a implementação de frontend foi necessário primeiramente importar a biblioteca de dynamic links do firebase, sendo de seguinda no código de inicialização da aplicação colocado um método para em caso de a aplicação ser aberta a partir de um link, este o ler. Quando este método lê o link primeiramente este extrai a página indicada pelo link e de seguida extrai a lista de parametros recebidos, sendo então o utilizador redirecionado para a página do link indicando como parametros os dados recebidos no link.

Aquando a testagem da implementação diversas tentativas de abertura de links foram realizadas mas sem sucesso, a grande dificuldade desta implementação é que os links dinamicos não permitem realizar debug, sendo que ou funcionará na totalidade ou então não funcionará lenvando que seja complicado identificar bugs, pelo que a documentação do serviço foi de grande auxilio pois estava em falta a indicação do nome do pacote da aplicação para android e ios, após a colocação destes dados, tudo seguiu em completo funcionamento.

1.1.15 Ios

Bibliografia

Jin, Brenda, Saurabh Sahni & Amir Shevat. 2018. *Designing web apis*. O'Reilly Media, Inc.

Masse, Mark. 2011. *Rest api design rulebook*. O'Reilly Media, Inc.