Dedico este trabalho a minha mãe, que sempre esteve, está e estará comigo.

AGRADECIMENTOS

Agradeço a minha família, em especial meus pais, Joseane e Evandro, por me apoiarem e acreditarem em mim, sempre fazendo o possível e o impossível para que eu conseguisse concluir está etapa. Agradeço a minha "família"de Lavras, Julia, Bruno e Victor e amigos, Gustavo, Ivan e Bruno, por viverem e compartilharem dessa fase comigo. Agradeço ao professor Neumar, pela oportunidade deste projeto, orientação e conhecimento passado.

Espaço reservado a epígrafe.

(Autor Desconhecido)

RESUMO

A quantidade de usuários que utilizam dispositivos móveis para acesso a serviços e aplicações na Internet está cada vez maior. Desta forma, o desenvolvimento de aplicativos para dispositivos móveis é uma área muito relevante. Há diversos desafios no projeto e implementação de soluções para dispositivos móveis, por exemplo, a heterogeneidade de dispositivos e plataformas. As diferenças entre os dispositivos e suas linguagens nativas tornam difícil a implementação de aplicativos e sua portabilidade. Neste trabalho, foi desenvolvido um aplicativo para uma plataforma de notícias na Web. O aplicativo foi implementado com a tecnologia denominada Flutter e ele interage com um serviço RESTful para acessar os dados da plataforma. Este relatório contemplará as atividades de projeto e implementação do aplicativo.

Palavras-chave: Aplicativos. Flutter. RESTful.

ABSTRACT

The number of users using mobile devices to access services and applications on the Internet is increasing. Thus, mobile app development is a very relevant area. There are several challenges in designing and implementing mobile device solutions, for example, device and platform heterogeneity. Differences between devices and their native languages make application deployment and portability difficult. In this work, an application for a web news platform was developed. The application was deployed using technology called Flutter and it interacts with a RESTFul service to access the platform data. This report will cover the project activities and application implementation.

Keywords: Application. Flutter. RESTful.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

O mundo está cada vez mais conectado, e a utilização de dispositivos móveis para acesso a Internet e aplicações se torna cada vez maior. Em 2018, houveram 137.112.147 acessos a Internet por dispositivos móveis, contra 67.552.948 acesso via computador (TIC Domicílios, 2018). Portanto, é de grande importância o desenvolvimento, em conjunto a aplicações web responsivas, um aplicativo para estes dispositivos.

O objetivo deste relatório é apresentar os conceitos por trás de aplicativos para dispositivos móveis, tal como demonstrar a aplicação de um framework de desenvolvimento híbrido denominado Flutter no desenvolvimento de uma plataforma de notícias na Web.

Este relatório tem a seguinte disposição. O Capítulo 2 apresenta o referencial teórico, que aborda os conceitos básicos sobre o desenvolvimento de aplicações para aplicativos móveis e tecnologias envolvidas. O Capítulo 3 apresenta a parte prática do relatório como atividades e soluções de implementações. Por fim, o Capítulo 4 apresenta comentários e observações finais.

2 REFERENCIAL TEÓRICO

Neste capítulo, são descritos os principais conceitos e ferramentas utilizadas para o desenvolvimento desse projeto.

2.1 Desenvolvimento de aplicativos AINDA EM DESENVOLVIMENTO

No desenvolvimento para dispositivos móveis temos desafios como a heterogeneidade dos dispositivos, seja por parte do sistema operacional ou pelo tamanho dos dispositivos, capacidade de processamento, memória, disponibilidade de rede, entre outros. Começando pelo desenvolvimento nativo, que visa o desenvolvimento da aplicação com a linguagem da plataforma, precisa-se pensar em duas principais, Xcode utilizando a linguagem Objective-C para iOS e AndroidStudio com Java para Android.

É de se esperar que as empresas que utilizam desse método tenham duas equipes especializadas em cada um, aumentando assim o seu custo. Como forma de amenizar este problema, existem as metodologias de desenvolvimento cross-plataform que tem como finalidade disponibilizar à ambas as plataformas uma aplicação, utilizando apenas um código.

Como primeira solução cross-plataform, pode-se citar sites responsivos que não são aplicativos propriamente dito, mas que garantem a disponibilidade dos conteúdos necessários mesmo que necessitando de um browser para o acesso. A segunda solução é chamada de desenvolvimento hibrido que mistura o desenvolvimento web com apis nativas. Estes utilizam de WebView para funcionarem o que acarreta numa sensação menos fluida e nativa para os usuários.

2.2 Flutter AINDA EM DESENVOLVIMENTO

Flutter é um SDK (Kit de desenvolvimento de software) desenvolvido pela Google, para construir aplicativos para iOS e Android a partir de um único código.

Utiliza da linguagem de programação "Dart"no desenvolvimento do aplicativo e conta com o modelo reativo para interações do usuário.

Por fim, ao gerar o aplicativo, o motor em C/C++ do Flutter é compilado para código nativo utilizando Android NDK (Android) ou LLVM (iOS). Os códigos em Dart utilizam compilação AOT (compilação antes da execução) para geração de código nátivo ARM e bibliotecas x86.

2.3 RESTful API

2.3.1 HTTP e serviços web

HTTP (HyperText Transfer Protocol) é o protocolo que especifica a comunicação na Web (GOURLEY et al., 2002). O HTTP, situa-se na camada de aplicação e utiliza do protocolo TCP para o transporte dos dados.

A comunicação é feita de forma cliente-servidor, onde os conteúdos ou recursos estão localizados em um servidor HTTP, que ao serem requisitados por um cliente HTTP, retornam os dados em uma resposta HTTP.

A mensagem HTTP consiste de três partes:

Linha de começo, que indica qual o método e endereço da requisição (endereço do recurso), ou qual o código do estado da resposta.

Cabeçalho, que indica em conforma de chave-valor, informações da mensagem HTTP, como tipo de dado, conteúdos aceitos, línguas aceitas e etc. Finalizado por uma linha em branco.

Corpo, que contém os dados que serão enviados ou recebidos pelo servidor web. Estes dados podem ser de diversos tipos, como imagens, textos simples e linguagem de marcação. Nem todos os métodos HTTP utilizam do corpo, como é o caso do GET e DELETE.

Figura de mensagem HTTP

O HTTP suporta diferentes tipos de requisições, aos quais são chamados de métodos. Toda mensagem HTTP tem um método, e este, especifica ao servidor qual ação será feita.

Tabela 2.1 – Métodos HTTP comuns

Método	**Descrição**
`GET`	Retorna um recurso do servidor para o cliente.
`POST`	Envia um recurso ao servidor.
`PUT`	Atualiza um recurso do servidor.
`DELETE`	Exclui um recurso do servidor.

Fonte: Autor

Toda resposta HTTP tem um código de estado. Os estados são números de três dígitos e estão divididos em cinco classes (RFC,7231):

1xx (Informativo), informa que a requisição foi recebida mas ainda está sendo processada.

2xx (Sucesso), informa que a requisição foi recebida e aceita.

3xx (Redirecionamento), informa que ações devem ser tomadas para completar a requisição.

4xx (Erro no cliente), informa que a requisição tem erros de sintaxe ou não pode ser completada.

5xx (Erro no servidor), informa que o servidor falhou ao completar a requisição.

Tabela 2.2 – Códigos de estado HTTP comuns

Código	Descrição
200	OK. Requisição bem sucedida.
400	Bad Request. Requisição rejeitada pelo servidor por haver um erro do cliente.
403	Forbidden. Servidor entendeu a requisição mas não autoriza.
404	Not Found. O servidor não achou o recurso alvo.
405	Method Not Allowed. O método da requisição não está disponível no recurso alvo.
500	Internal Server Error. Servidor encontrou condições inesperadas que impedem de completar a requisição.

Fonte: Autor

2.3.2 Estilo arquitetural REST

REST (Representational State Transfer) é um estilo arquitetural para sistemas Web distribuídos (FIELDING; TAYLOR, 2000). Este, foi definido perante algumas restrições no âmbito de comunicação entre os componentes e manipulação de dados.

A primeira e mais perceptiva restrição do estilo, é a separação da interface do usuário dos dados e controles do sistema, seguindo o estilo cliente-servidor, facilitando a portabilidade e escalabilidade do sistema.

Outra restrição importante, é manter as comunicação sem estado, sendo assim, requisições feitas pelo cliente devem conter todas as informações necessárias para ser compreendida pelo servidor, evitando acoplamento de requisições e liberação de recurso mais rápida.

Respostas podem ser rotuladas como cacheáveis, dando o direito do cliente reutilizar os dados, melhorando a eficiência das requisição e uso dos recursos do servidor.

REST abstrai as informações como "recurso". Qualquer informação que pode ser nomeada pode ser um recurso, um documento, um serviço ou uma coleção de outros recursos, por exemplo. Como REST é construído nos conceitos

de HTTP, recursos são identificados por meio de URL, e as interações disponíveis com o recurso é definida pelos métodos HTTP.

Quadro 2.1 – Exemplo de representação de recurso REST

Identificação/URL	Método HTTP	Descrição
api/noticia	GET	Retorna todas as notícias
api/noticia	POST	Cria uma notícia
api/noticia/:id	PUT	Atualiza a notícia pelo id
api/noticia/:id	DELETE	Deleta a notícia pelo id

Fonte: Autor

3 ATIVIDADES REALIZADAS

Neste capítulo, são descritas as principais atividades e implementações realizadas no projeto.

3.1 Estrutura do projeto

O projeto foi estruturado em torno de quatro grandes pontos.

As telas, que são as telas do aplicativo, responsáveis pela renderização da interface do usuário, apresentação dos dados e chamada dos serviços.

Os componentes, que são Widgets que compõem as telas, tem as mesmas funcionalidades das telas, mas não existem sozinhos e não há navegação para os mesmos.

Os serviços, que são as classes responsáveis pela comunicação com a API e responsável pelo retorno dos recursos para as telas. Para a implementação da requisições foi utilizada a biblioteca "http"da linguagem Dart, que facilita a montagem e envio de mensagens Http.

Os modelos, que são as classes responsáveis por encapsular os recursos no sistema. Contam com métodos de parse de JSON, para transformação de dados recebidos da API em informações úteis para o sistema.

3.2 Visualização de notícias

A base do aplicativo é a visualização das notícias, sejam elas na página inicial (notícias de todos os canais), na página de um canal especifico (contendo apenas as notícias do mesmo), ou na tela de busca (filtrando apenas as notícias desejadas). Para evitar duplicação de código, foi desenvolvido um único componente, responsável pelo carrossel de notícias.

Este componente, além de ter atributos para mudança da distribuição de seus conteúdos (dado que a disposição do carrossel na tela de canal é um pouco

diferente das demais), conta com um teste para saber qual rota da API consumir. Caso um id de canal seja passado, a requisição feita será a GET de notícias de um canal. Caso uma String de busca seja passada, será feita a requisição GET de notícias onde o título contem a string de busca. Por fim, chegamos ao caso base, onde é feito uma requisição GET de todas as notícias.

Como todas as três requisições tem por finalidade retornar o recurso "notícia", pode-se garantir que a estrutura em ambas será a mesma, podendo assim armazena-las em uma List<News>, para renderização dos itens do carrossel de notícias.

3.3 Autenticação de usuário

Algumas rotas da API estão disponíveis apenas para aqueles que dispõem de autenticação no sistema. A mesma é feita no backend via token e este é recebido no header das requisições que o requerem.

No aplicativo ele é recebido no sucesso do serviço de login ou cadastro. Para ter rápido acesso a este token, o mesmo é salvo em uma classe global no aplicativo.

Como há a opção do usuário manter-se logado na plataforma, foi implementada uma solução utilizando a biblioteca "shared preferences", que empacota as funções nativas NSUserDefaults (iOS) e SharedPreferences (Android), que proveem persistência de dados simples. Assim, neste caso, o token é salvo tanto na classe global, quanto no dispositivo e sempre que o aplicativo é aberto o mesmo é lido.

Caso o usuário já não queira mais permanecer autenticado, foi implementada uma função para deletar este token do dispositivo e chamar o serviço de notificação do backend que o usuário já não está mais utilizando o token em questão.

A persistência dos dados com o sharedPreference dispõem de funções de set, get e remove para os tipos bool, int, double e String. Onde trabalham no padrão chave e valor.

Figura 3.1 – Exemplo da utilização do SharedPreference

```
Future<dynamic> getToken() async{
  var session = await SharedPreferences.getInstance();
  String token = session.getString(_token);
  return token;
}

void setToken(String token) async{
  await SharedPreferences.getInstance().then((session){
    session.setString(_token, token);
  });
}

void logout() async{
  await SharedPreferences.getInstance().then((session){
    session.remove(_token);
  });
}
```

Fonte: Autor

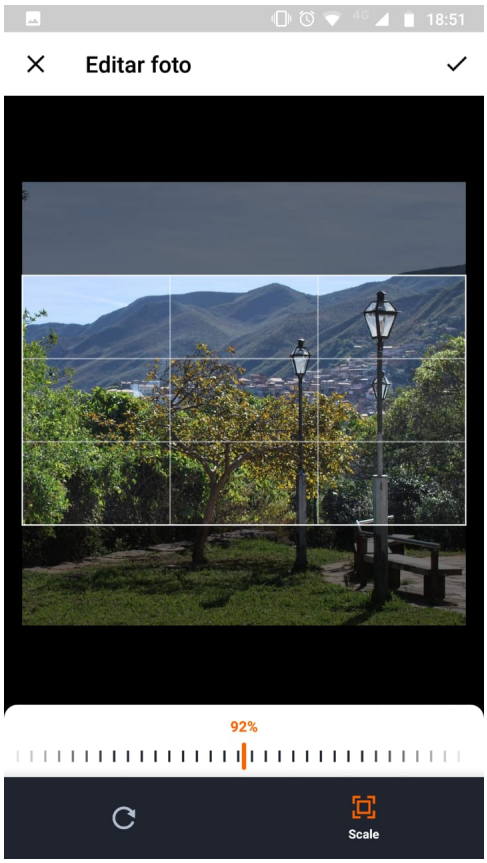
3.4 Enviar notícias

Para a implementação do envio de notícias foram utilizadas duas bibliotecas, "image picker", que empacota as funções nativas de acesso a galeria do celular e câmera. Necessitando apenas da criação da interface de usuário que chame as suas funções. Além de comportar compressão de imagem, que no cenário de aplicativos para dispositivos móveis é de extrema importância, pelo consumo e velocidade das redes móveis.

A segunda biblioteca, "image cropper", faz comunicação com duas bibliotecas nativas de corte e rotação de imagem, uCrop e TOCropViewController. Esta biblioteca já disponibiliza a interface de ajuste de imagens. No escopo deste pro-

jeto foi necessário a implementação desta função, pois era necessário que todas as imagens fossem padronizadas na proporção 16:9.

Figura 3.2 – Interface de ajuste de imagem

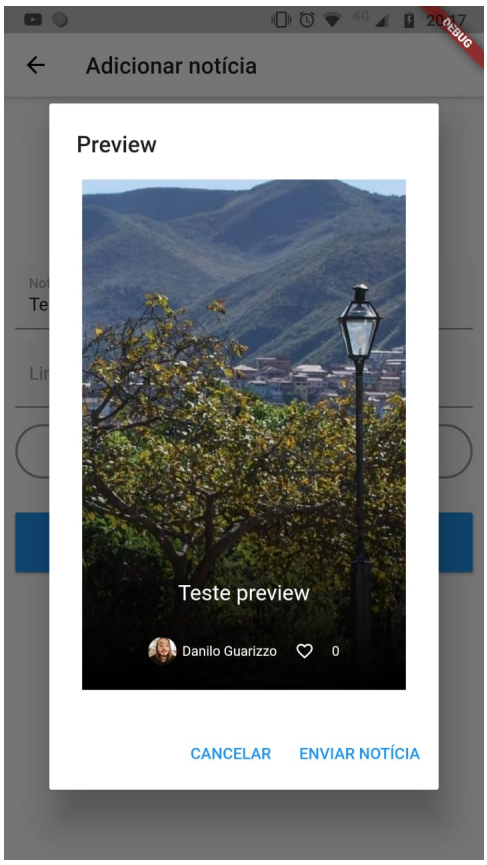


Fonte: Autor

Com estas duas bibliotecas foi implementado o caminho geral das imagens, que é a escolha e compressão de uma imagem, a utilização desta foto comprimida no recortador de fotos, para deixa-lá do tamanho necessário.

Também foi implementado uma função de prévia para a notícia. Após os dados da notícia serem preenchidos, antes do usuário enviar, uma modal com a notícia montada (como aparece no visualizar notícia) é mostrada a ele, para a validação da aparência e dados.

Figura 3.3 – Interface de preview de notícia



Fonte: Autor

3.5 Paginação de dados

Há recursos na aplicação que quando requisitados podem ser uma lista de *n* valores, podendo assim ter uma requisição lenta e pesada caso *n* seja muito grande, assim necessitando de paginação. Como exemplos estão as notícias e notificações da plataforma, onde para solucionar este problema as requisições destes recursos contêm atributos para definir como a paginação será feita, como número da página requisitada e quantidade de itens por página. Já na resposta, além de conter os valores do recurso requisitado, conterão também atributos como: número total deste recurso disponível e recursos abrangidos nesta consulta.

No caso das notícias como se tratam de recursos mais pesados, por conterem fotos com maior qualidade, foi escolhido um número de dez itens por página, sendo essa requisição chamada sempre que o usuário passar pela sétima notícia, deixando assim a leitura do usuário fluida e sem fazer requisições desnecessárias.

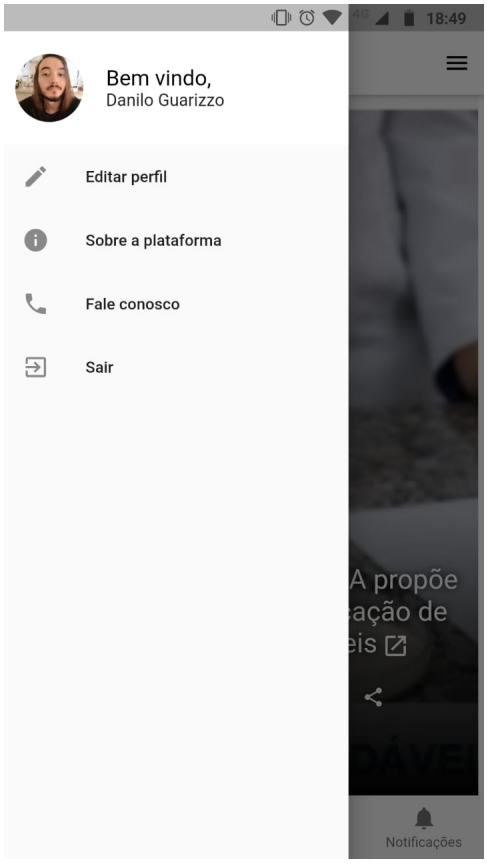
Isso foi atingido, ao implementar uma variável de controle a qual informa em qual notícia o usuário está no carrossel. Já para notificações, este controle é feito ao chegar no final da lista carregada, um evento é disparado pelo componente de lista do Flutter e a função de requisitar mais notificações e chamada.

3.6 Menu lateral (Drawer)

Graças ao Flutter a tarefa de desenvolver um menu lateral não se torna difícil. O Flutter disponibiliza um componente que contem os controles padrões de um menu lateral, como o ícone de abertura do menu e as ações de arrastar para abrir e fechar. Além do componente, é necessário estilizar e alocar as informações contidas no menu.

Na aplicação, o menu lateral foi utilizado para conter os menus de navegação referentes a informações do usuário, informações sobre a plataforma e ação de deslogar da plataforma.

Figura 3.4 – Resultado menu lateral



Fonte: Autor

4 CONCLUSÃO

O objetivo deste relatório foi apresentar a importância do desenvolvimento de aplicações para dispositivos móveis, conceitos por trás desta área e a maior facilidade gerada ao utilizar um framework de geração de aplicativos nativos híbridos.

Este projeto contribuiu para que eu estivesse a par das tecnologias e estrutura de projetos que estão sendo utilizadas nos dias de hoje, como RESTful API, que como neste projeto, facilitam a distribuição dos dados entre as plataformas e o desenvolvimento híbrido de aplicações para dispositivos móveis, que estão em alta no mercado, por terem a capacidade de criar aplicações em menor tempo e com custo menor.

Entre as dificuldades encontradas durante o desenvolvimento, pode-se destacar o aprendizado do padrão reativo, utilizado em muitas das tecnologias atuais, assim como no Flutter, que controla toda a renderização dos elementos e dados no aplicativo, o que era desconhecido no começo do projeto.

Pode-se ressaltar que o grande destaque para uma boa atuação na área de desenvolvimento, acima do conhecimento das tecnologias atuais, é uma boa base de lógica de programação, onde as matérias da universidade tem grande impacto.

REFERÊNCIAS BIBLIOGRÁFICAS

FIELDING, R. T.; TAYLOR, R. N. **Architectural styles and the design of network-based software architectures**. [S.l.]: University of California, Irvine Doctoral dissertation, 2000. v. 7.

GOURLEY, D. et al. **HTTP: the definitive guide**. [S.l.]: "O'Reilly Media, Inc.", 2002.