universidade de aveiro  theoria poiesis praxis

Projeto de ICM
(Módulo Flutter)

101043 - Marcel Souza
101099 - Victor Melo

SongLearn

A aplicação SongLearn tem como principal objetivo a criação de um ambiente virtual de aprendizagem de música e conexão entre os usuários, utilizando para isto aulas em vídeo, metrônomo,diapasão, mapas interativos entre os utilizadores e registo de novas aulas por meio de formulário. Além disso, fornecemos um scanner de QRCodes para acessar novas aulas, um criador de QRCodes para partilhar suas aulas e a possibilidade de buscar por letras de músicas de diversos artistas.

Com todas estas features, desejamos criar um ambiente propício para a aprendizagem online entre os utilizadores do nosso produto.

Solução geral e features implementadas

O nosso projeto utiliza como base o próprio flutter, alguns sensores acessados através de bibliotecas e o Firebase. Podemos elencar e descrever a utilização da seguinte forma:

- Registo e Login: O utilizador pode se registar e realizar login na plataforma. Para isso, deve preencher todo o formulário respeitando as regras para a criação de senha.
- Assistir aulas: O utilizador logado tem acesso a uma lista de aulas que pode pressionar para assistir dentro da aplicação.
- Upload de aulas: O utilizador logado pode realizar upload de aulas na plataforma e torna estas aulas acessíveis para todos os outros utilizadores.
- Metrônomo: O utilizador logado pode ter acesso a um metrônomo que funciona por meio de vibração. Inicialmente, utilizamos as 3 mais conhecidas BPM (batidas por minuto): 60, 90 e 120.
- Diapasão: O utilizador logado pode selecionar dentre as 6 Notas da afinação padrão de uma guitarra (EADGBE), onde reproduz a respectiva nota para que possa afinar o seu próprio instrumento.
- Busca de letras e álbum de um artista: O utilizador logado consegue buscar letras de canções de um determinado artista. Basta por o parte do nome da canção e o artista correspondente (opcional).
- Mapa Interativo: O utilizador logado tem acesso a um mapa interativo com marcações nas posições onde os outros utilizadores foram registados. Posteriormente, será alterado para o local onde os utilizadores estão.
- Scan de Lições e outros códigos: A aplicação contém um leitor de QRCode built-in. Assim, os utilizadores logados podem acessar outras aulas ou qualquer QRCode que desejem.
- Criação de QR Code exclusivo para cada aula, onde o utilizador pode assim compartilhar com colegas ou com alunos (no caso de um utilizador professor).
- Notificações diárias: Os utilizadores são notificados todos os dias às 10:00 para começarem os estudos diários de música por meio da nossa aplicação.

BLOC

No nosso projeto, o Bloc é utilizado para gerenciar a lista de vídeos e o estado de favoritos, e foi implementado da seguinte forma:

- A classe Video define o modelo de dados para um vídeo, contendo o nome, a URL e um indicador de favorito.

- A classe VideoBloc é responsável por gerenciar a lista de vídeos. Ela possui uma lista de vídeos privada, um objeto BehaviorSubject chamado _videosSubject e um getter videosStream que retorna um fluxo de vídeos.
- O método fetchVideos() na classe VideoBloc busca os vídeos do Firebase Storage e os adiciona à lista _videos. Em seguida, ele emite a lista atualizada para o fluxo _videosSubject usando o método add(). Se a lista estiver vazia, o fluxo é fechado chamando o método close().
- O método toggleFavorite() é chamado quando o usuário alterna o estado de favorito de um vídeo. Ele atualiza o estado de favorito do vídeo na lista e emite a lista atualizada para o fluxo _videosSubject.
- O método getFavoriteVideos() retorna uma lista de vídeos favoritos com base no estado de favorito na lista de vídeos.
- O método dispose() é responsável por fechar o fluxo _videosSubject quando não for mais necessário.

A classe VideoListPage é um widget StatefulWidget que exibe uma lista de vídeos. Ele possui um objeto VideoBloc que é passado como argumento durante a construção. O VideoBloc é inicializado no método initState() e é atualizado sempre que a página é recarregada. O método reloadPage() reinicializa o VideoBloc e busca os vídeos novamente.

A página exibe os vídeos em um ListView.builder, onde cada item da lista é um Card de vídeo. O estado de carregamento, erro ou sucesso é tratado no StreamBuilder que monitora o fluxo videosStream do VideoBloc.

A classe VideoPlayerPage é outro widget StatefulWidget que reproduz um vídeo usando o VideoPlayerController.

A classe MyApp é o ponto de entrada do aplicativo e define o VideoBloc como um membro da classe. É usado como argumento ao criar a VideoListPage.

Por fim, a classe MenuPage é outro widget StatefulWidget que exibe uma página de menu com um BottomNavigationBar. Ela também possui um objeto VideoBloc que é passado como argumento durante a construção e é usado para criar a VideoListPage no primeiro item do menu.

Widget Tree Rationale

A estrutura de árvore de widgets do main.dart (principal) é:

```
MaterialApp
└─ LearningMusicApp
   └─ _LearningMusicAppState
      └─ Scaffold
         ├─ Stack
         │  └─ Container
         │     └─ _pages[_selectedIndex]
         └─ Positioned
            └─ Container
               └─ BottomNavigationBar
                  ├─ BottomNavigationBarItem
                  ├─ BottomNavigationBarItem
                  ├─ BottomNavigationBarItem
```

```
                    ├─ BottomNavigationBarItem
                    ├─ BottomNavigationBarItem
                    └─ BottomNavigationBarItem
```

Uma breve explicação sobre cada widget:

- MaterialApp: Este é o widget raiz do app. Ele configura a aplicação com um tema, define o título e fornece a navegação entre as telas usando a classe Navigator.
- LearningMusicApp: Este é um widget StatefulWidget que representa o aplicativo Learning Music. Ele possui um estado interno e controla o índice da tela selecionada.
- _LearningMusicAppState: Este é o estado do LearningMusicApp. Ele é responsável por atualizar a tela com base no índice selecionado e controlar a instância do VideoBloc.
- Scaffold: É um widget que fornece a estrutura básica para as páginas do aplicativo, incluindo o appBar e o corpo da página.
- Stack: É um widget que empilha seus filhos um em cima do outro (children). Aqui, é usado para sobrepor o Container que contém a página selecionada sobre o BottomNavigationBar.
- Container: É um widget que fornece um layout flexível. Ele envolve a página selecionada (definida por _pages[_selectedIndex]) e ocupa todo o espaço disponível.
- _pages[_selectedIndex]: Representa a página selecionada com base no índice _selectedIndex. O conteúdo da página é definido em diferentes widgets, como VideoListPage, MapScreen, QRCodeScannerPage, VideoUploadPage, MetronomeScreen e LyricsPage.
- Positioned: É usado para posicionar o Container do BottomNavigationBar na parte inferior da tela.
- Container: É usado para envolver o BottomNavigationBar e fornecer uma borda superior.
- BottomNavigationBar: É o widget que exibe a barra de navegação na parte inferior da tela. Ele contém os ícones de navegação para as diferentes telas.
- BottomNavigationBarItem: Representa cada item na barra de navegação. Cada item possui um ícone e um rótulo associados. Aqui, todos os rótulos estão vazios para ocultá-los.

Firebase

Durante todo o projeto a utilização do Firebase está presente. Inicialmente, utilizamos o Firebase Auth para o registo e login de utilizadores. Ainda no registo, criamos também um nó para cada utilizador no Firebase Real-Time Database, com as informações de Latitude e Longitude do utilizador na hora do registo (obtidas por meio do GPS).

Para as aulas em vídeo, utilizamos o Firebase Cloud Storage, tendo como objetivo disponibilizar as aulas lá guardadas para todos os utilizadores da aplicação terem acesso ao conteúdo informativo.

Para finalizar, utilizamos o Firebase Messaging para enviar notificações aos utilizadores todos os dias às 10:00 (horário local do utilizador), com o objetivo de relembrar que deve estudar música por meio da nossa aplicação naquele dia.

Bibliotecas para sensores e API requests

HTTPDart: A biblioteca HTTPDart é usada para fazer solicitações HTTP em aplicativos Flutter. Ela fornece uma maneira fácil e eficiente de se comunicar com APIs e servidores da web. Foi utilizado para requests à API de letras de músicas.

google_maps_flutter: Essa biblioteca permite a integração de mapas do Google em seus aplicativos Flutter. Com ela, você pode exibir mapas interativos, adicionar marcadores, desenhar rotas e personalizar a aparência dos mapas. Foi utilizado para mapa interativo de utilizadores.

vibration: A biblioteca vibration permite que você controle a vibração do dispositivo em aplicativos Flutter. Com ela, você pode emitir vibrações curtas ou longas em resposta a eventos específicos no seu aplicativo.Foi utilizado para o metrônomo.

QrScan e URLScanner: Essas bibliotecas fornecem funcionalidades para ler códigos QR e escanear URLs em aplicativos Flutter. Com elas, você pode capturar e processar informações contidas em códigos QR, bem como extrair e manipular URLs de imagens ou texto. Foi utilizado para a câmera e scan de QRCodes na aplicação.

Geolocator: A biblioteca Geolocator é usada para obter a localização atual do dispositivo em aplicativos Flutter. Ela permite que você acesse informações como latitude, longitude, altitude e precisão do GPS. Foi utilizado para o registo de latitude e longitude dos utilizadores no Firebase.

Video_player: Essa biblioteca oferece recursos de reprodução de vídeo em aplicativos Flutter. Com ela, você pode exibir vídeos de diferentes formatos e controlar a reprodução, como pausar, retomar e avançar/retroceder. Foi utilizada para assistir às aulas dentro da aplicação.

file_picker: A biblioteca file_picker permite que você selecione arquivos em dispositivos para uso em aplicativos Flutter. Com ela, você pode abrir um explorador de arquivos no dispositivo e permitir que o usuário selecione um arquivo específico. Foi utilizada para seleção de vídeos para upload na nuvem do Firebase.

font_awesome_flutter: Essa biblioteca permite usar ícones do Font Awesome em um aplicativo Flutter. Ela fornece um conjunto abrangente de ícones vetoriais que podem ser facilmente integrados ao seu aplicativo, oferecendo uma ampla variedade de opções de personalização. Utilizada em toda a aplicação para adicionar ícones com maior qualidade e personalidade.

rxdart: é uma biblioteca de programação reativa para Dart e Flutter. Ela fornece uma implementação reativa dos Observables, Streams e Subjects do Dart, juntamente com uma série de operadores que permitem compor e manipular fluxos de dados assíncronos de maneira conveniente e poderosa. Utilizada na implementação do BLOC.

audioplayers: Essa biblioteca fornece uma maneira fácil de reproduzir arquivos de áudio em um aplicativo Flutter. Ela oferece recursos para controlar a reprodução de áudio, como reproduzir, pausar, retomar, parar, além de fornecer opções de controle de volume e avanço/retrocesso. É útil para reproduzir sons, música ou qualquer outro tipo de áudio em um aplicativo Flutter. Utilizazda na implementação do Diapasão (Tuning Fork) para reproduzir as notas musicais.

Avaliação Geral

Como avaliação geral do projeto, acreditamos que conseguimos cumprir todos os objetivos propostos inicialmente. Obviamente, entendemos que a aplicação ainda pode evoluir muito em diversos aspetos, tais como: perfis para professores, mapa com localização real (não do registo), implementação de um afinador e etc.

Uma dificuldade que tivemos de forma muito explícita foi para a implementação do afinador. Apesar de termos tentado implementar o mesmo, muitas das bibliotecas disponíveis tinham problemas de compatibilidade com o SDK que utilizamos inicialmente. Neste sentido, o erro de Null Safety era quase inevitável em todas as nossas tentativas.

Além disso, diferentemente do que tínhamos planeado inicialmente, não foi possível fornecer um sistema de buscas de partituras e tablaturas de guitarra, uma vez que todas as APIs externas de que

tínhamos conhecimento estão atualmente offline ou depreciadas. Ainda que apenas para letras de canções, a busca foi difícil, mas conseguimos encontrar esta opção ao menos.

No entanto, como um projeto de uma cadeira universitária e com pouco tempo para implementação, acreditamos que conseguimos atingir o objetivo inicial: que os utilizadores consigam aprender música de forma interativa, gratuita e inovadora.

A contribution assessment

50% - 50%

Marcel de Araújo Santos Souza:

- Implementação inicial de todas as features da aplicação (sem design e sem BLOC);
- Implementação de parte do relatório;

Victor Barros Melo:

- Implementação do BLOC no projeto;
- Implementação de parte do relatório;
- Conserto de todos os bugs na implementação das features;
- Implementação do Design, Responsividade, UX e UI de toda a aplicação;
- Implementação do tratamento de Erros nas páginas;

Neste sentido, o grupo considera justo a divisão 50% - 50%.

A Tutorial/Manual

Link do youtube com um vídeo mostrando algumas features da aplicação

Tutorial/Showcase Songlean

Link do Repositório no Github do projeto.