PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA E DE ARTES
GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

PUC
GOIÁS

APLICATIVO PARA GESTÃO E EXPOSIÇÃO DE COLEÇÕES

FERNANDO CARLOS BRANDÃO FILHO

GOIÂNIA
2024

FERNANDO CARLOS BRANDÃO FILHO

APLICATIVO PARA GESTÃO E EXPOSIÇÃO DE COLEÇÕES

Trabalho de Conclusão de Curso apresentado à Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação.

Orientador(a):
Prof. Me. Gustavo Siqueira Vinhal

Banca examinadora:
Prof. Me. Carlos Alexandre Ferreira De Lima
Prof. Dr. Nilson Cardoso Amaral

GOIÂNIA
2024

FERNANDO CARLOS BRANDÃO FILHO

APLICATIVO PARA GESTÃO E EXPOSIÇÃO DE COLEÇÕES

Trabalho de Conclusão de Curso aprovado em sua forma final pela Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, para obtenção do título de Bacharel em Ciência da Computação, em _____/_____/_____.

_____

Orientador(a): Prof. Me. Gustavo Siqueira Vinhal

_____

Prof. Carlos Alexandre Ferreira De Lima

_____

Prof. Nilson Cardoso Amaral

GOIÂNIA,
2024

Agradecimentos

Agradeço imensamente a minha noiva que não deixou de confiar em mim nem quando eu mesmo duvidei e foi a base que me sustentou quando fraquejei.

Agradeço a Deus pelas oportunidades que me apresentou e pela minha família que torce por mim sem parar.

Resumo

Este trabalho descreve o desenvolvimento de um aplicativo para colecionadores, que visa facilitar a gestão e visualização das coleções de itens em dispositivos móveis. O principal objetivo do aplicativo é permitir que os usuários organizem e visualizem suas coleções de forma prática e intuitiva, utilizando funcionalidades como a adição de itens, imagens e a visualização de coleções em formato de *cards* interativos. O desenvolvimento do aplicativo é realizado utilizando o Flutter, um *framework* de código aberto, com o gerenciamento de estado sendo feito pelo GetX, uma biblioteca que simplifica a manipulação de estados e dependências. Para comunicação com a API, é utilizado o Dio, um cliente HTTP que facilita as requisições de rede e manipulação de dados. O *backend* do sistema é implementado em Laravel, com um foco na criação de rotas e funções que permitem o cadastro e consulta das coleções e itens. Além disso, a manipulação de imagens é feita com o auxílio de pacotes como *ImagePicker* e *ImageCropper*, permitindo que os usuários escolham e editem imagens diretamente pelo dispositivo móvel. A integração entre o aplicativo e o *backend* ocorre através de requisições HTTP, que buscam informações e atualizam o banco de dados, com a conversão de imagens em formato base64 para armazenamento. O sistema também incorpora uma funcionalidade de exibição de imagens em formato base64, o que garante uma visualização de itens enriquecida, com a possibilidade de exibir essas imagens como fundo de *cards*. Este trabalho oferece uma solução integrada que atende às necessidades dos colecionadores, tornando a gestão de suas coleções mais acessível e organizada.

Palavras-chave: *Flutter*, GetX, *Laravel*, Coleções, Imagens, API.

Abstract

This work describes the development of an application for collectors, which aims to facilitate the management and visualization of collections of items on mobile devices. The main objective of the application is to allow users to organize and view their collections in a practical and intuitive way, using features such as adding items, images and visualization of collections in interactive card format. The application development is carried out using *Flutter*, an open source *framework*, with state management being done by GetX, a library that simplifies the manipulation of states and dependencies. To communicate with the API, Dio is used, an HTTP client that facilitates network requests and data manipulation. The system's backend is implemented in Laravel, with a focus on creating routes and functions that allow the registration and consultation of collections and items. Furthermore, image manipulation is done with the help of packages such as ImagePicker and ImageCropper, allowing users to choose and edit images directly from their mobile device. The integration between the application and the backend occurs through HTTP requests, which search for information and update the database, converting images into base64 format for storage. The system also incorporates a functionality for displaying images in base64 format, which guarantees an enriched visualization of items, with the possibility of displaying these images as the background of *cards*. This work offers an integrated solution that meets the needs of collectors, making the management of your collections more accessible and organized.

Keywords: Flutter, GetX, Laravel, Collections, Images, API.

Lista de Figuras

Lista de Siglas

RIUnB - Repositório Institucional da UnB

RAG - Repositório Acadêmico de Graduação da PUC Goiás

TCC - Trabalho de Conclusão de Curso

API - Application Programming Interface (Interface de Programação de Aplicações)

JWT - Json Web Token

UnB – Universidade de Brasília

Sumário

1 Introdução

Colecionar objetos não é um fenômeno recente, povos pré-históricos já demonstravam essa característica ao guardar, proteger e atribuir valores simbólicos a determinados objetos, mesmo que eles não tivessem relação com sua sobrevivência (OLIVEIRA, 2017). As coleções não são criadas com objetos sem algum tipo de valor, seja ele mercadológico ou sentimental, de outra forma seria apenas acumulação. Em uma coleção “cada item terá um significado em si mesmo e muito provavelmente uma história, e a coleção como um todo único terá também um significado e uma história. Dessa forma, uma coleção é basicamente determinada pela natureza do valor atribuído aos objetos” (HADDAD, 2018).

O desenvolvimento da sociedade levou esse hábito a evoluir e se diversificar. Cirne (2018) revela em seu trabalho o seguinte:

> “O ato de colecionar é inerente ao homem, tanto na salvaguarda de coisas quanto na produção de utensílios. Na Pré-História, além dos objetos desenvolvidos para utilização cotidiana, outros eram esculpidos para serem admirados, como estátuas de homenagens a deuses, ou para servirem como registro, vide as pinturas rupestres.
>
> […] Até o final da Idade Média, os conjuntos de objetos reunidos pelas sociedades ocidentais eram vinculados à religião, sendo dotados de significados. A partir do século 11, juntam-se objetos executados em materiais preciosos, além de livros e instrumentos científicos.
>
> É no período do Renascimento, com o crescimento do urbanismo, do comércio e da burguesia, que o ato de colecionar deixa de ser exclusividade da Igreja e das famílias reais. Surge o colecionismo privado, uma transformação que, segundo Costa (2007, p.31), foi desempenhada na tentativa de compreender o mundo.” (CIRNE, 2018, p.31)

No período do Renascimento, na Europa, eram muito comuns os gabinetes de curiosidades ou as salas privadas com todo tipo de objetos encontrados no Velho e e no Novo Mundo (CIRNE, 2018). Com o tempo essas coleções foram se espalhando e crescendo, Cirne (2018, p. 33) afirma ainda que “esta maturidade do

colecionismo privado resultou na criação dos primeiros museus de colecionadores ao final do século 19”.

Uma coleção pode começar sem a intenção inicial de se realizar uma mas “como um acidente ou como a manifestação de um ato espontâneo e apaixonado, e às vezes como um processo fundamentado em alguma motivação ou interesse específico” (HADDAD, 2018). O colecionador pode vir a sentir excitação e ansiedade apenas ao imaginar o objeto desejado, pensando em como se sentirão ao possuí-lo e integrá-lo a sua coleção. A busca e pesquisa sobre os itens de interesse exigem dos colecionadores muito empenho, comparando tal situação a uma “caçada” e a obtenção como a “emoção da caçada” (HADDAD, 2018).

A posse dos objetos colecionados dá aos colecionadores o sentimento de pertencimento a um grupo, seja ele formal ou não, onde seus integrantes são ligados pelos objetos comuns dentro de determinada categoria. O ato de colecionar surge para suprir necessidades pessoais do colecionador e também sua vontade de pertencimento em que a cooperação, comparação e competição na posse dos objetos é inerente aos grupos. Nesse contexto ocorrem as fases de manipular, catalogar e expor os itens colecionáveis onde:

> “Catalogar proporciona ao colecionador um meio tangível de controlar o atingimento dos objetivos de coleção. Manipular os objetos conforme o desejado pelo colecionador garante uma unicidade à coleção. E a exposição dos objetos assume um caráter ritualístico, já que o colecionador busca criar um contexto apropriado (espaço, iluminação, organização) para dispor os objetos.” (HADDAD, 2018, p.36)

É relevante estudar esse tema pois os colecionadores podem se aproveitar dos avanços tecnológicos para facilitar a gestão de sua coleção. Através de um celular é possível catalogar, manipular a qualquer momento pela internet e expor virtualmente a coleção para outros interessados, permitindo a conexão entre colecionadores. Também pode incentivar a disputa e facilitar o encontro de itens entre caçadores/colecionadores.

1.1 Objetivos

1.1.1 Objetivo Geral

- Desenvolver um aplicativo para melhorar a gestão *online* de coleções usando os recursos disponíveis em um celular.

1.1.2 Objetivos Específicos

- Estudar sobre colecionismo e suas práticas;
- Estudar as tecnologias necessárias para criação do aplicativo;
- Definir requisitos para desenvolvimento do site e aplicativo;
- Implementar aplicação *mobile* com *Flutter*.

1.2 Metodologia

Esta pesquisa segundo sua natureza é um resumo de assunto, buscando compreender a área de conhecimento com base em sua evolução histórica, levando em conta suas aplicações e o que foi desenvolvido até hoje (WAZLAWICK, 2014).

Segundo seus objetivos é uma pesquisa descritiva. Essa pesquisa é muitas vezes considerada como parte dos primeiros passos de uma pesquisa mais aprofundada, onde busca-se obter dados mais consistentes sobre determinada realidade (WAZLAWICK, 2014).

Segundo seus procedimentos técnicos esta pesquisa é bibliográfica e experimental. A primeira é um passo importante e prévio da pesquisa, é realizada com estudo de materiais como artigos, teses e livros disponíveis em repositórios universitários ou por editoras e afins para embasamento teórico (WAZLAWICK, 2014). A pesquisa também contou com a consulta as documentações de referidas linguagens disponibilizadas online. A pesquisa experimental caracteriza-se pela manipulação de um aspecto da realidade pelo pesquisador (WAZLAWICK, 2014).

Para Gil (2017) a pesquisa bibliográfica deve conter as seguintes etapas:

1. Escolha do tema: desenvolver um aplicativo para gestão e exposição de coleções;

2. Levantamento bibliográfico preliminar: um estudo preliminar ajuda o pesquisador a se familiarizar com o tema proposto;
3. Formulação do problema: é possível desenvolver um aplicativo para melhorar a gestão e exposição online de coleções usando os recursos disponíveis em um celular?;
4. Busca das fontes: artigos, teses e dissertações disponíveis virtualmente em repositórios universitários ou editoras, livros, periódicos científicos e afins. A pesquisa foi realizada no Google Acadêmico, Repositório Institucional da UnB (RIUnB), Repositório Acadêmico de Graduação da PUC Goiás (RAG) e Periódicos das CAPES;
5. Leitura do material: relacionar os estudos preliminares e as fontes ao tema escolhido;
6. Fichamento: realizar o fichamento dos materiais que possam se relacionar ao tema para ajudar no desenvolvimento;
7. Redação do texto: escrita do Trabalho de Conclusão de Curso (TCC);

A pesquisa experimental deve conter as seguintes etapas, segundo Gil (2027):

1. Formulação do problema: é possível desenvolver um aplicativo para melhorar a gestão e exposição online de coleções usando os recursos disponíveis em um celular?;
2. Definição do plano experimental:
    2.1. Levantamento de Requisitos;
    2.2. Análise de requisitos;
    2.3. Prototipação;
    2.4. Preparação do ambiente:
        a. Instalação dos softwares necessários para desenvolvimento;
        b. Construção do banco de dados segundo modelo;
    2.5. Criação de um protótipo.

3. Determinação dos sujeitos: colecionadores;
4. Determinação do ambiente:
    4.1. celular:
        a. Marca: Samsung;
        b. Modelo: Galaxy S23;
        c. Versão do Android: 14;
        d. Versão do One UI (Interface): 6.1;
        e. Memória Ram: 8,00 GB;
        f. CPU: Qualcomm Snapdragon 8 Gen 2 (octa-core de 3,36 GHz);
        g. Armazenamento Interno: 256,00 GB.
5. Coleta de dados: Serão realizados testes para coleta de dados e futuras análises;
6. Análise e interpretação: Os resultados obtidos dos testes serão minuciosamente compilados e analisados com base nos dados adquiridos com os estudos realizados nesse projeto para chegar a uma compreensão melhor dos resultados;
7. Redação do relatório: escrita do TCC.

1.3 Resultados Obtidos

Espera-se que os resultados desse trabalho possam auxiliar colecionadores a:

- Garantir facilidade no cadastro dos itens colecionados;
- Garantir controle e acesso via internet ao aplicativo;
- Contar a quantidade de itens na coleção;
- Usar de ferramentas presentes no celular como câmera e acesso a internet para melhorar o cadastro dos itens.

2 Referencial Teórico

2.1 *Flutter* e GetX

O Flutter é um *framework* de desenvolvimento multiplataforma criado pelo Google, que permite a criação de aplicações para Android, iOS, web e *desktop* a partir de um único código-base. Ele utiliza a linguagem de programação Dart e oferece uma experiência nativa, graças ao uso de um motor gráfico chamado Skia. A principal vantagem do Flutter está em seu sistema de *widgets*, que permite construir interfaces personalizadas e responsivas de maneira eficiente. Além disso, ferramentas como *Hot Reload* agilizam o ciclo de desenvolvimento, permitindo visualizar mudanças no código em tempo real. Segundo a documentação oficial do Flutter, ele foi projetado para garantir alta performance e flexibilidade: "Flutter transforma o processo de desenvolvimento de aplicativos. Crie, teste e implante aplicativos móveis, web, desktop e embutidos atraentes a partir de uma única base de código." (FLUTTER BRASIL, 2024).

O GetX é um *framework* complementar para o Flutter que oferece uma abordagem simplificada para gerenciamento de estado, dependências e rotas. Ele é baseado na reatividade e elimina a necessidade de configurações complexas, permitindo maior agilidade no desenvolvimento. A documentação oficial destaca: "Get é uma biblioteca poderosa e extraleve para Flutter. Ela combina um gerenciador de estado de alta performance, injeção de dependência inteligente e gerenciamento de rotas de uma forma rápida e prática." (JONATASLAW, 2021).

O GetX possui três pilares:

1. **Gerenciamento de Estado:** Ao trabalhar com Flutter, as páginas podem possuir estados (*Stateful*) ou não (*Stateless*). Quando uma página *Stateful* sobre alteração em seu estado ela é inteira renderizada de novo para atender ao novo estado, no getx isso não acontece. Ao usar variáveis observáveis é possível incluir os widgets onde essas variáveis serão usadas para que apenas eles sejam renderizados quando houver mudança de estado, o que facilita o controle dos estados.

2. **Gerenciamento de Rotas:** Com o get é possível transitar entre as diversas páginas e objetos, como *snackbars* e *dialogs*, do Flutter de forma prática e rápida. Criar um arquivo de rotas garante melhor controle sobre as rotas e clareza no código ao trabalhar com rotas nomeadas. Caso queira ir a uma página sem tirá-la da hierarquia você pode usar o comando *Get.toNamed(Routes.nextPage)* sendo *Get* a biblioteca e *Routes* a classe do arquivo com as rotas nomeadas, *nextPage* é a variável que contem a rota nomeada. Para voltar basta usar o *Get.back().* Se ao mudar de página deseja-se tirar a anterior da hierárquia usa-se *Get.offNamed(Routes.nextPage),* o *Get.back()* nesse caso, retornaria para a página padrão ou inicial definida. Isso retira a necessidade do uso do *context* (um objeto que fornece informações sobre a localização do *widget* atual na árvore de *widgets)* e facilita a transição entre páginas e objetos, agilizando o desenvolvimento.
3. **Gerenciamento de Dependência:** O *framework* permite que uma dependência seja instanciada de maneira prática e eficiente com uma única linha de comando. Ainda é possível instanciar uma dependência apenas quando ela for necessária ou quando for chamada. Por padrão as instâncias são descartadas quando não são mais usadas, mas ainda é possível que sejam mantidas caso seja informado, assim é possível usar uma instancia em todo o aplicativo.

2.2 Laravel

O Laravel é um *framework* PHP que adota o padrão de arquitetura *Model-View-Controller* (MVC), sendo reconhecido por sua elegância e simplicidade. Ele facilita o desenvolvimento de aplicações web modernas ao oferecer ferramentas integradas para tarefas comuns, como autenticação, manipulação de banco de dados e gerenciamento de rotas. Além disso, o Laravel utiliza o *Eloquent ORM*, que simplifica o mapeamento objeto-relacional, e o sistema de templates Blade, que permite a criação de interfaces dinâmicas e organizadas. Segundo a documentação oficial, "Laravel é um *framework* de aplicação web com sintaxe expressiva e

elegante. Acreditamos que o desenvolvimento deve ser uma experiência agradável e criativa para ser verdadeiramente gratificante" (LARAVEL, 2023, tradução própria[1]).

Com o suporte do PHP 8.1.2, o Laravel aproveita recursos modernos como tipagem aprimorada, atributos e corrotinas, proporcionando maior segurança e desempenho nas aplicações.

2.3 PostgreSQL

O PostgreSQL é um sistema de gerenciamento de banco de dados relacional e *open-source*, amplamente reconhecido por sua confiabilidade e desempenho. Ele oferece suporte a operações complexas, extensões personalizadas e integrações com diversas linguagens de programação. Segundo a documentação oficial, o PostgreSQL é projetado para garantir alta conformidade com padrões SQL e flexibilidade: "PostgreSQL é um poderoso sistema de banco de dados relacional de objeto de código aberto com mais de 35 anos de desenvolvimento ativo que lhe rendeu uma forte reputação de confiabilidade, robustez de recursos e desempenho" (POSTGRESQL, 2024, tradução própria[2]).

2.4 Arquitetura MVC

A arquitetura Model-View-Controller (MVC) é uma abordagem estruturada para o desenvolvimento de software, que separa a aplicação em três componentes principais:

- *Model*: Lida com os dados e a lógica de negócios.
- *View*: Responsável pela interface com o usuário.
- *Controller*: Faz a intermediação entre o modelo e a visão.

---

1 **Laravel is a web application framework with expressive, elegant syntax. We believe development must be an enjoyable and creative experience to be truly fulfilling.**

2 **PostgreSQL is a powerful, open source object-relational database system with over 35 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance.**

O uso dessa arquitetura se dá pela forma como a documentação do laravel apresenta sua estrutura de pastas, segundo TATTERSALL (2015), "Em PHP, o padrão MVC é particularmente benéfico porque organiza o código, facilitando sua manutenção e escalabilidade"[3].

3 **In PHP, the MVC pattern is particularly beneficial because it organizes the code, making it easier to maintain and scale.**

3 Implementação

3.1 Arquitetura/Projeto

O projeto foi planejado usando da arquitetura de camadas onde cada camada tem uma responsabilidade clara e definida e só se comunica com as camadas imediatamente adjacentes a ela, todas as outras estão ocultas. Aplicações que seguem esse padrão “geralmente apresentam variações de quatro camadas lógicas: 1) apresentação; 2) negócios; 3) persistência; e 4) banco de dados. Algumas vezes, as camadas de persistência e banco de dados são “mescladas” e ampliadas para uma descrição mais genérica, como ‘infraestrutura’” (JUNIOR, 2021).

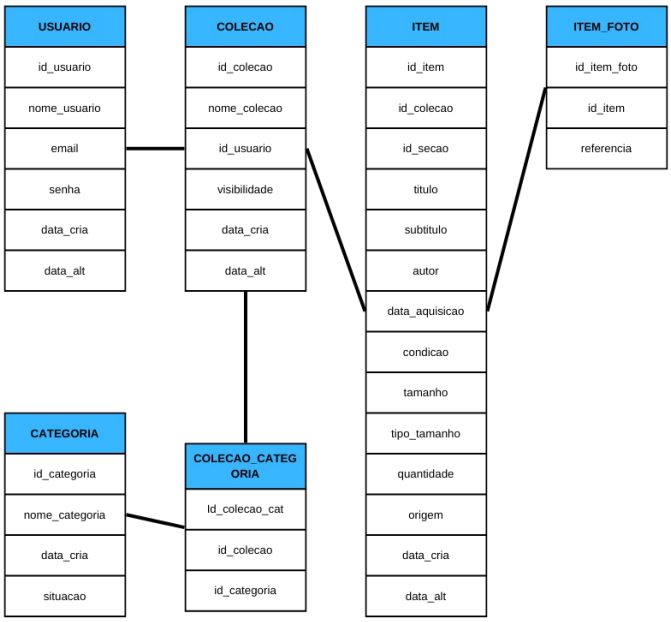
A camada de apresentação do projeto é feita o *framework* flutter com *framework* de desenvolvimento Getx. Assim criando um aplicativo leve e atual com bom controle de estado, possibilidade de escalabilidade e de ser facilmente aplicado em mais de uma plataforma, visto que o Flutter permite que você desenvolva em Dart e gere aplicações para as mais diversas plataformas como Windows, Linux, navegadores de computador, *android* e IOS. O foco é a criação de um aplicativo de celular *android* mas pode ser expandido para um site, por exemplo. O foco está na possibilidade de disponibilizar isso para qualquer colecionador com um celular e acesso a internet.

A camada de negócios, apesar do nome, não possui muitas regras de negócio por conta do tipo de projeto mas apresenta algumas regras e consistências que devem ser levadas em consideração para garantir o melhor uso do sistema e dados mais confiáveis. Nessa camada os dados são manipulados para serem persistidos no banco de dados e é essa camada que faz a ligação entre essas outras camadas, a de apresentação citada anteriormente e a de banco de dados. Essa camada foi desenvolvida em PHP com a *framework* de linguagem laravel usando o padrão arquitetural de MVC o qual o laravel se beneficia muito. Com ela foi criado uma api para comunicação entre o aplicativo e banco de dados a qual será tratada ainda nesse trabalho.

Para verificar o funcionamento da API e suas rotas foi amplamente utilizado para o sistema *Postman* para testes antes de aplicar a API no projeto. Segundo o site da empresa o "*Postman* é sua plataforma única para desenvolvimento colaborativo de API.[...] Prototipe, documente, teste e demonstre todas as suas APIs em um só lugar. Obtenha *feedback* antecipado conversando no contexto de qualquer API (privada, pública ou de parceiro) e não espalhada por ferramentas." (POSTMAN, 2024) (tradução própria).

A última camada contem apenas o banco de dados Postgres para persistência dos dados. Para consultar os dados foi usado o software pgAdmin. Uma plataforma de administração e desenvolvimento de código aberto para manter o banco de dados e permitir sua manipulação. Mesmo assim as tabelas foram criadas com uso da Nesse caso foram aproveitadas algumas funções do Laravel para atuar junto ao banco de dados, porém a arquitetura foi mantida, e nenhuma regra de negócio foi aplicada nele. Para criação de tabelas e suas relações e consistências criamos os objetos de banco de dados fazendo uso das *Migrations*, um conjunto de instruções que se usa para fazer alterações no esquema do banco de dados. Com ela podemos criar um arquivo com as definições de tabelas ou mesmo alterações em tabelas que ao executar é aplicado diretamente ao banco de dados. Com ela foi possível implementar o modelo lógico, conforme Figura 1, sugerido ao banco de dados sem o acesso direto a ele.

Figura 1: Modelagem Lógica



Fonte: elaborado pelo autor

3.2 Implementação

3.2.1 O Usuário

Ao iniciar o aplicativo é possível verificar a *splash screen* (Figura 2), uma tela inicial que é gerada enquanto o aplicativo está inicializando. Para criá-la foi instalada a dependência *flutter_native_splash* que facilita a configuração da *splash screen*. Seguindo a documentação foi possível configurá-la para apresentar a cor que determinamos e a imagem da Logo do aplicativo.

Figura 2: Splash screen criada com flutter_native_splash

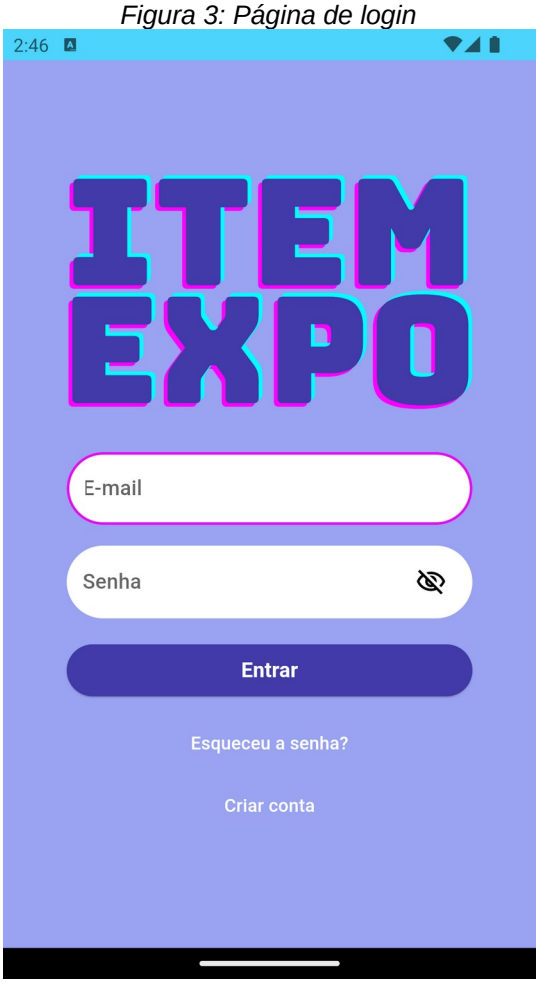
Fonte: elaborado pelo autor

A tela de login foi implementada com o propósito de testar a autenticação e a implementação de rotas autenticadas no projeto. Para proteger a API foi usado o

JWT (*Json Web Token*) para gerar uma chave secreta a ser indicada no *enviroment* (o arquivo .*env* do laravel) e permitir que a API fique segura. Dessa forma, ao criar uma rota indicando o *middleware* como da API ele deverá ser autenticado para poder ser usado, caso contrário a rota retornará erro.

A rota de login retorna um *token* que é armazenado no celular e usado no cabeçalho de todas as chamadas http para a API. Ainda na tela inicial (Figura 3) é possível acessar duas páginas sem autenticar, a de recuperação de senha ao clicar em ‘Esqueceu a senha?’ (Figura 4) e a de criação de conta. Na primeira vez acessando o aplicativo o usuário deverá se cadastrar na tela de registro (Figura 5) acessada clicando em criar conta. Com alguns dados básicos e uma senha que será encriptada na API através do uso de *hash* o usuário terá acesso ao aplicativo rapidamente.

Figura 3: Página de login

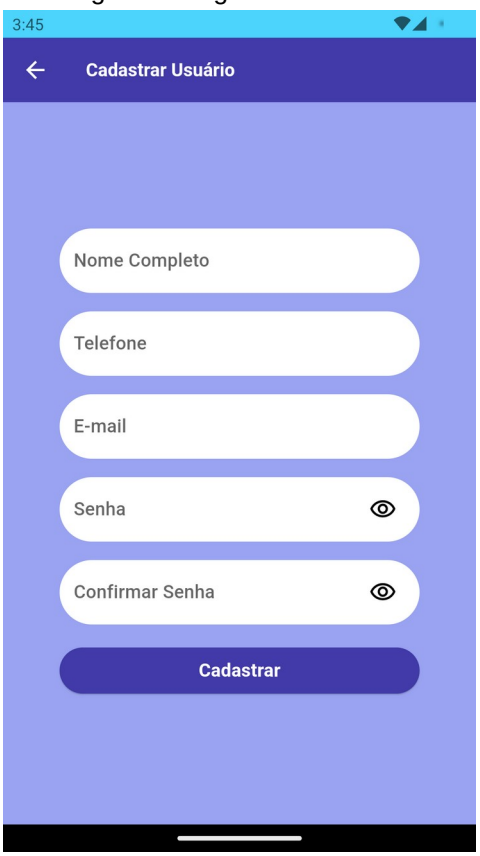


Fonte: elaborado pelo autor

Figura 4: Página de recuperação de senha

Fonte: elaborado pelo autor

Figura 5: Página de cadastro



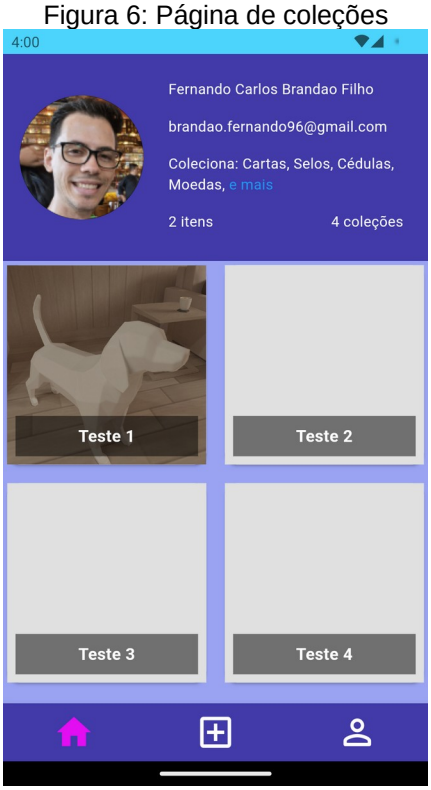
Fonte: elaborado pelo autor

3.2.2 As Coleções

A página inicial é a tela das coleções, aqui o usuário poderá ver e acompanhar as suas coleções além de alguns dados básicos sobre seu tipo de colecionador. A tela apresenta uma barra de navegação inferior que fica presente nas três telas que pela qual ela navega, são elas: a tela de coleções (Figura 6), o cadastro de coleções (Figura 7) e os dados do usuário (Figura 8).

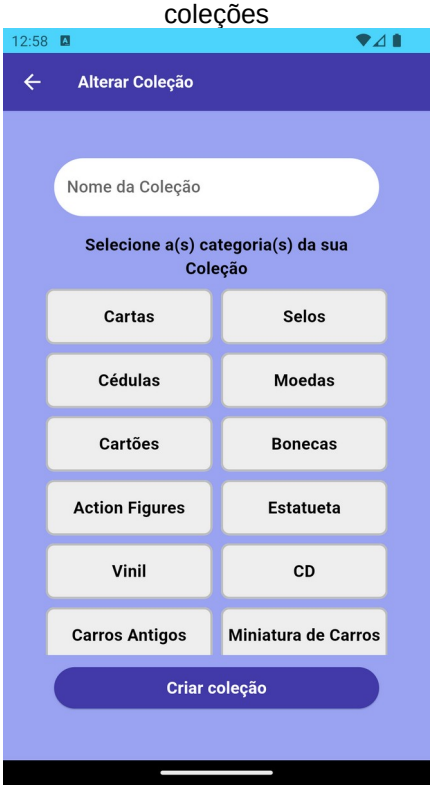
Os cards de coleção sempre apresentarão o nome da coleção e a imagem do primeiro item da coleção. Para que uma coleção apareça na tela inicial é preciso clicar no botão centro inferior na barra de navegação. A coleção pede apenas nome e categoria para criação. Você pode criar várias coleções com diferentes categorias ou mesmo uma só para poder separar as suas coleções segundo suas categorias. Quando uma coleção é cadastrada ela aparecerá na tela de coleções sem imagem, apenas o título, a imagem aparecerá quando um item for cadastrado.

Figura 6: Página de coleções



Fonte: elaborado pelo autor

Figura 7: Página de cadastro de coleções



Fonte: elaborado pelo autor

Figura 8: Página de alteração de dados de usuário

Fonte: elaborado pelo autor

A tela de dados do usuário te permitir alterar seus dados, até mesmo a senha mas apara isso é preciso conhecer sua senha antiga. Ainda é possível realizar o logout do aplicativo e voltar para a tela de login. A tela de alteração de senha (Figura 9) é muito parecida com a tela de cadastro a diferença que é necessário informar a senha atual para alterá-la na tela.

Figura 9: Página de alteração de senha

Fonte: elaborado pelo autor

Para acessar a tela de Itens ou cadastrar um novo item você deve segurar o card da coleção a qual você quer ver os itens ou cadastrar um novo. Não é possível cadastrar itens sem ter uma coleção inicial, como mostrou o modelo entidade-relacionamento que as tabelas se comunicam através de Id.

3.2.3 Os Itens

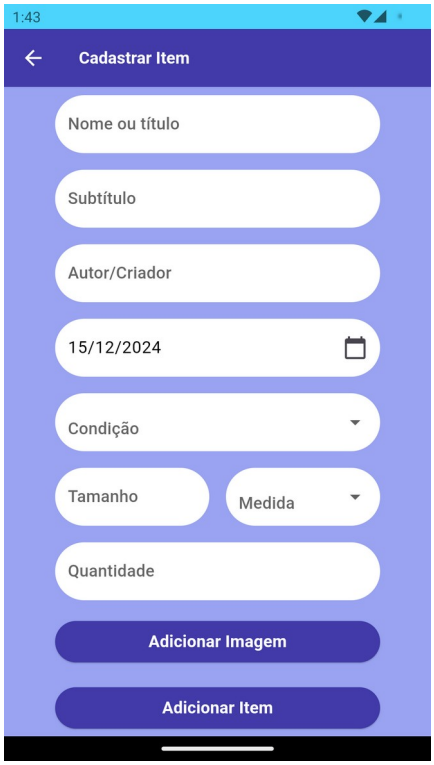
A tela de itens (Figura 10) possui um botão flutuante para poder fazer o cadastro de novos itens (Figura 11). Apesar de muitos dados, apenas nome, data de aquisição e a imagem são obrigatórios, os outros dados servem para enriquecer a descrição caso o colecionador deseje.

A opção de adicionar foto abrirá um modal para que o usuário escolha se ele deseja abrir a câmera para tirar uma do item ou a galeria para enviar a imagem do mesmo. Após o cadastro o item ficará visível na tela de coleções e você pode alterar suas informações clicando e segurando em cima do item, como na coleção.

Figura 10: Página de itens

Fonte: elaborado pelo autor

Figura 11: Página de cadastro de itens



Fonte: elaborado pelo autor

4 Considerações Finais

Este trabalho apresentou uma aplicação para colecionadores para mostrar as possibilidades que um aplicativo de celular tem para auxiliar na tarefa de gerir uma coleção. A proposta é uma ferramenta que tenta trazer um ar de novidade ao mercado não muito tocado dos colecionadores além de demonstrar uma boa aplicação das *frameworks Flutter*, getx e *laravel*, leve e rápida também.

Apesar disso ainda há melhorias que pode ser feitos para melhorar a experiência do colecionador e o processo de gestão. Uma página web onde você possa fazer todo o processo e ainda criar uma página de exposição que pode ser acessada por outros usuários é uma ideia que poderia ser implementada futuramente. A possibilidade de gerar uma planilha com as coleções e itens pode ajudar os colecionadores a ter outra forma de visualizar e manter seus dados. Ainda uma página de gestão para cadastro de categorias e controle de usuários pelo responsável pelo aplicativo poderia melhorar o atendimento de demandas de melhorias e/ou alterações feitas pelos usuários.

Este projeto foi uma oportunidade valiosa para aprofundar conhecimentos em desenvolvimento de software e representa uma base sólida para estudos futuros. O projeto do aplicativo e a API dele estão presentes nos seguintes *github*: https://github.com/carbran/app_item_expo.git para o aplicativo em flutter e https://github.com/carbran/api_item_expo.git para a api laravel.

REFERÊNCIAS

CIRNE, M. D. S. **Memorabilia: Uma coleção feita de objetos, escritos e memórias.** 2018. 119f. Dissertação (Mestrado em Artes Visuais) - Programa de Pós-Graduação em Artes Visuais, Centro de Artes, Universidade Federal de Pelotas, Pelotas, 2018.

FLUTTER BRASIL. **Flutter Brasil**. Disponível em: https://flutterbrasil.dev. Acesso em: 20 nov. 2024.

GIL, Antônio Carlos. **Como Elaborar Projetos de Pesquisa**. 6. ed. São Paulo: Editora Atlas Ltda., 2017.

HADDAD, Helder. **Collectible toys - fatores influenciadores e resultantes do colecionismo: um estudo com colecionadores de estátuas e figuras de ação brasileiros e norte-americanos**. Escola Superior De Propaganda e Marketing. 2018. Disponível em: https://sucupira.capes.gov.br/sucupira/public/consultas/coleta/trabalhoConclusao/viewTrabalhoConclusao.jsf?popup=true&id_trabalho=6397809. Acesso em: 12 abr. 2023.

JONATASLAW. **GetX**. GitHub, 2021. Disponível em: https://github.com/jonataslaw/getx/blob/master/README.pt-br.md. Acesso em: 20 nov. 2024.

JUNIOR, Elemar. ***Projetando software em camadas*. Arquitetura de Software**. Disponível em: https://arquiteturadesoftware.online/volume-1/projetando-software-em-camadas/. Acesso em: 01 dez. 2024.

JWT AUTH. **JSON Web Token Authentication for Laravel & Lumen**. Disponível em: https://jwt-auth.readthedocs.io/en/develop/. Acesso em: 16 nov. 2024.

LARAVEL. **Laravel**. GitHub, 2024. Disponível em: https://github.com/laravel/laravel. Acesso em: 20 nov. 2024.

OHASHI, Orlando. ***App Flutter: utilizando a câmera e/ou acessando a galeria***. Medium, 26 out. 2018. Disponível em: https://medium.com/@orlandoohashi/app-flutter-utilizando-a-camera-e-ou-acessando-a-galeria-647ad6237d77. Acesso em: 01 dez. 2024.

OLIVEIRA, Célia. **Coleções e colecionadores: as práticas de colecionar, motivações e simbologias. Museologia & Interdisciplinaridade**, [S. l.], v. 6, n. 12, 2017. DOI: 10.26512/museologia.v6i12.16356. Disponível em: https://periodicos.unb.br/index.php/museologia/article/view/16356. Acesso em: 10 abr. 2023.

POSTGRESQL. **Documentation**. Disponível em: https://www.postgresql.org/docs. Acesso em: 01 dez. 2024.

POSTMAN. **Postman: API Platform for API Development**. Disponível em: https://www.postman.com/. Acesso em: 01 dez. 2024.

TATTERSALL, Matt. **The MVC Pattern and PHP – Part 1**. SitePoint, 2015. Disponível em: https://www.sitepoint.com/the-mvc-pattern-and-php-1/. Acesso em: 20 nov. 2024.

WAZLAWICK, R. S. **Metodologia da Pesquisa para Ciência da Computação**. 2ª ed. [S.l.]: Campus, 2014.