REPÚBLICA FEDERATIVA DO BRASIL

MINISTÉRIO DA ECONOMIA

INSTITUTO NACIONAL DA PROPRIEDADE INDUSTRIAL

DIRETORIA DE PATENTES, PROGRAMAS DE COMPUTADOR E TOPOGRAFIAS DE CIRCUITOS INTEGRADOS

Certificado de Registro de Programa de Computador

Processo Nº: **BR512022001665-6**

O Instituto Nacional da Propriedade Industrial expede o presente certificado de registro de programa de computador, válido por 50 anos a partir de 1° de janeiro subsequente à data de 31/03/2022, em conformidade com o §2°, art. 2° da Lei 9.609, de 19 de Fevereiro de 1998.

Título: Pra Nadar App

Data de publicação: 31/03/2022

Data de criação: 30/03/2022

Titular(es): FUNDAÇÃO UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE - FUERN

Autor(es): ADALBERTO VERONESE DA COSTA; SEBASTIÃO EMIDIO ALVES FILHO; EXLLEY CLEMENTE DOS SANTOS; TIAGO DA SILVA MORAIS; JEFTER ROBERTO MOTA TARGINO; FRANCISCO CLEMENTINO MAIA JÚNIOR; MATHEUS DIOGENES DA SILVA; HÉLIO VICTOR APOLINÁRIO SOARES

Linguagem: JAVA SCRIPT; OUTROS

Campo de aplicação: AD-01; IF-01; SD-01

Tipo de programa: AP-01; GI-01; SO-05

Algoritmo hash: SHA-512

Resumo digital hash:
2c2b1f799dcd7d1f60ca267144f12e53ad34b6dfe293f543e8ca31912a76fab534c8d17ffb79ac75dbebd2e18d55a74d28d5fa02a96742c7dcbd9fe5a18fb039

Expedido em: 12/07/2022

Aprovado por:
Joelson Gomes Pequeno
Chefe Substituto da DIPTO - PORTARIA/INPI/DIRPA Nº 02, DE 10 DE FEVEREIRO DE 2021

UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE – UERN
FACULDADE DE CIÊNCIAS EXATAS E NATURAIS – FANAT
DEPARTAMENTO DE INFORMÁTICA – DI

FRANCISCO CLEMENTINO MAIA JÚNIOR

PRA NADAR: UMA APLICAÇÃO FRONT-END MOBILE PARA GERENCIAMENTO DO PROGRAMA PRA NADAR

MOSSORÓ - RN
2022

FRANCISCO CLEMENTINO MAIA JÚNIOR

PRA NADAR: UMA APLICAÇÃO FRONT-END MOBILE PARA GERENCIAMENTO DO PROGRAMA PRA NADAR

Relatório apresentado ao curso de Ciência da Computação da Universidade do Estado do Rio Grande no Norte como requisito da disciplina de Trabalho de Diplomação, sob a orientação do(a) Prof. Dr. Sebastião Emidio Alves Filho e coorientação do Me. Exlley Clemente dos Santos.

MOSSORÓ - RN

2022

FRANCISCO CLEMENTINO MAIA JÚNIOR

PRA NADAR: UMA APLICAÇÃO FRONT-END MOBILE PARA GERENCIAMENTO DO PROGRAMA PRA NADAR

Registro de software apresentado como pré-requisito para a obtenção do título de Bacharel em Ciência da Computação da Universidade do Estado do Rio Grande do Norte – UERN, submetida à aprovação da banca examinadora composta pelos seguintes membros:

Aprovada em: 19/04/2022

Banca Examinadora

Prof. Dr. Sebastião Emídio Alves Filho
Universidade do Estado do Rio Grande do Norte - UERN

Me. Exlley Clemente dos Santos
Universidade do Estado do Rio Grande do Norte - UERN

Prof. Dr. Maximiliano Araújo da Silva Lopes
Universidade do Estado do Rio Grande do Norte - UERN

Prof. Dr. Adalberto Veronesse da Costa
Universidade do Estado do Rio Grande do Norte - UERN

SUMÁRIO

1 INTRODUÇÃO

O Programa Pra Nadar é um projeto de extensão fundado e dirigido pela Faculdade de Educação Física (FAEF) da Universidade do Estado do Rio Grande do Norte (UERN), oferecendo práticas de atividades aquáticas, como natação e hidroginástica. As práticas são oferecidas para a comunidade em geral, o que inclui alunos, professores e servidores da UERN, além da comunidade externa. O processo de matrícula dos praticantes passa pela necessidade da entrega de atestado médico, dando a devida condição para o exercício das atividades. O Pra Nadar funciona através das contribuições financeiras, provenientes da participação dos alunos matriculados no programa.

O programa inicialmente era organizado pelos secretários e professores da faculdade e não existia algo para contribuir na gestão dos dados que estavam sendo inseridos e tratados. O processo de gerenciamento de turmas, alunos, instrutores e secretários no programa eram feitos de forma manual, ocasionando na dificuldade do gerenciamento.

Diante da dificuldade do gerenciamento, contando com problemas de controle do número de alunos em cada turma e vagas disponíveis para matrículas, registro das contribuições feitas durante o decorrer do programa, controle das frequências dos alunos e instrutores, além de outros. Com estas adversidades, a FAEF necessitava de um sistema de gerenciamento para ter melhor controle do programa. Foi desenvolvido um sistema que visa administrar os dados do programa de forma rápida, intuitiva e dinâmica.

2 OBJETIVOS

O projeto tem como finalidade auxiliar no gerenciamento do programa Pra Nadar - programa no qual realiza aulas de natação. Então, para poder ter um melhor controle e administração de todos os envolvidos no Pra Nadar (Professores, Alunos, Secretários…), o objetivo deste trabalho é a criação de um sistema que possibilite deixar o controle deste trabalho mais fácil, rápido e eficaz.

Esse projeto tem como foco a criação de um app mobile, permitindo o uso de forma mais simples e prática. Um exemplo dessa praticidade é o instrutor, no caso, o mesmo pode, enquanto acompanha seus alunos na aula, realizar a chamada, observações de um determinado aluno, criar planos de aula, deixando o projeto mais fluído.

…

3 METODOLOGIA

Visando essa automatização, foi implementado uma aplicação *front-end mobile* - sendo utilizada diretamente pelo usuário, com o intuito de facilitar o acesso (de forma móvel) usufruindo de suas funcionalidades e atividades no sistema. Para ter acesso ao sistema, foram criados três tipos de usuários (Administrador, Secretário e Instrutor). Todos com suas interfaces e permissões específicas.

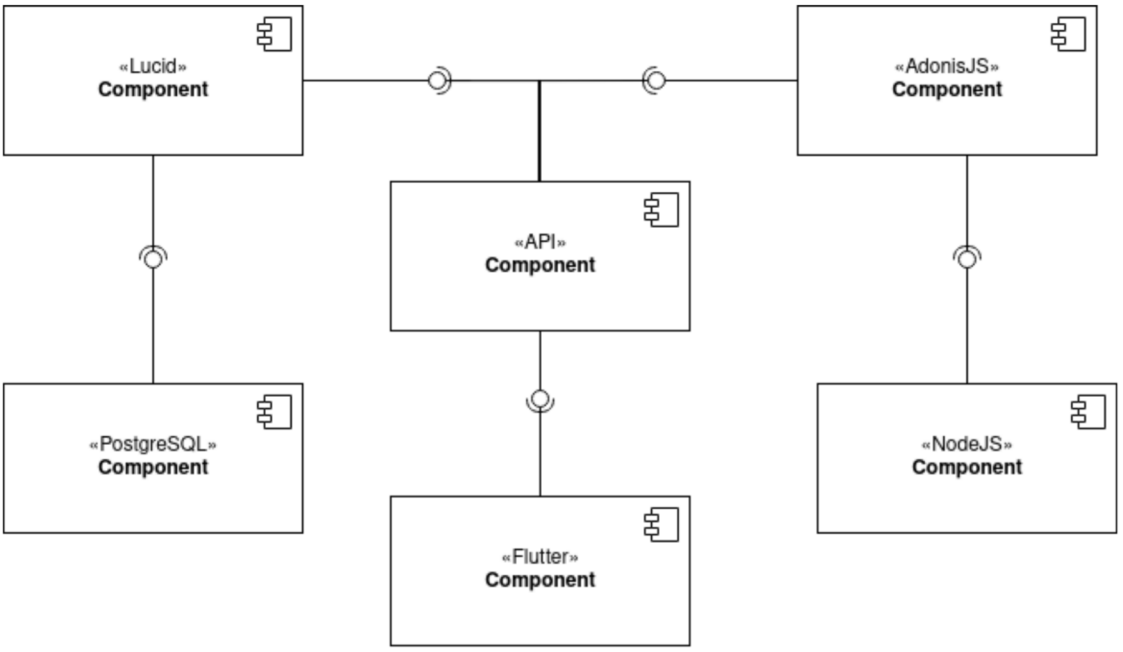
O aplicativo dirige-se principalmente para o usuário do tipo instrutor, antes da aula o instrutor tem a possibilidade de criar seu plano de aula para uma turma específica, também sendo possível realizar a frequência no momento da aula. O administrador, usuário desse tipo de permissão, tem acesso a adição, edição e exclusão de alunos, instrutores, secretários e turmas, podendo também realizar o *reset* de senha para alunos, instrutores e secretários. O secretário possui algumas permissões parecidas com a do administrador, porém o mesmo não poderá realizar quaisquer ações de adição, edição, exclusão e *reset* de senha para usuários do tipo secretário.

Tendo como base a metodologia *Feature Driven Development (FDD)* e no *SCRUM* que de acordo com Jeff Sutherland no livro Scrum: a arte de fazer o dobro do trabalho na metade do tempo (LEYA, 2016), afirma que: “a metodologia consegue mais resultados com menos gente, tempo e recursos, mas com qualidade melhor”. Não segue-se à risca, devido a presença de momentos de dificuldades com algumas funcionalidades - tendo como duração média de duas semanas. E em cada nova reunião, debatia-se se as funcionalidades estão de acordo com o exigido e quais são as próximas a serem entregues.

4 ARQUITETURA DO SISTEMA

Similar a plataforma *Web*, o sistema é composto por três componentes consumindo a API (*Application Programming Interface*) que está conectada com o banco de dados. Para facilitar o entendimento sobre a arquitetura do projeto, a figura abaixo irá representar (Figura 1), como o sistema funciona.

Figura 1 - arquitetura Pra nadar



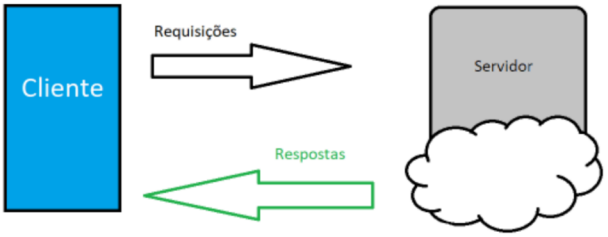
Fonte: Autoria própria (2022)

Faz-se uso do modelo cliente-servidor, no qual o cliente faz requisições para o servidor, não compartilhando seus recursos com o servidor, mas solicitando alguma função do servidor. Sendo ele o cliente, responsável por iniciar a comunicação com o servidor, enquanto o mesmo aguarda requisições de entrada. A exemplificação dessa dinâmica encontra-se na Figura 2.

4.1 INTEGRAÇÃO

Hospedado na *Play Store* - loja de aplicativos oficial do *Google*. É por ela que os usuários *Android* podem fazer *download* de apps de forma gratuita ou paga. Além disso, através dessa plataforma é possível gerenciar a aplicação. Nela existem normas e procedimentos para o lançamento do aplicativo, como por exemplo: conteúdo, classificação - se é livre (L), maior de 18 anos (+18)... -, em quais países o app vai estar disponível para ser baixado, categoria, se o lançamento do app é fechado - para pessoas selecionadas - ou aberto - para todos. Após todas essas observações, a aplicação fica em análise por 7 dias. Por fim, o aplicativo Pra Nadar obteve a classificação Livre (L), com *download* disponível para todos os países, sem conteúdo pago.

Figura 2 - Arquitetura cliente-servidor



Fonte: Autoria própria (2022)

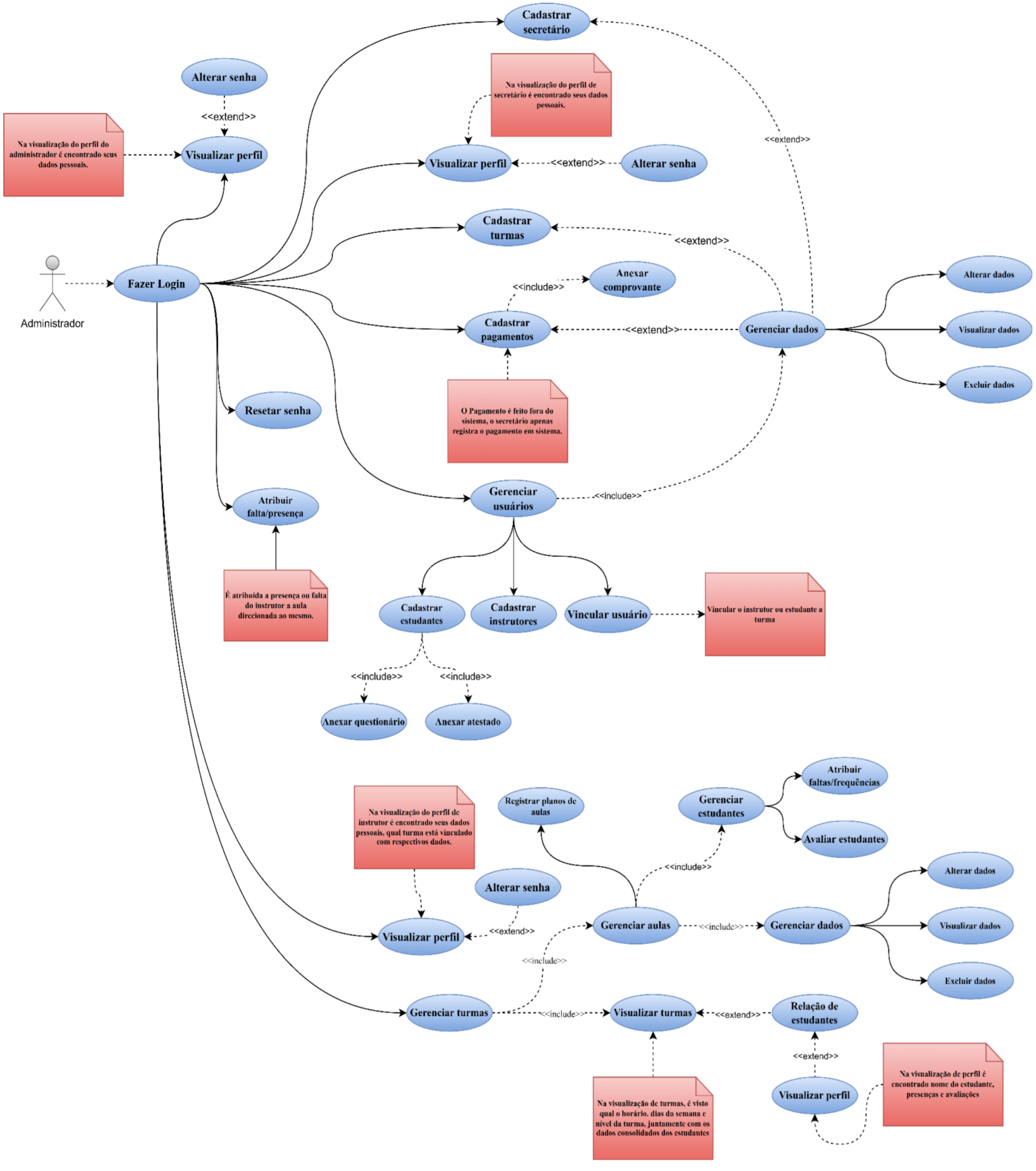
4.2 CASOS DE USO

Em relação aos casos de uso, a versão mobile do projeto possui 3 atores: o administrador, secretário, instrutor; todos com suas hierarquias no sistema e suas respectivas permissões.

O usuário do tipo administrador é o que possui maior privilégio no sistema, como a figura 3, o usuário desse tipo de permissão pode realizar cadastros de alunos, instrutores, secretários e turmas. Como também excluí-los do sistema, editar seus dados, resetar senhas. Pode também vincular alunos a uma determinada turma como também um instrutor a uma determinada turma.

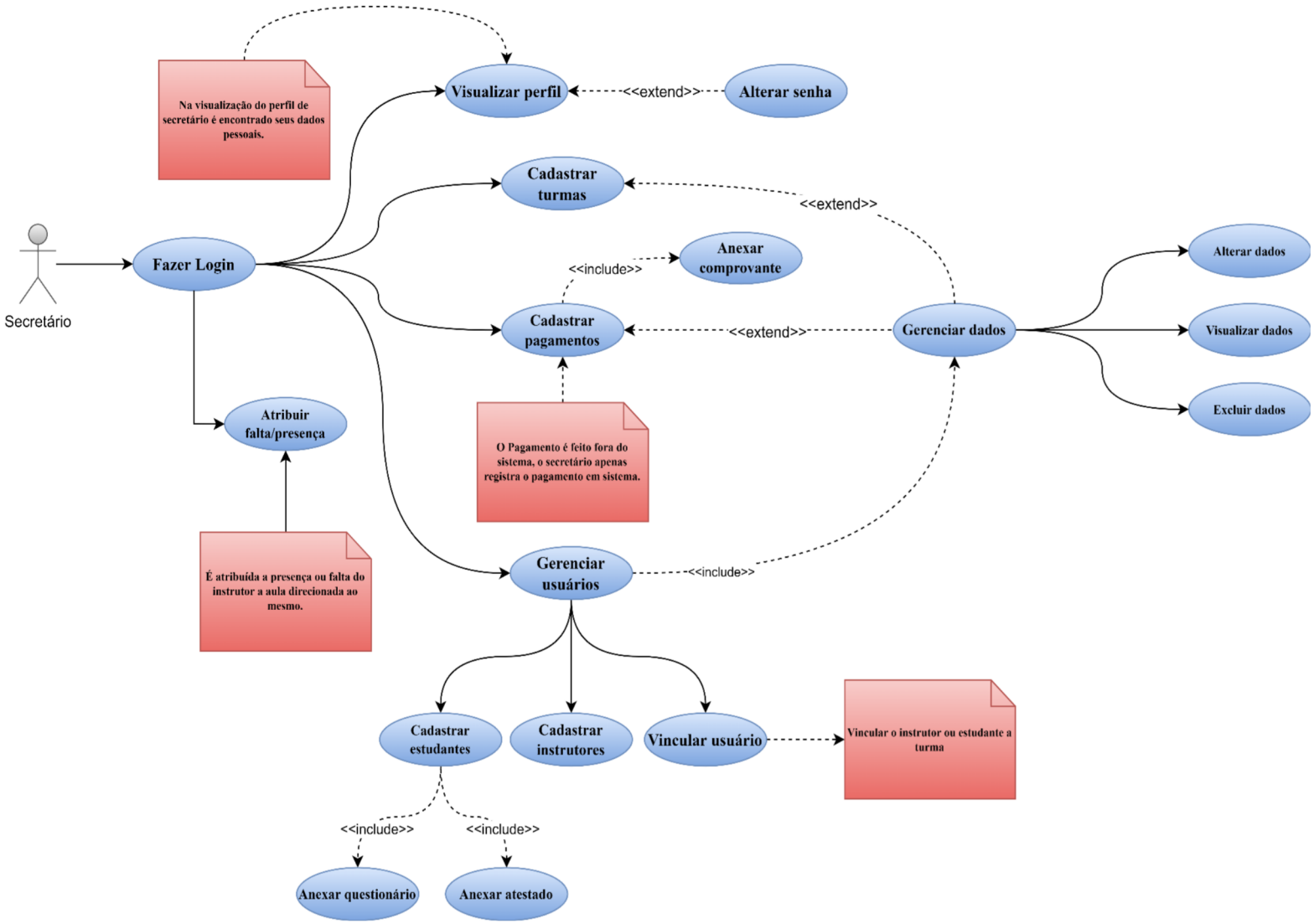
...

Figura 3 - Diagrama de caso de uso administrador



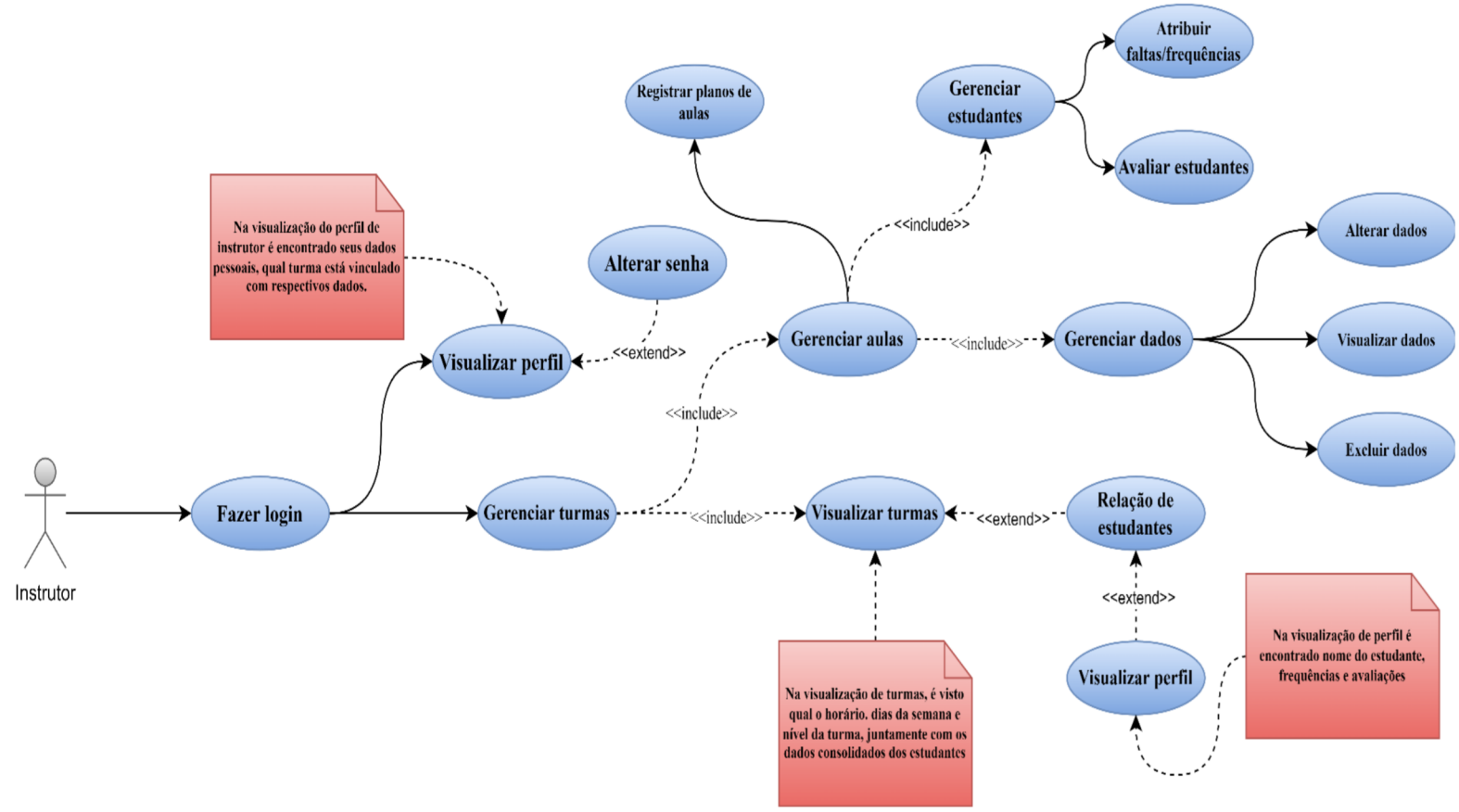
No caso de uso do secretário ele pode realizar cadastros de turmas, alunos e instrutores, excluir seus dados do sistema, resetar senhas, vincular alunos e instrutores a uma determinada turma, porém, o mesmo não pode criar um usuário do tipo secretário, pois quem tem domínio sobre todos eles é o usuário administrador. A figura 4, mostra as suas permissões.

Figura 4 - Diagrama de caso de uso secretário



Por fim, o usuário instrutor em que o projeto mobile tem seu foco principal, pode gerenciar as aulas do programa Pra Nadar, como por exemplo: registrar planos de aulas, a atribuição de faltas(frequências) e observações de determinado estudante, se o mesmo precisa melhorar as braçadas, realizar algum tipo de exercícios para fortalecer determinado músculo, usar equipamentos solicitados para a aula. Também é possível visualizar turmas e relação de alunos das turmas.

Figura 5 - Diagrama de caso de uso instrutor



5 DESCRIÇÃO DO PROJETO

Nesta seção aborda-se de forma mais detalhada o funcionamento do projeto, como: tecnologias utilizadas e estrutura de arquivos.

5.1 TECNOLOGIAS UTILIZADAS

Como se trata do *front-end mobile,* o *framework Flutter* foi a melhor opção para o projeto Pra nadar, sendo de código aberto sobre a BSD *License* e multiplataforma, tendo como linguagem base o *Dart*. Criado em 4 de dezembro de 2018 pela *Google*, o *Flutter* atualmente se torna a ferramenta mais versátil para desenvolvimento *mobile*. Comparado com outros *framework's* para desenvolvimento *mobile* como *React Native*, *Kotlin*, *Java*, o *Flutter* tem uma grande vantagem por ser multiplataforma, como por exemplo, através desse código fonte é possível gerar um apk tanto para *smartphones* IOS e *Android*. Outra vantagem é a possibilidade de criar sistemas de navegação de carros (*Toyota*), *Desktop* (*Canonical - Linux*, *Microsoft*), *Back-end*, ainda conta com a rapidez na hora da compilação, facilidade para criar seu app e conta com o suporte da *Google* (QUEIROZ, 2021).

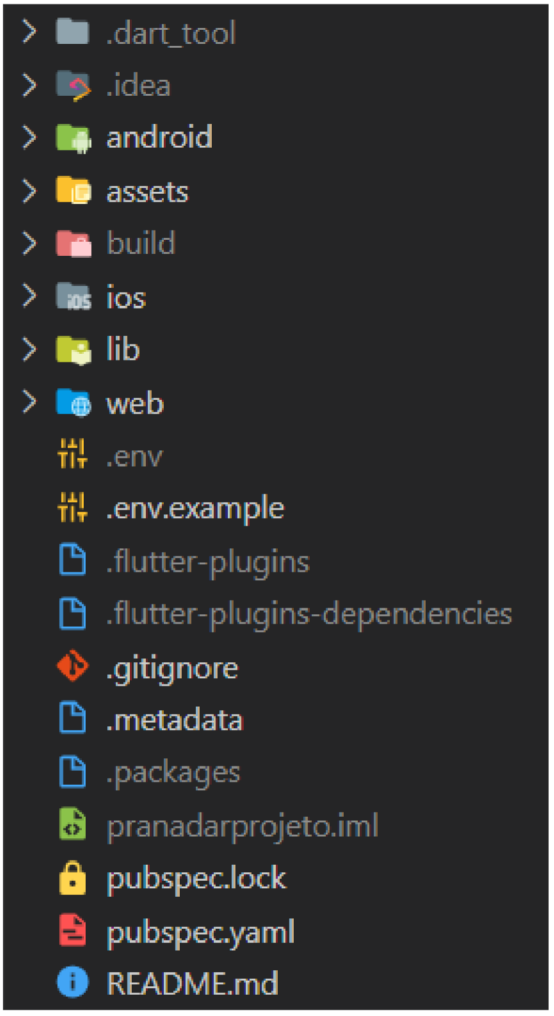
Na parte escrita, o editor de código *Visual Studio Code* juntamente com algumas extensões que facilitaram o desenvolvimento e produção do app (Visual Studio Code, 2015). Na emulação, o *Vysor - software* que espelha o *smartphone* - possibilita utilizar o *smartphone* conectado tanto por usb quanto por rede, para espelhar a aplicação (VYSOR, 2015).

5.2 ESTRUTURA DE ARQUIVOS

O projeto em si inicia-se com sua estrutura de pastas organizadas, porém no decorrer do desenvolvimento, foram implementados novos arquivos e pastas. Nesse projeto o código fonte se localiza na pasta *lib*. A estrutura do projeto é demonstrada com as imagens a seguir.

Estrutura geral: A imagem abaixo apresenta a organização padrão do projeto em *Flutter*, foi adicionado 1 (um) diretório - *assets* e de dois arquivos, *.env* e *.env.example* . Logo à frente é abordado os demais (Figura 3).

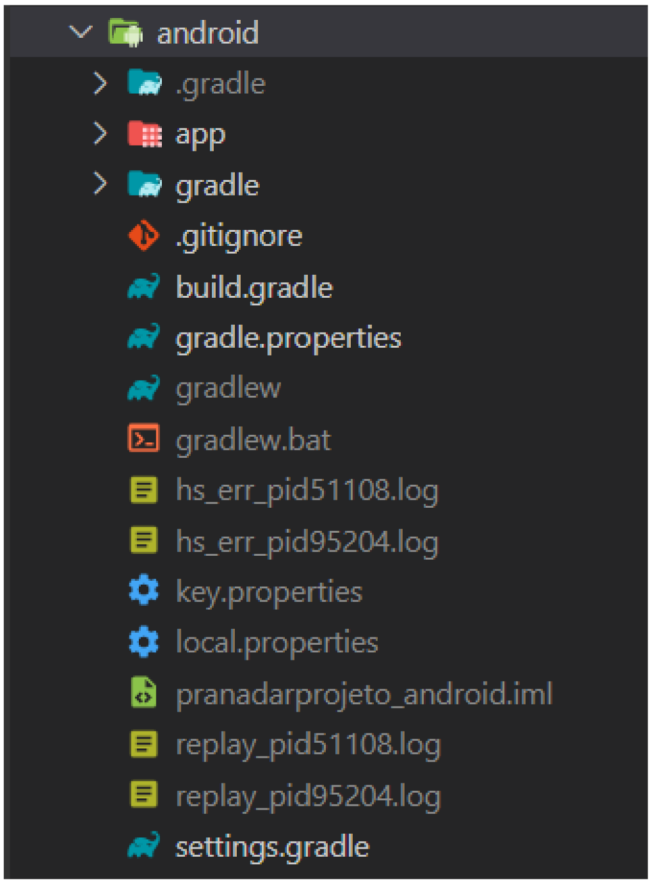
Figura 6 - estrutura geral do nosso projeto.



Fonte: Autoria própria (2022)

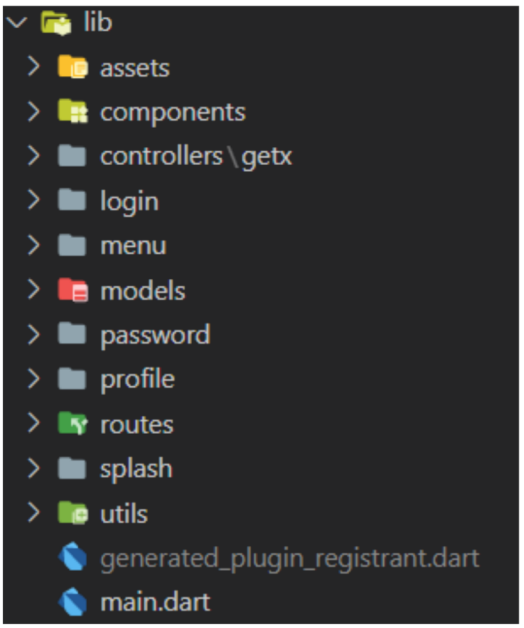
- **android:** Diretório na qual se encontra os arquivos de configuração do app para *android*, nele é permitido a alteração de imagem do app, nome, permissões e autenticidade - assinatura do app (Figura 4).
- **lib:** Nela encontram-se os diretórios e arquivos que buildam o projeto (Figura 5). Como mostrado, a pasta *lib* é a principal do projeto, nela é possível ver novamente a pasta *assets*, porém com uma função diferente.
- **assets:** Diretório onde é armazenado as imagens e ícones utilizados no projeto (Figura 6).

Figura 7 - estrutura da nossa pasta android.



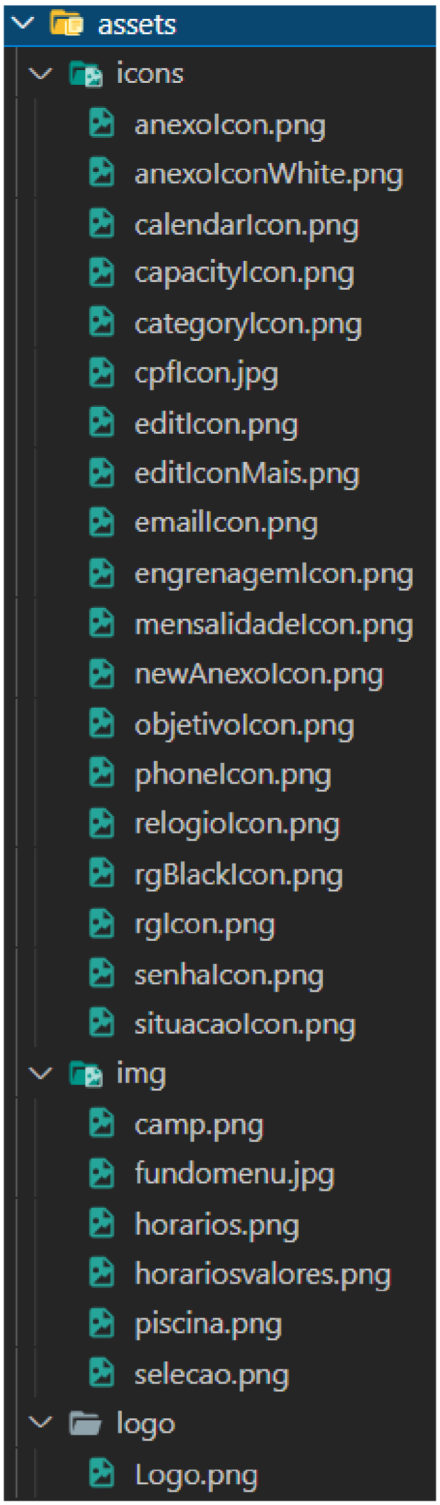
Fonte: Autoria própria (2022)

Figura 8 - estrutura da nossa pasta lib.



Fonte: Autoria própria (2022)

Figura 9 - estrutura da nossa pasta assets.



Fonte: Autoria própria (2022)

- **lib/assets:** Arquivo com os *icons* da pasta *assets,* instanciados. Nesse exemplo, um objeto do tipo *Widget* com o nome *anexoIcon*, no seguinte o tipo de *Widget* que ele é, nesse caso ele é do tipo *Image*, e após isso só setar a origem da localidade da imagem (Código 1).

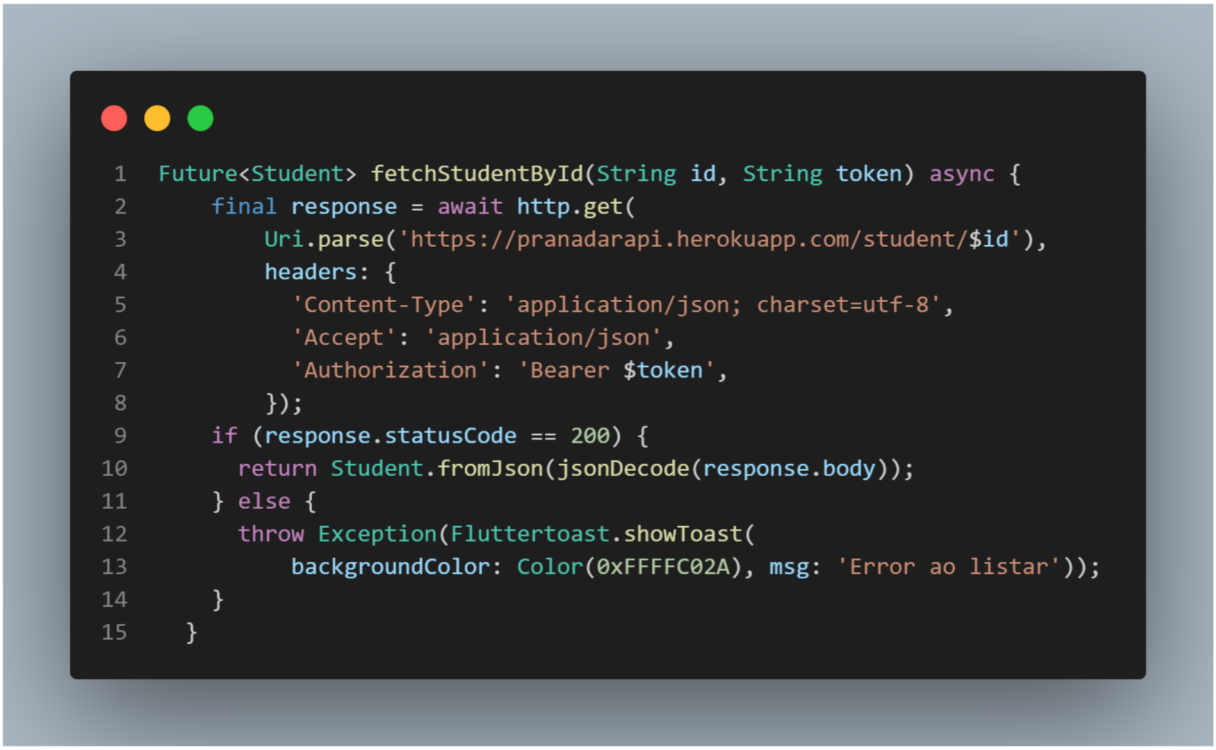
Código 1 - lib/assets

```
final Widget anexoIcon = Image.asset(
  'assets/icons/anexoIcon.png',
  width: 25,
  height: 25,
);
```

Fonte: Autoria própria (2022)

- **lib/controllers:** Diretório na qual se encontram os *controllers* para acessos aos verbos *http*, objetos e funções. Abaixo, uma função do verbo *http get* da *API* do *profileStudentPageController*, que através do ***ID*** é possível retornar um aluno (Código 2).

Código 2 - lib/controllers



```dart
Future<Student> fetchStudentById(String id, String token) async {
    final response = await http.get(
        Uri.parse('https://pranadarapi.herokuapp.com/student/$id'),
        headers: {
          'Content-Type': 'application/json; charset=utf-8',
          'Accept': 'application/json',
          'Authorization': 'Bearer $token',
        });
    if (response.statusCode == 200) {
      return Student.fromJson(jsonDecode(response.body));
    } else {
      throw Exception(Fluttertoast.showToast(
          backgroundColor: Color(0xFFFFC02A), msg: 'Error ao listar'));
    }
  }
```

Fonte: Autoria própria (2022)

- **lib/components:** Onde se encontra os *widget's*, componentes das telas, por exemplo, ao criar um *widget textformfield* é possível usá-lo em outras telas. O código abaixo é um exemplo do botão adicionar para a tela de turmas, nele tem como props um *token*, *typeUser*, *name*, *userId*, *emailuser*. Logo após essas propriedades são setadas como obrigatórias (Código 3).

Código 3 - lib/components

```dart
import 'package:flutter/material.dart';
import 'package:pranadarprojeto/profile/addProfile/addClassroom/addClassroomPage.dart';
import 'package:get/get.dart';

class ButtonAddClassroom extends StatefulWidget {
  final String? token;
  final String? typeUser;
  final String? name;
  final int? userId;
  final String? emailUser;

  ButtonAddClassroom({
    required this.token,
    required this.typeUser,
    required this.name,
    required this.userId,
    required this.emailUser,
  });
  @override
  _ButtonAddClassroomState createState() => _ButtonAddClassroomState();
}

class _ButtonAddClassroomState extends State<ButtonAddClassroom> {
  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.all(10.0),
      child: Align(
        alignment: Alignment.centerRight,
        child: Stack(
          children: [
            FloatingActionButton(
              backgroundColor: Color(0xFF002D60),
              onPressed: () {
                Get.to(AddClassroom(
                  token: widget.token,
                  typeUser: widget.typeUser,
                  emailUser: widget.emailUser,
                  name: widget.name,
                  userId: widget.userId,
                ));
              },
              child: Icon(Icons.add),
            ),
          ],
        ),
      ),
    );
  }
}
```

Fonte: Autoria própria (2022)

- **lib/login, lib/menu, lib/password, lib/profile, lib/splash:** Nelas estão as *pages* e telas do app. Segue abaixo um exemplo da tela de *login* (Figura 7).

Figura 10 - lib/login

Fonte: Autoria própria (2022)

- **lib/models:** Modelo dos dados que retorna da api. Segue abaixo um exemplo do *model/UnboundInstructor*. O *factory* do tipo *UnboundInstructor*, liga os objetos com os dados que recebe do *json*, já o *Map* irá gerar o *json* e torná-los em objetos. (Código 4).

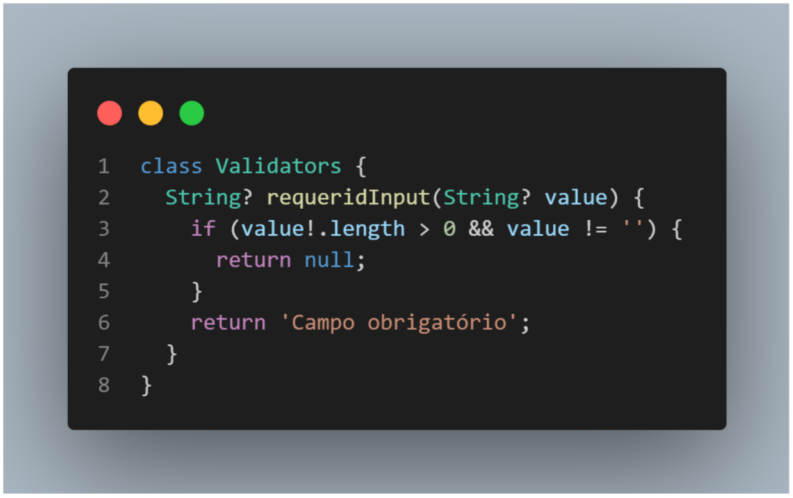
Código 4 - lib/model

```dart
class UnboundInstructor {
  int? instructorId;
  String? instructorName;

  UnboundInstructor({
    this.instructorId,
    this.instructorName,
  });

  factory UnboundInstructor.fromJson(Map<String, dynamic> json) {
    return UnboundInstructor(
      instructorId:
          json["instructor_id"] == null ? null : json["instructor_id"],
      instructorName: json["name"] == null ? null : json["name"],
    );
  }

  Map<String, dynamic> toJson() => {
        "instructor_id": instructorId,
        "name ": instructorName,
      };
}
```

Fonte: Autoria própria (2022)

- **lib/utils:** Diretório onde se encontra os arquivos de conexão, *inputFormatters*, *helper* - formatação dos valores dos *textformfield*, *Validators* - validação dos campos. Abaixo, uma função do tipo *String* que tem como parâmetro um valor, se o tamanho do valor for maior que 0 e valor diferente de vazio, ele retornará *null*, mas caso a condição não seja satisfeita ele retornará “Campo obrigatório” (Código 5).

Código 5 - lib/Validators.



```dart
class Validators {
  String? requeridInput(String? value) {
    if (value!.length > 0 && value != '') {
      return null;
    }
    return 'Campo obrigatório';
  }
}
```

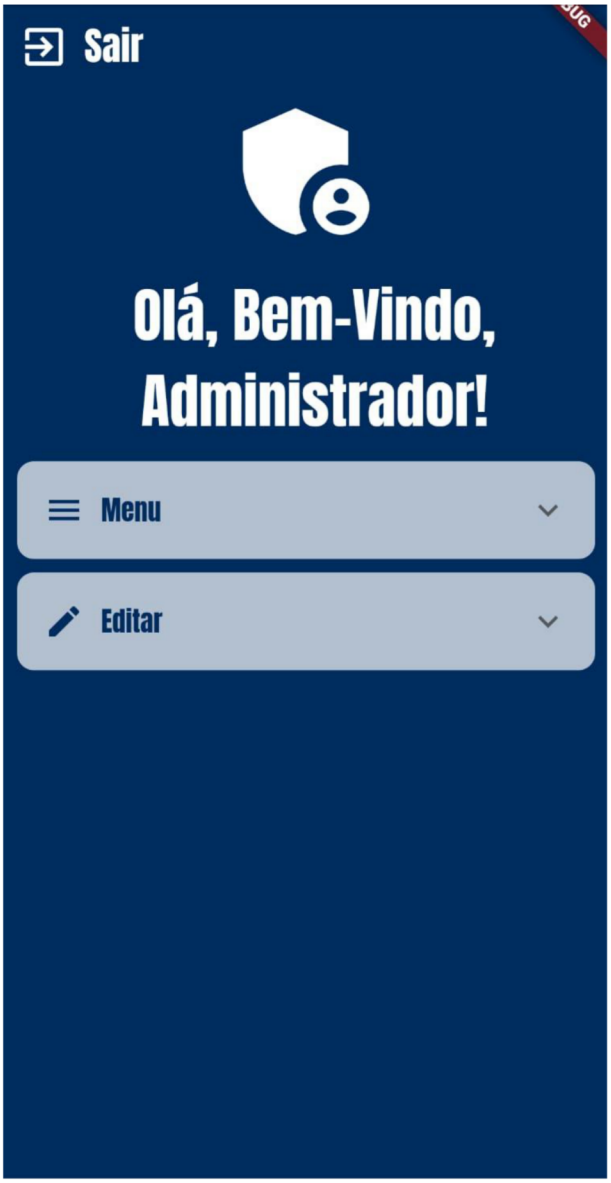
Fonte: Autoria própria (2022)

5.3 TELAS

Nesta seção são mostradas as telas do aplicativo em execução. As imagens foram capturadas utilizando o *software Vysor*.

A Figura 8 apresenta a tela inicial assim que o usuário realiza o *login*. Nela é mostrado a tela inicial do usuário do tipo administrador, com as opções menu e editar.

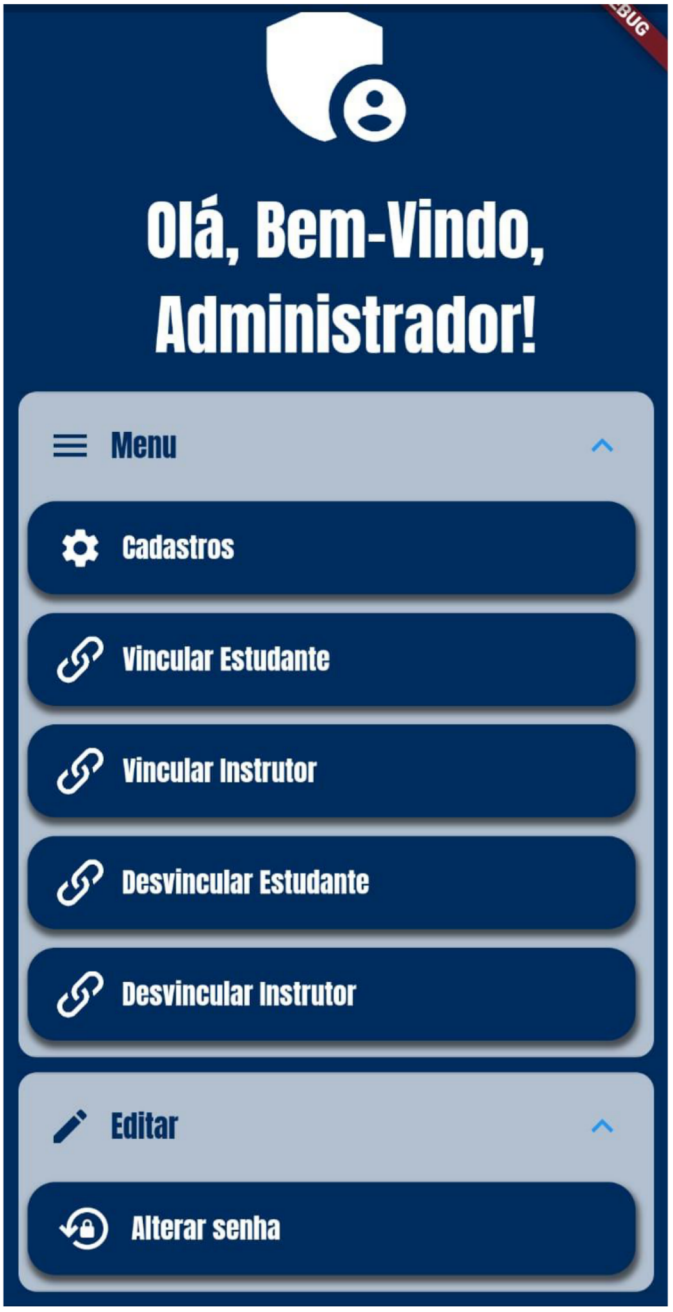
Figura 11 - tela administrador



Fonte: Autoria própria (2022)

Após clicar na opção menu, o usuário administrador pode realizar cadastros de alunos, instrutores, secretários e turmas. Pode vincular ou desvincular um determinado aluno a uma turma assim como um instrutor também.

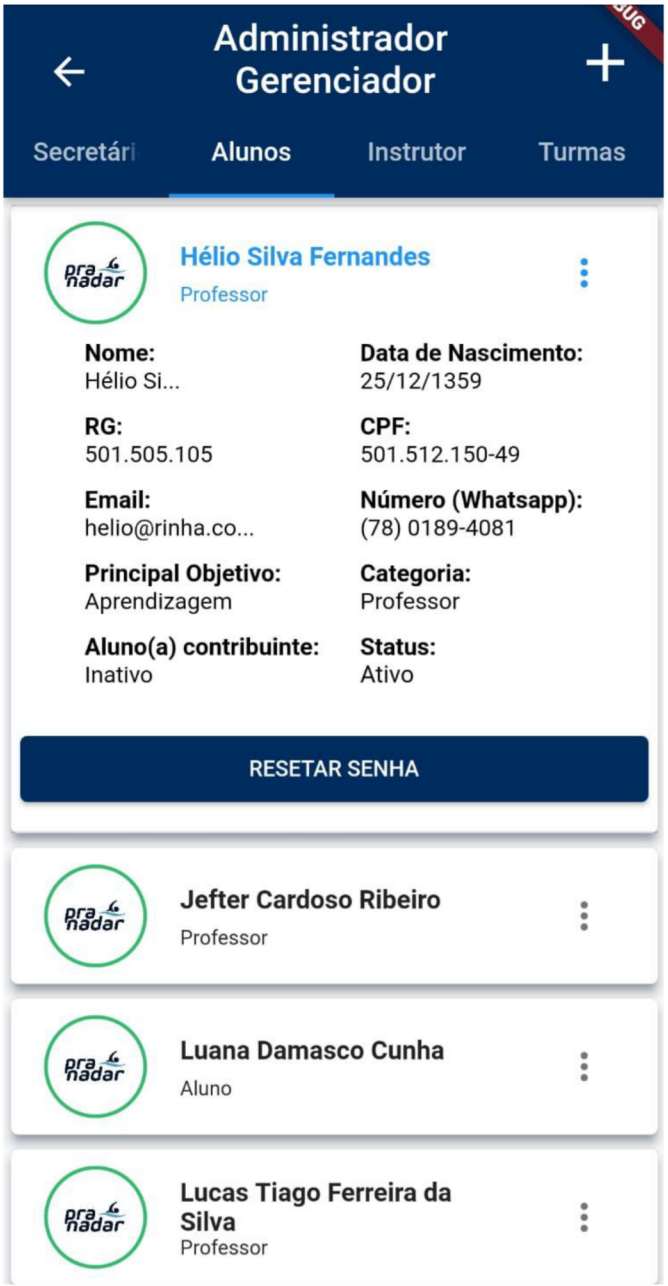
Figura 12 - tela admin, opções



Fonte: Autoria própria (2022)

Na figura 10, é listado todos os alunos cadastrados no sistema, exibindo os dados do aluno e abaixo o botão para resetar a senha.

Figura 13 - tela admin, com os dados do aluno e com o botão de resetar senha.



Fonte: Autoria própria (2022)

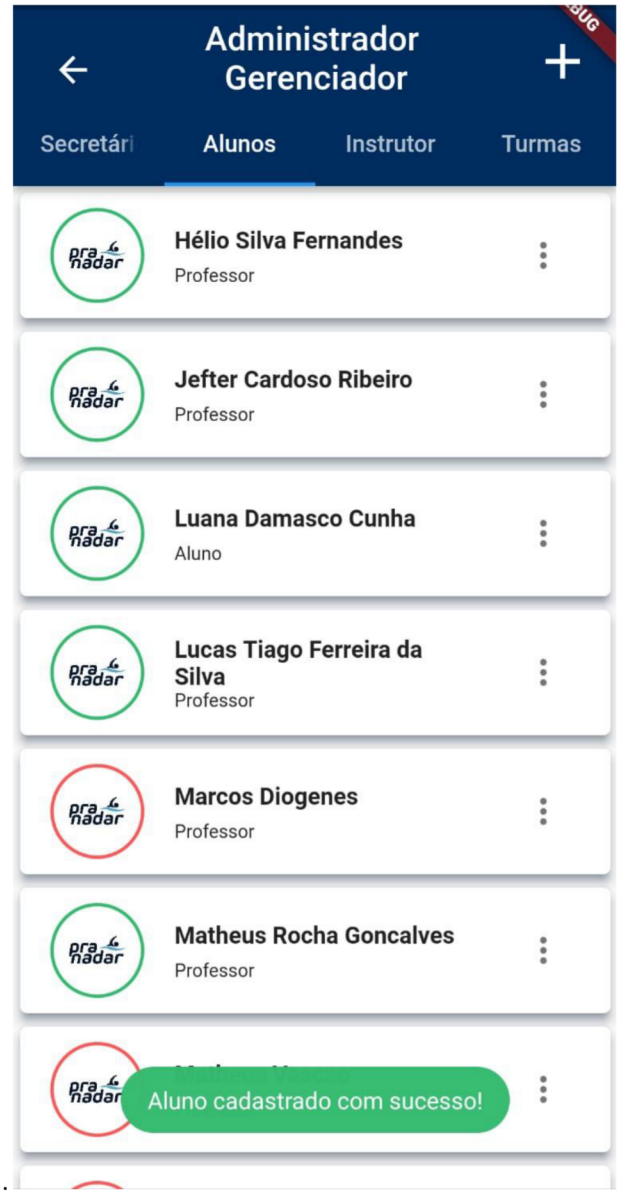
Para adicionar um aluno é necessário preencher todos os campos. Os campos possuem validações e formatações, portanto o aluno só será cadastrado se todos os campos estiverem preenchidos corretamente.

Figura 14 - tela admin, adicionando aluno.

Fonte: Autoria própria (2022)

Após realizar o cadastro do aluno, é mostrado uma mensagem de cor de fundo verde informando que o aluno foi cadastrado com sucesso.

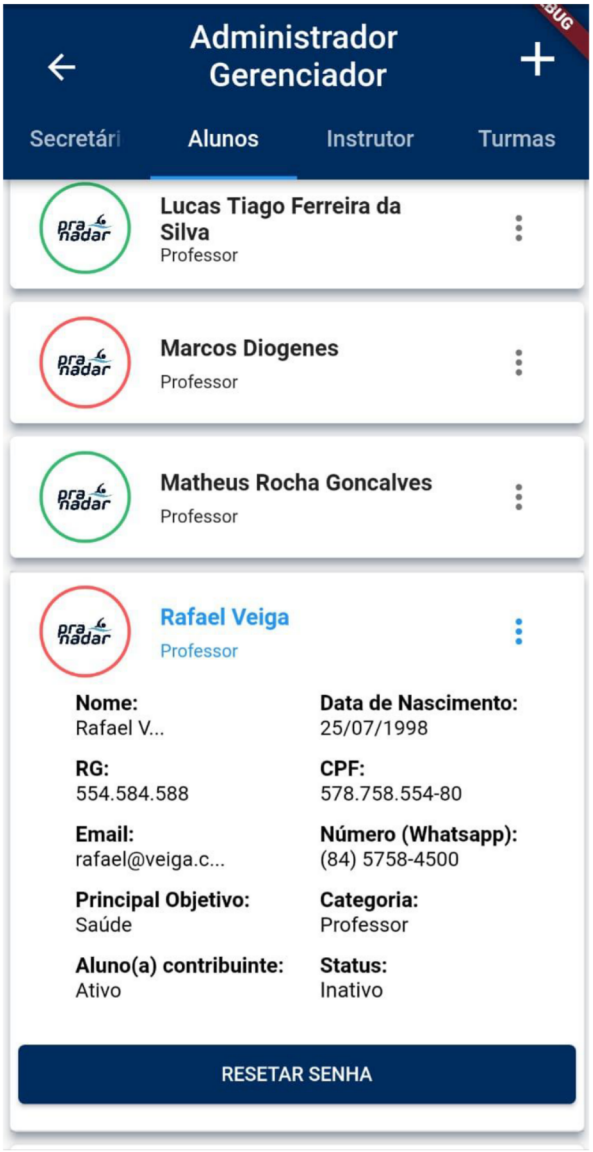
Figura 15 - tela admin, aluno cadastrado.



Fonte: Autoria própria (2022)

O aluno cadastrado tem por padrão um borda vermelha na imagem, pois aquele aluno está somente cadastrado no sistema e não ativo, após fazer o upload do atestado médico, essa borda muda para a cor verde, simbolizando que o aluno está ativo no sistema. Quando clicado, o card é expandido mostrando mais informações sobre o aluno.

Figura 16 **-** tela admin, aluno.



Fonte: Autoria própria (2022)

6 CONCLUSÃO

Os dispositivos móveis estão cada vez mais usados, um fato que aumentou o uso, foi a pandemia que fez com que várias empresas implementarem seus aplicativos gerando mais qualificações e empregos. Este trabalho mostrou a implementação do Pra nadar Mobile, um aplicativo que com todas as suas funcionalidades, obteve uma resposta muito positiva, pois está disponível na palma da mão uma ferramenta capaz de suprir as necessidades requisitadas pelo Pra Nadar. A previsão de lançamento da versão de produção está prevista para abril de 2022..

7 PERSPECTIVAS FUTURAS

As perspectivas futuras estão relacionadas a possibilidade de implementação da tela de *login* do aluno, a tela para visualizar, editar suas informações, foto de perfil; *upload* de arquivos do contribuinte, atestado médico ou alguma nova funcionalidade que seja necessária para melhorar o uso do aplicativo; relatório do sistema em arquivo e gerenciar as avaliações dos alunos; melhorias do código, como algumas funções, telas, importações para deixar o aplicativo mais leve; considera se interessante também fazer testes unitários já que não foi possível implementar.

REFERÊNCIAS

CONTA AZUL. **Metodologia Scrum: o que é, métodos ágeis e guia prático**.2021. Disponível em: https://blog.contaazul.com/metodologia-scrum. Acesso em: Março de 2022.

DART. **Dart language**. Disponível em: https://dart.dev/. Acesso em: Março de 2022

DISPOVYSOR. [S. l.], 4 abr. 2022. Disponível em: https://www.vysor.io/. Acesso em: 4 abr. 2022.

MICROSOFT. **Getting Started.** [S. l.], [202?]. Disponível em: https://code.visualstudio.com/docs. Acesso em: 04 Abril. 2022.

FABIO. **Introdução ao FDD - Feature Driven Development**. 2013. Disponível em: https://www.devmedia.com.br/introducao-ao-fdd-feature-driven-development/27971. Acesso em: Março de 2022

FLUTTER. **Flutter Mobile**. Disponível em: https://flutter.dev/. Acesso em: Março de 2022

QUEIROZ, Vinícius. **Flutter: o que é, como funciona, vantagens e desvantagens**:[S. l.], 20 dez. 2021. Disponível em: https://blog.tinnova.com.br/o-que-e-flutter/. Acesso em: 4 abr. 2022.