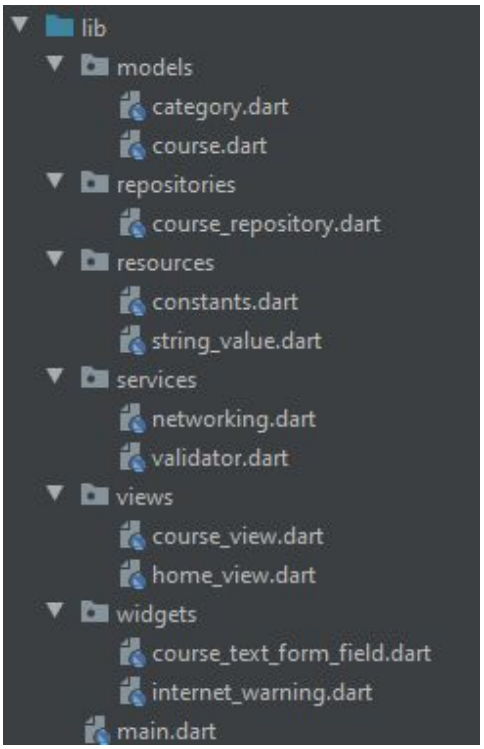
CAST - Flutter Versão 1.17.5

Frontend desenvolvido com Flutter, um framework criado e mantido pela Google.No qual é capaz de compilar Android, iOS, Windows, Mac, Linux, Google Fuchsia e Web usando a **Linguagem de Programação Dart**. Dependências utilizadas no projeto:

http versão ^0.12.0+2: pacote utilizado para fazer requisições HTTP
connectivity versão ^0.4.6+2: listener para qualquer mudança de estado da internet

Estrutura atual do projeto:



Models
Os models estão responsáveis pela validação, controle e manipulação de dados dos seus objetos modelos.

```dart
class CourseModel {
  final repository = CourseRepository();

  Future<Course> createCourse(Course course) async {
    return await repository.createCourse(course);
  }

  Future<List<Course>> getCourses() async {
    return await repository.getCourses();
  }

  Future<void> deleteCourse(int id) async {
    await repository.deleteCourse(id);
  }
}
```

Repository

Responsável por criar, deletar e buscar cursos (CRUD).

```dart
class CourseRepository {
  final network = Network(baseUrl);

  Future<Course> createCourse(Course course) async {
    var response = await network.post("/course", course.toJson());
    return Course.fromJson(response);
  }

  Future<List<Course>> getCourses() async {
    var response = await network.get("/courses");
    List<dynamic> data = response;
    List<Course> courses = data.map((e) => Course.fromJson(e)).toList();

    return courses;
  }

  Future<void> deleteCourse(int id) async {
    network.delete('/courses/$id');
  }
}
```

Validator e Networking

Ambos estão localizados dentro do service. O Validator possui validações que podem ser usadas e reutilizadas em outros lugares e passadas para formulários

```dart
class Validator {
  static String description(String value) {
    if (value != null) {
      if (value.trim().isEmpty) return 'Ops, a descrição não foi preenchido';
      if (value.trim().length < 1) return 'Mínimo de 1 caracteres';
    }
    return null;
  }

  static String data(String value) {
    if (value != null) {
      if (value.trim().isEmpty) return 'Ops, data não foi preenchida';
    }
    return null;
  }

  static String studentAmount(String value) {
    if (value != null) {
      if (value.trim().isEmpty)
        return 'A quantidade de aluno não foi preenchida';
    }
    return null;
  }
  static String email(String value) {
    if (value != null) {
      if (value.trim().isEmpty) return 'Ops, o e-mail não foi preenchido';
      if (!RegExp(
          r"^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,253}[a-zA-Z0-9])
          .hasMatch(value)) {
        return "Ops, email inválido";
      }
    }
    return null;
  }
```

O networking realiza as chamadas na API. O intuito era isolar ele como se fosse um interceptor, caso precisasse passar algum token no header ou algum outro tipo de configuração.

```dart
final String _baseUrl;
var _header = {"Content-Type": "application/json"};

Future<void> delete(String path) async {
  await http.delete(_baseUrl + path);
}

Future<dynamic> get(String path) async {
  try {
    http.Response response = await http.get(_baseUrl + path);

    if (response.statusCode == 200) {
      var data = response.body;

      var decodedData = jsonDecode(data);

      return decodedData;
    }
  } catch (e) {
    throw Exception(e.toString());
  }
}

Future<dynamic> post(String path, Map<String, dynamic> data) async {
  var body = jsonEncode(data);

  try {
    http.Response response =
        await http.post(_baseUrl + path, headers: _header, body: body);

    if (response.statusCode == 200) {
      var data = response.body;

      var decodedData = jsonDecode(data);
```

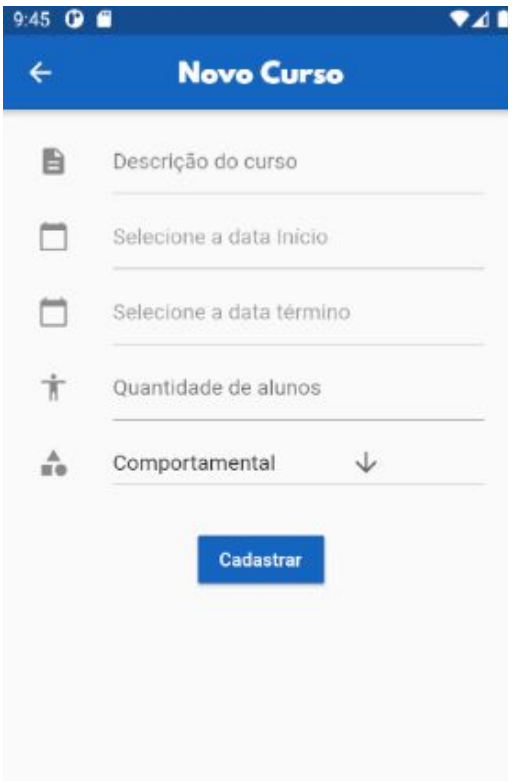
Views e Widget

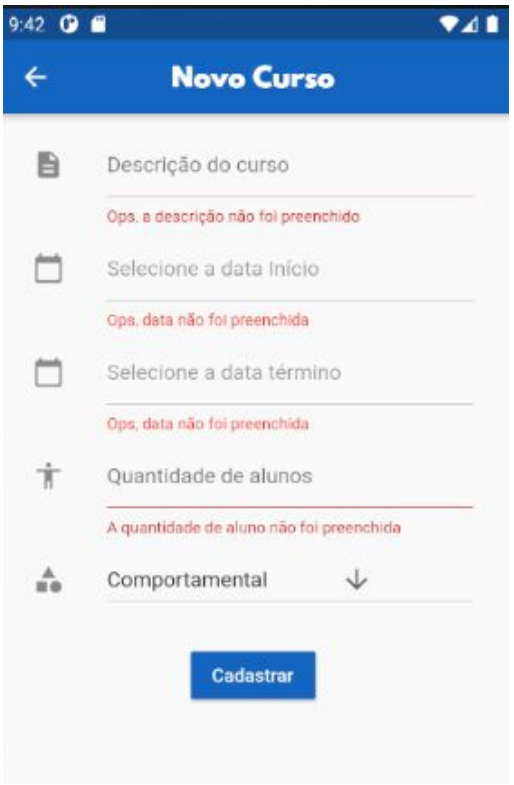
As views são as interfaces que interagem com os usuários, as mesmas são compostas por widgets.

Fluxo base do aplicativo

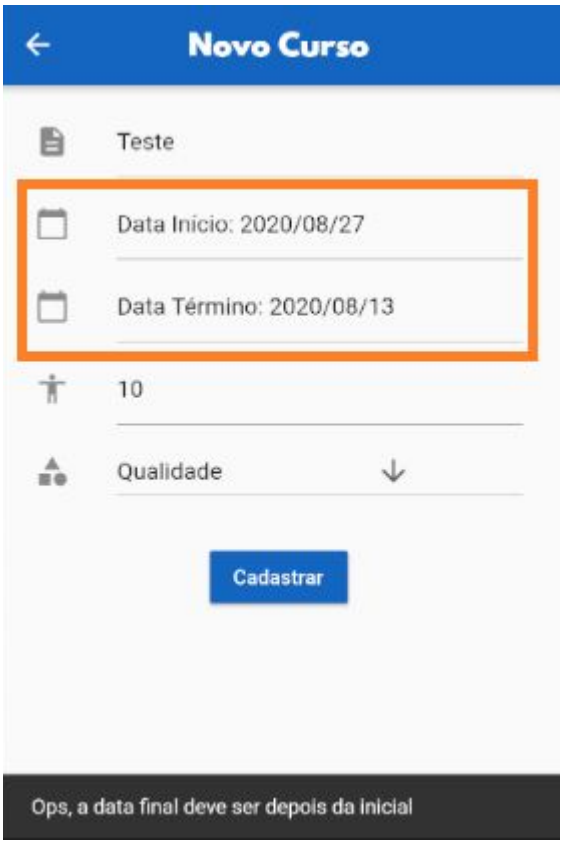
Home View onde o usuário pode ver, filtrar e cadastrar os cursos.

Quando o usuário clicar no botão “+”, um formulário aparecerá para o cadastro do novo curso. Cada campo possui seu próprio validador, caso o usuário insira dados inválidos ou deixe os campos em branco, o mesmo será notificado sobre o erro encontrado.

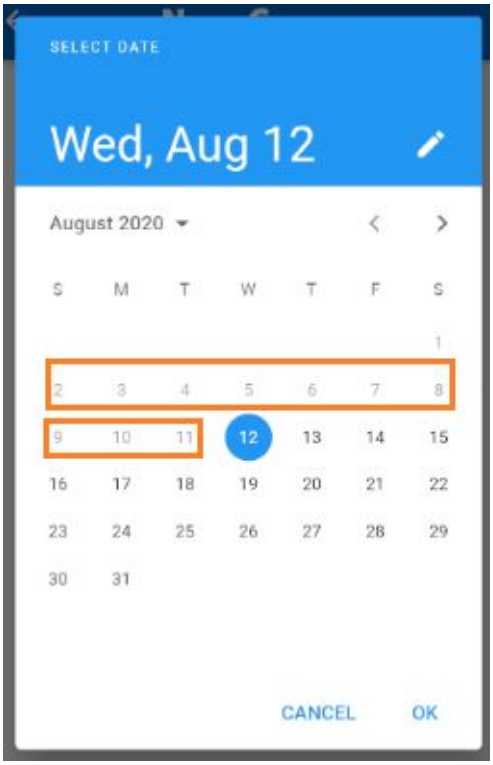




Se o usuário inserir dados válidos, porém que não respeitem a regra de negócio, uma mensagem será exibida no rodapé do app. Como no caso mostrado a seguir.



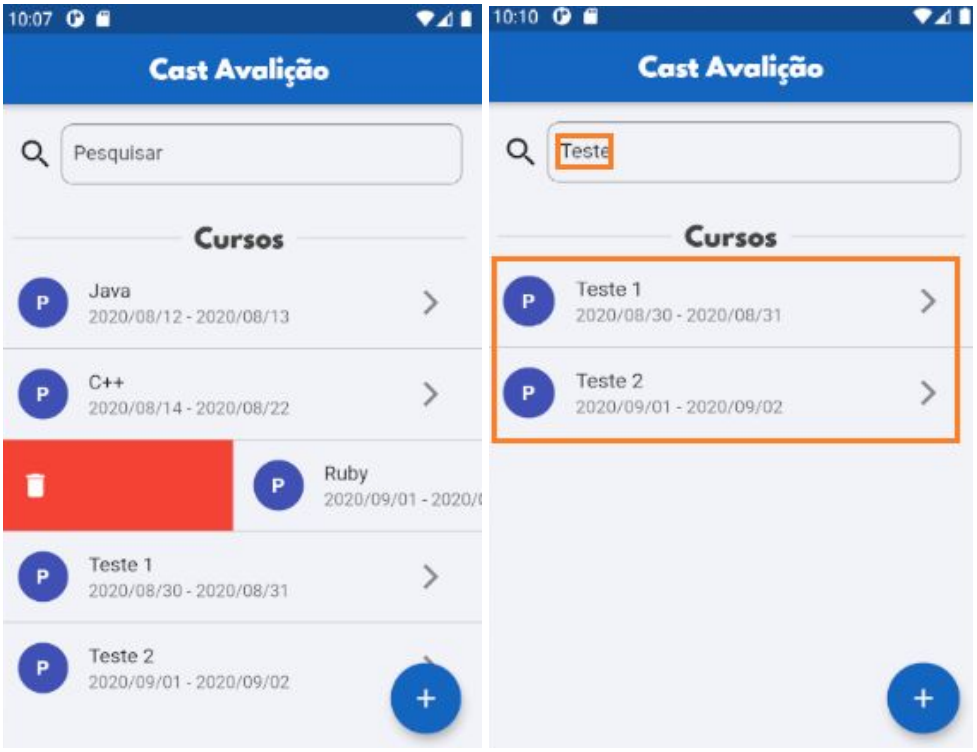
Aplicando a **regra de negócio 2 da avaliação técnica**, o usuário não será capaz de escolher uma data anterior ao dia atual. Sendo assim, as datas anteriores ao dia atual vem desabilitada por padrão.



Ao realizar o cadastramento do curso ou a alteração de algum existente, uma notificação será mostrada no rodapé.

e/ou

Para excluir um curso cadastrado basta deslizar o curso desejado da esquerda para direita. Digite na caixa de pesquisa para filtrar tanto por nome quanto por categoria



Para evitar tentativas desnecessária na API, o app tem um listener no status da internet e caso o mesmo perca a conexão, uma mensagem será exibida. O modo avião foi ativado para forçar tal cenário.