Aplicativa Login

- Utiliza um serviço API RESTful com informações dos usuários
- Cria uma tela de login funcional

JSON Server

- JSON Server é uma ferramenta que permite criar uma API RESTful de forma rápida e simples utilizando arquivos JSON como fonte de dados.
- API RESTful utiliza os métodos HTTP (como GET, POST, PUT e DELETE) para realizar operações em recursos, que podem ser representados por URLs específicas.
- É uma biblioteca do Node.js que simula um servidor backend, fornecendo endpoints para consulta, criação, atualização e exclusão de dados.

1. Criar base de dados JSON

2. Recuperar Informações via API

3. Exibir Aplicação Funcionando

4. Esqueleto da Aplicação

4.1 Refatorar arquivo main.dart

4.2 Criar entidade para representar um usuário cadastrado

4.3 Criar widget para o botão

``dart class AppTextField extends StatefulWidget {

```
- Colocar os campos necessários
```dart
  final labelText;
  final String? Function(String?) onValidate;
  final Function(String?) onSaved;
  var keyboardType = TextInputType.text; // tipo do teclado padrão
  var obscureText = false; // por padrão exibe o texto
```

- Criar os construtores
    1. Padrão
    2. Para um TextField com senha
        - TextInputType.visiblePassword
        - obscureText = true
- Criar o layout do botão
    1. Padding
    2. TextFormField
```

        - keyboardType
        - obscureText
        - onSaved
        - validator
        - decoration
            - laberText
            - border: OutlineInputBorder

4.4 Criar arquivo login.dart

4.5 Criar classe Login

4.6 Criar chave de formulário

```
final formKey = GlobalKey<FormState>();
```

- GlobalKey é uma classe que permite criar e gerenciar uma chave global para um widget específico. Um GlobalKey pode ser usado para acessar o estado interno de um widget e interagir com ele a partir de outras partes do aplicativo.
- No caso específico de FormState, GlobalKey é usado para criar uma chave global que controla e valida o estado de um formulário no Flutter.
- Usando uma chave global, você pode acessar o estado de um formulário de qualquer lugar em seu aplicativo e realizar operações como validação, redefinição ou envio do formulário.
- Criar Classe

```
class Login extends StatefulWidget {
```

- Colocar variáveis de instância

```
  String username = "";
  String password = "";
```

- Construir layout
    - Form
        - key
    - AppTextField (username)
        - criar validação
        - criar salvamento
    - AppTextField (password)
        - criar validação
        - criar senha
    - Botão Enviar (ElevatedButton)
        - validar estado atual do formulário

```
final isValid = formKey.currentState!.validate();
```

```
    - salvar valores do formulário
```

```
formKey.currentState!.save();
```

```
    - pegar todos os usuários e filtrá-los
```

````
final users = await LoginClient.fetchAll();
final user = users.firstWhere(
  (obj) => obj['username'] == username,
  orElse: () => {},
);
```      - checar se a senha é válida
        - exibir snack bar para
            - senha inválida
            - senha válida
        - caso logado com sucesso navegar para janela principal
```dart
if (isValidPassword) {
    Navigator.pushNamed(context, '/home');
} else {
    const snackBar = SnackBar(
      content: Text('Nome de usuário ou senha inválidos!'),
    );
    ScaffoldMessenger.of(context).showSnackBar(snackBar);
}
````

4.7 Criar cliente para o servidor JSON

```
import 'dart:convert';

import 'package:http/http.dart' as http;

class LoginClient {
  static const host = "http://localhost:3031/user";
  static dynamic fetchAll() async {
    final uri = Uri.parse(host);
    final response = await http.get(uri);
    if (response.statusCode == 200) {
      return jsonDecode(response.body);
```

```
      }
      return {};
    }
  }
```