

# Flutter & Dart Complete Guide

Prepared by Eng. Yousef Salah Kassem

DAFL 3  
API

**Lecture 5**



Chapter  
**05**

## Table Of Content

- Introduction
- Rest Api
- FakeStore API Application



# Introduction

# Introduction

## What is REST API?

### Definition:

A *REST* (Representational State Transfer) API lets two systems talk to each other over HTTP, like a waiter taking your order at a restaurant.





# Rest Apis

# Rest API

## Key Concepts

### Resources

- *What are they?*  
Resources are like nouns: "Users," "Products," "Cats with funny hats."

URL for a resource:

`GET /users/123`

- *(Fetch details of user #123)*

### Roast:

- REST resources act like VIPs:

*You must approach with the exact ID and format, or they'll ignore you.*



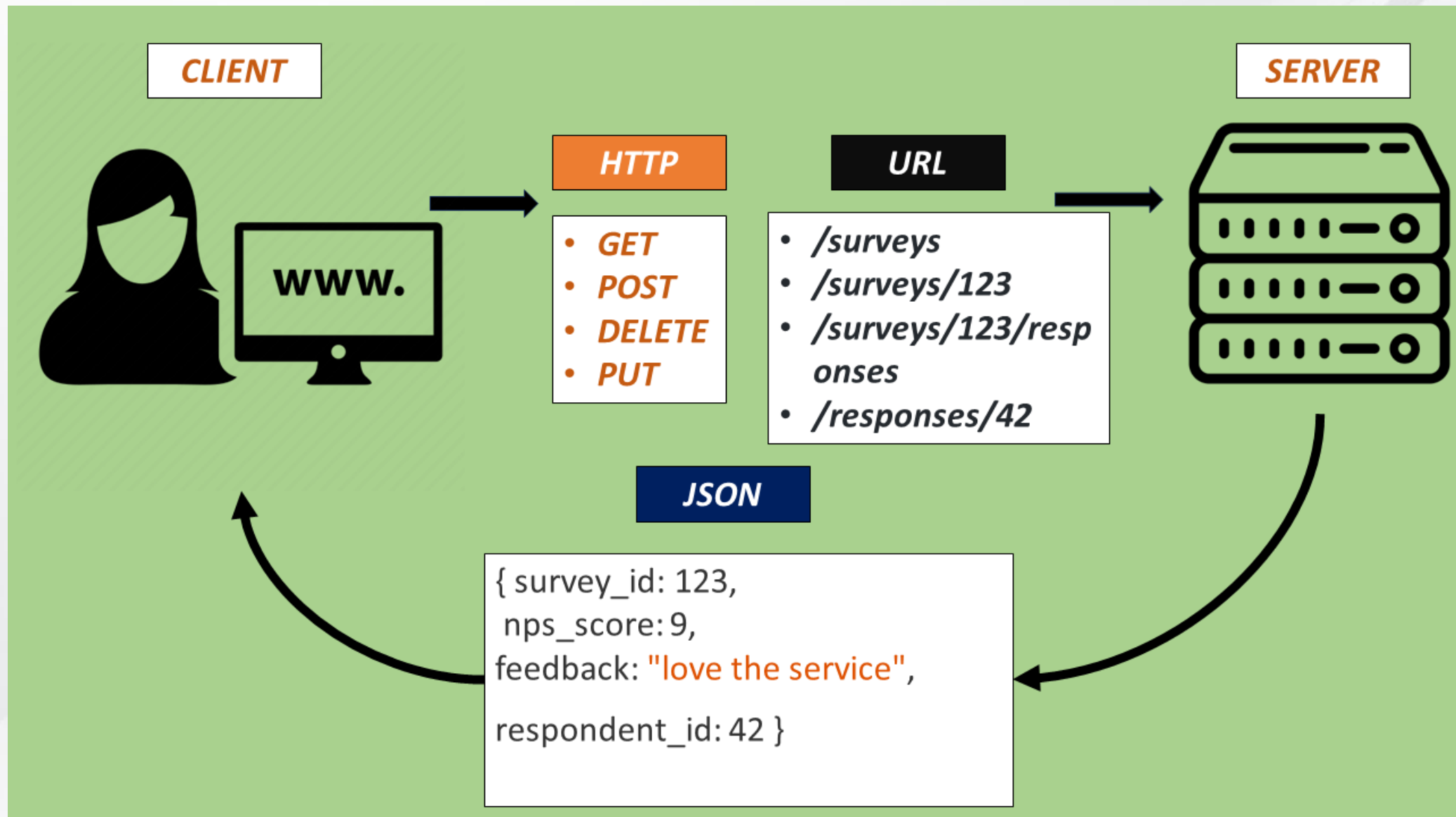
# Rest API

## HTTP Methods

REST uses HTTP methods to interact with resources. Think of them like verbs:

| HTTP Method | Action                   | Example                              |
|-------------|--------------------------|--------------------------------------|
| GET         | Fetch data (Read)        | GET /users (Get all users)           |
| POST        | Create data              | POST /users (Add a new user)         |
| PUT         | Replace data (Overwrite) | PUT /users/123 (Replace user #123)   |
| DELETE      | Remove data              | DELETE /users/123 (Delete user #123) |
| PATCH       | Update part of data      | PATCH /users/123 (Change name only)  |





# REST API

## 1. Fetch all books:

### Client:

GET /books

### Server Response:

```
json
[
  {"id": 1, "title": "1984"},
  {"id": 2, "title": "The Catcher in the Rye"}
]
```

**Roast:** "You asked for *all books*. Hope you have storage for the 10GB response."



# REST API

**Add a new book:**

**Client:**

POST /books

```
{  
  "title": "Harry Potter"  
}
```

**Roast:** REST doesn't know how to spell "magic." It just blindly saves whatever you send.

**Server Response:**

```
json  
  
{  
  "id": 3,  
  "title": "Harry Potter"  
}
```



# Status Codes

| Code | Meaning               | Roast                                          |
|------|-----------------------|------------------------------------------------|
| 200  | OK                    | <i>Congrats, you didn't screw up.</i>          |
| 201  | Created               | <i>Wow, you made something. Clap, clap!</i>    |
| 400  | Bad Request           | <i>Your input is garbage.</i>                  |
| 401  | Unauthorized          | <i>No key? No entry.</i>                       |
| 404  | Not Found             | <i>You lost something? So did the server.</i>  |
| 500  | Internal Server Error | <i>You broke it. The server hates you now.</i> |



# **Fake Store API Flutter Application**

# Project

## Problem Story: The FakeStore API Application

You have been hired as a **backend developer** for a new startup called **FakeStore CLI Services**. The company's mission is to create a command-line tool that interacts with the [FakeStore API](#) to enable users to manage their shopping experience directly from the terminal.

Your task is to create a **Flutter application** that provides users with the following features:



# Project

## 1. Login:

Users should be able to log in using their username and password to authenticate with the FakeStore API. Upon successful login, they receive a token, which will be displayed.

## 2. View Product Categories:

Users can fetch and view a list of product categories available in the store.

## 3. View Products by Category:

Users can choose a specific category and view all products under that category. Each product should display its:

- a. ID
- b. Title
- c. Price



# Project

## User-Friendly Experience:

The application should use a **menu-based system** to guide users through the options.

## Error Handling:

The application must handle errors gracefully. For example:

- Invalid credentials during login.
- Network issues or timeouts.
- Invalid inputs, such as choosing a non-existent category.

## Code Requirements:

- Follow **OOP principles** by dividing responsibilities into classes like `ApiClient`, `AuthService`, and `ProductService`.
- Use the **Dio** package for HTTP requests.
- Ensure the application is modular, readable, and maintainable.



# Project

## User-Friendly Experience:

The application should use a **menu-based system** to guide users through the options.

## Error Handling:

The application must handle errors gracefully. For example:

- Invalid credentials during login.
- Network issues or timeouts.
- Invalid inputs, such as choosing a non-existent category.

## Code Requirements:

- Follow **OOP principles** by dividing responsibilities into classes like `ApiClient`, `AuthService`, and `ProductService`.
- Use the **Dio** package for HTTP requests.
- Ensure the application is modular, readable, and maintainable.



# Project

- Add the **dio** package to your **pubspec.yaml**:

```
dependencies:  
  dio: ^5.0.0
```



## User login

```
fetch('https://fakestoreapi.com/auth/login',{
  method:'POST',
  body:JSON.stringify({
    username: "mor_2314",
    password: "83r5^_"
  })
})
.then(res=>res.json())
.then(json=>console.log(json))
```

Hide output

```
//output
{
  token: "eyJhbGciOiJIUzI1NiIsInR"
}
```

## Products

### Get all products

```
fetch('https://fakestoreapi.com/products')  
  .then(res=>res.json())  
  .then(json=>console.log(json))
```

Hide output

```
//output  
[  
  {  
    id:1,  
    title:'...',  
    price:'...',  
    category:'...',  
    description:'...',  
    image:'...'  
  },  
  /*...*/  
  {  
    id:30,  
    title:'...',  
    price:'...',  
    category:'...',  
    description:'...',  
    image:'...'  
  }  
]
```

## Get all categories

```
fetch('https://fakestoreapi.com/products/categories')  
  .then(res=>res.json())  
  .then(json=>console.log(json))
```

Hide output

```
//output  
[  
  "electronics",  
  "jewelery",  
  "men's clothing",  
  "women's clothing"  
]
```

<https://www.figma.com/design/jDCfm30VHot486hrCqrCsX/Grocery-App--Big-Cart---Community-?node-id=2464-1073&p=f&t=tZJLKJnXI7erYLxH-0>

Splash1-4

### Get Discounts On All Products

Lorem ipsum dolor sit amet, consetetur  
sadipscing elitr, sed diam nonumy



Get started

Splash1-3

### Buy Premium Quality Fruits

Lorem ipsum dolor sit amet, consetetur  
sadipscing elitr, sed diam nonumy



Get started

Splash1-2



Splash1-1



Auth1-welcome

A mobile app interface for a grocery store. The top half features a background image of a woman smiling while pushing a shopping cart filled with various fruits and vegetables in a grocery store aisle. The bottom half is a white card with a 'Welcome' heading, a paragraph of placeholder text, a 'Continue with google' button with the Google logo, and a link for users who already have an account.

9:41

Welcome

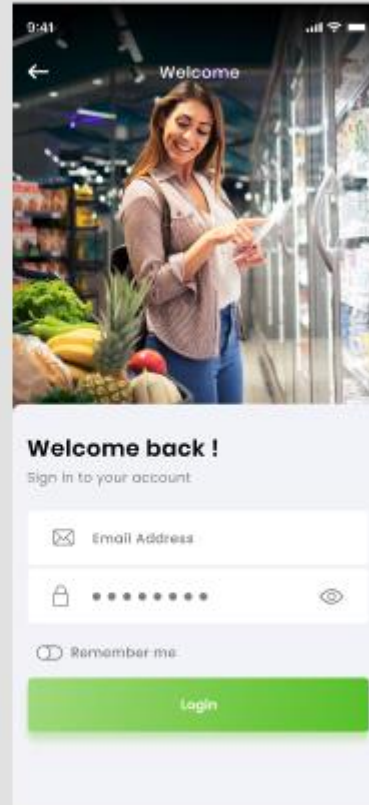
**Welcome**

Lorem ipsum dolor sit amet, consetetur  
sodipscing elitr, sed diam nonumy

 Continue with google

Already have an account ? [login](#)

Auth1-login

A mobile app interface for a grocery store, showing the login screen. The top half features a background image of a woman smiling while looking at a smartphone in a grocery store aisle. The bottom half is a white card with a 'Welcome back !' heading, a 'Sign in to your account' subheading, input fields for 'Email Address' and a password, a 'Remember me' checkbox, and a green 'Login' button.

9:41

Welcome

**Welcome back !**

Sign in to your account

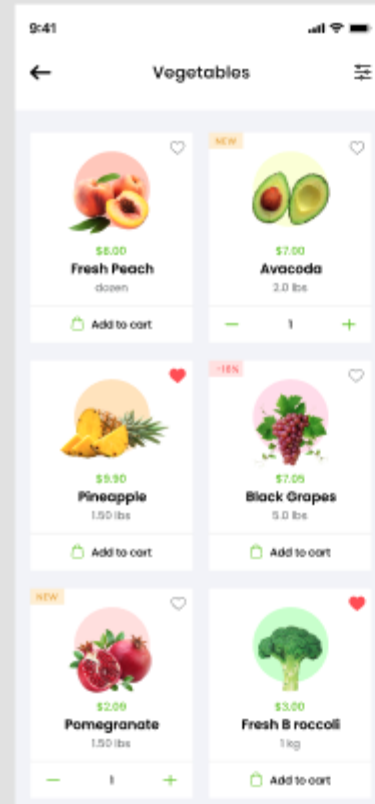
 Email Address

 •••••••• 

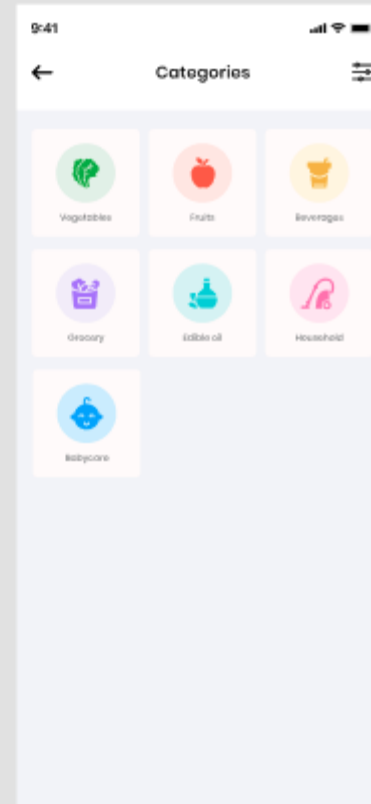
☐ Remember me

Login

## Products



## Category



product detail

9:41





\$2.22

Organic Lemons

1.50 lbs

4.5      (89 reviews)

Organic Mountain works as a seller for many organic growers of organic lemons. Organic lemons are easy to spot in your produce aisle. They are just like regular lemons, but they will usually have a few more scars on the outside of the lemon skin. Organic lemons are considered to be the world's finest lemon for juicing [more](#)

Quantity

-

3

+

Add to cart



cart

9:41

 Shopping Cart



\$2.22 x 4

Fresh Broccoli

1.50 lbs

+

5

-



\$2.22 x 4

Black Grapes

5.0 lbs

+

5

-





\$2.22 x 4

Avocado

1.50 lbs

+

5

-



\$2.22 x 4

Pineapple

4.0 lbs

+

5

-

Subtotal

\$56.7

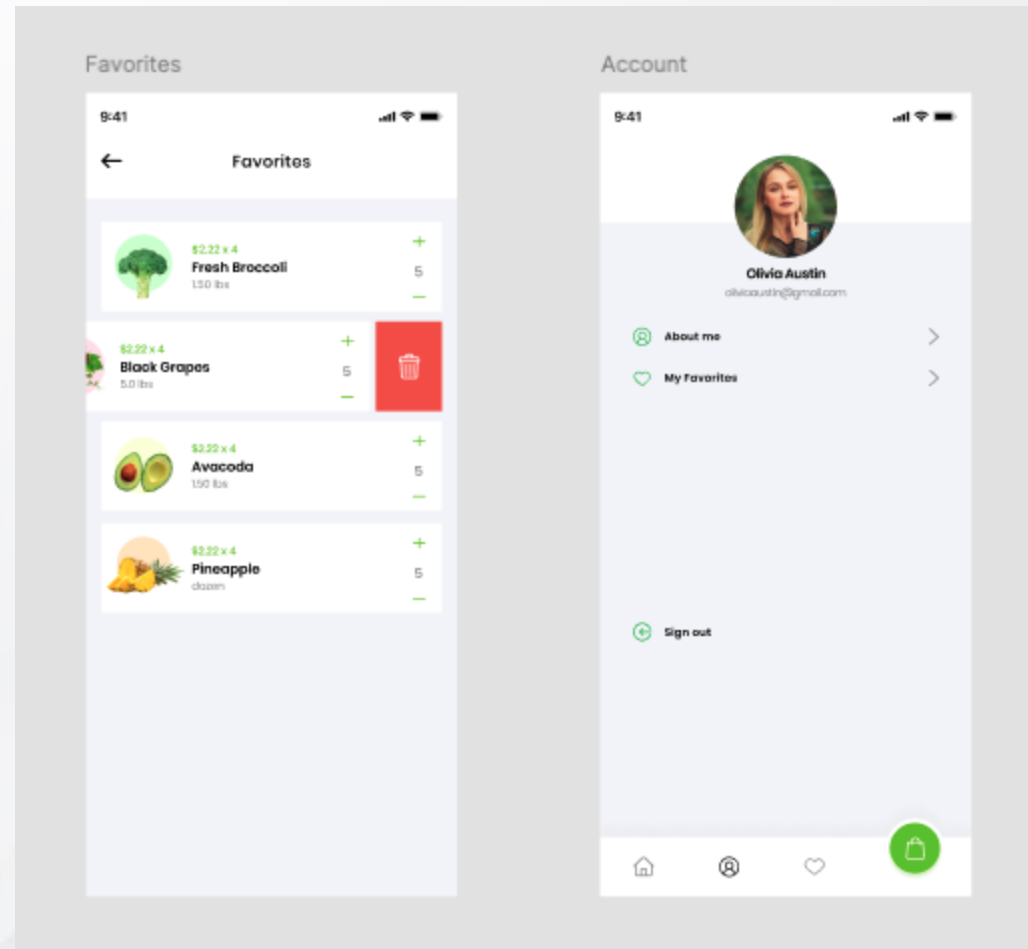
Shipping charges

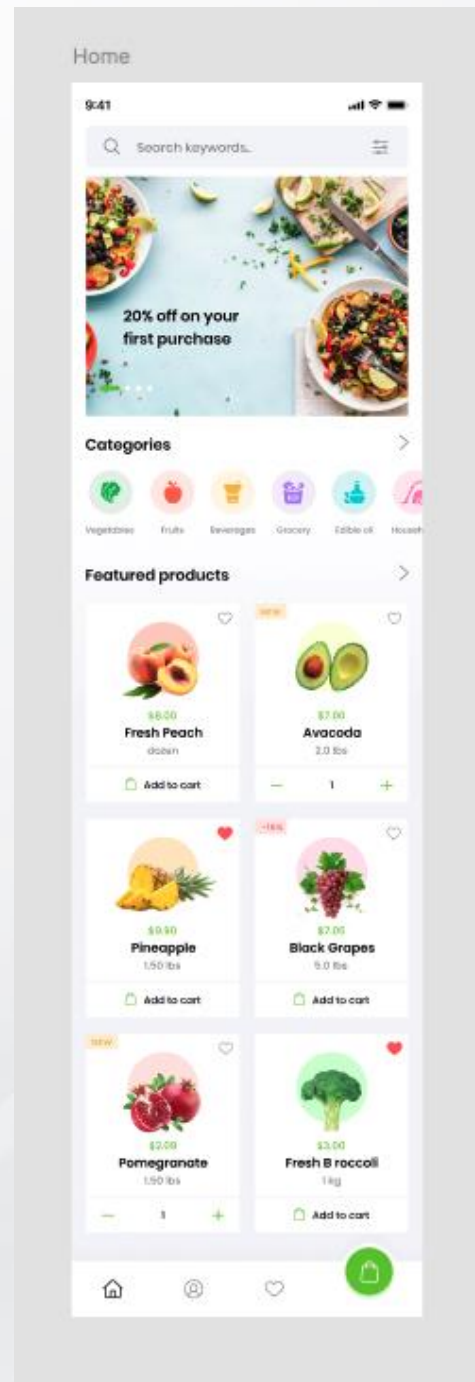
\$0.0

Total

\$58.2

Checkout





About me

9:41



About me

Personal Details



Russell Austin



russell.partner@gmail.com



+1 202 555 0142

Change Password



Current password



•••••



Confirm password

Save settings

```
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:midocean_app/core/cache/base_caching_service.dart';

final authControllerProvider = ChangeNotifierProvider((ref) {
  return AuthController(
    ref.watch(baseCachingServiceProvider(CachingType.secure)),
  ); // AuthController
}); // ChangeNotifierProvider

class AuthController extends ChangeNotifier {
  AuthController(this._cachingService);

  final BaseCachingService _cachingService;

  bool _isLoggedIn = false;

  bool get isLoggedIn => _isLoggedIn;

  Future<void> checkIfUserLoggedIn() async {
    ⚡ _isLoggedIn = await _cachingService.readString('token') != null;
    debugPrint('Is logged in: $_isLoggedIn');
    notifyListeners();
  }

  Future<void> logout() async {
    await _cachingService.delete('token');
    _isLoggedIn = false;
    notifyListeners();
  }
}
```

**POST**  `https://fakestoreapi.com/auth/login`

Params   Authorization   Headers (9)   **Body**    Scripts   Settings

☐ none   ☐ form-data   ☐ x-www-form-urlencoded   ☒ raw   ☐ binary   ☐ GraphQL   **JSON** 

```
1 {
2   "username": "mor_2314",
3   "password": "83r5^_"
4 }
```

Body   Cookies   Headers (14)   Test Results    **200 OK**

**{ }**     Preview    Visualize   

```
1 [
2   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJlbnVzZXIiOiJ0b3JfMjM1IiwiaWF0IjoxNjE0ODUwMDA.cQe0CwPJsTKJRsQE3aGymhQfrkn1n16E0oXVWzCMb08"
3 ]
```

```
class LoginController extends AsyncNotifier {
  @override
  FutureOr build() {}

  final TextEditingController emailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();
  final formKey = GlobalKey<FormState>();

  final _loginPath = 'auth/login';

  Future<void> login() async {
    if (!formKey.currentState!.validate()) return;
    state = const AsyncValue.loading();

    final result = await ref
      .read(dioConsumerProvider)
      .post(
        _loginPath,
        data: {
          'username': emailController.text,
          'password': passwordController.text,
        },
      );

    result.fold(
      (failure) {
        state = AsyncValue.error(...);
      },
      (data) {
        final token = data['token'] as String;
        ref.read(dioConsumerProvider).addToken(token);
        ref
          .read(baseCachingServiceProvider(CachingType.secure))
          .writeString('token', token);
        state = AsyncValue.data(token);
      },
    );
  }
}
```

```
final routerProvider = Provider((ref) {
  return GoRouter(
    initialLocation: Routes.splash.path,
    routes: [
      GoRoute(
        path: Routes.splash.path,
        builder: (context, state) => const SplashScreen(),
        redirect: (context, state) async {
          final isSplashViewedBefore =
            ref.watch(SplashController.provider).isSplashViewedBefore;

          if (!isSplashViewedBefore) {
            return null;
          } else {
            return Routes.login.path;
          }
        },
      ), // GoRoute
      GoRoute(
        path: Routes.login.path,
        builder: (context, state) => const LoginScreen(),
        redirect: (context, state) async {
          final isUserLoggedIn = ref.watch(authControllerProvider).isLoggedIn;

          if (!isUserLoggedIn) {
            return null;
          } else {
            return Routes.home.path;
          }
        },
      ), // GoRoute
    ],
  ); // GoRouter
}); // Provider
```

```
class SplashController extends ChangeNotifier {
  SplashController(this._cachingService) {
    pageController.addListener(() {
      currentIndex = pageController.page!.toInt();
      notifyListeners();
    });
  }

  static final provider = ChangeNotifierProvider(
    (ref) => SplashController(
      ref.watch(baseCachingServiceProvider(CachingType.simple)),
    ), // SplashController
  ); // ChangeNotifierProvider

  final pages = SplashModel.pages;

  final BaseCachingService _cachingService;

  int currentIndex = 0;

  final PageController pageController = PageController();

  bool isSplashViewedBefore = false;
  final _splashKey = 'isSplashViewedBefore';

  Future<void> checkIfSplashViewedBefore() async {
    isSplashViewedBefore = await _cachingService.containsKey(_splashKey);
    notifyListeners();
  }

  Future<void> setSplashViewed() async {...}

  @override
  void dispose() {
    pageController.dispose();
    super.dispose();
  }
}
```

# **What is Next ?**



**Thank You**