**Coffee API Service WITH API IMPLEMENTATION**

import 'dart:convert';  
import 'package:http/http.dart' as http;  
import '../models/coffee\_model.dart';  
  
class CoffeeApiService{  
 static const String *baseUrl* = 'http://localhost:3001/api';  
  
 //Get all coffees  
 static Future<List<Coffee>> *getAllCoffees*() async {  
 try {  
 final response = await http.get(Uri.*parse*('$*baseUrl*/coffees'));  
 if (response.statusCode == 200){  
 final Map<String, dynamic> data = json.decode(response.body);  
 final List<dynamic> coffeesJson = data['coffees'];  
 return coffeesJson.map((json) => Coffee.fromJson(json)).toList();  
 } else {  
 throw Exception('Failed to load coffees');  
 }  
 } catch(e){  
 throw Exception('Failed to connect to API: $e');  
 }  
 }  
  
 //Get coffee by ID  
 static Future<Coffee> *getCoffeeById*(int id) async {  
 try {  
 final response = await http.get(Uri.*parse*('$*baseUrl*/Coffees/$id'));  
 if(response.statusCode == 200){  
 final Map<String,dynamic> coffeeJson = json.decode(response.body);  
 return Coffee.fromJson(coffeeJson);  
 }  
 else throw Exception('Failed to load coffee');  
 }catch(e){ throw Exception('Failed to connect to API: $e');  
 }  
 }  
  
 //Get coffees by category  
 static Future<List<Coffee>> *getCoffeesByCategory*(String category) async {  
 try {  
 final response = await http.get(Uri.*parse*('$*baseUrl*/coffees/category/$category'));  
 if(response.statusCode == 200){  
 final Map<String, dynamic> data = json.decode(response.body);  
 final List<dynamic> coffeesJson = data['coffees'];  
 return coffeesJson.map((json) => Coffee.fromJson(json)).toList();  
 }else throw Exception('Failed to load coffees by category');  
 }catch(e){  
 throw Exception('Failed to connect to API: $e');  
 }  
 }  
  
 //Get all Categories  
 static Future<List<String>> *getCategories*() async {  
 try {  
 final response = await http.get(Uri.*parse*('$*baseUrl*/categories'));  
 if(response.statusCode == 200){  
 final Map<String, dynamic> data = json.decode(response.body);  
 final List<dynamic> categoriesJson = data['categories'];  
 return categoriesJson.map((category) => category.toString()).toList();  
 }else throw Exception('Failed to load categories');  
 }catch(e){  
 throw Exception('Failed to connect to API:$e');  
 }  
 }  
}

**BREAKDOWN**

This file defines a **service class (CoffeeApiService)** responsible for **communicating with a REST API** (most likely your local Node.js/Express server running on localhost:3001).  
It fetches coffee data (like list, single coffee, categories) using the **HTTP protocol**.

**🧩 1. Import Statements**

import 'dart:convert';

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

import '../models/coffee\_model.dart';

**Explanation:**

* dart:convert → A built-in Dart library used for **encoding and decoding JSON** data.
  + Example: json.decode() converts a JSON string into a Dart Map.
* package:http/http.dart → A popular package for making **HTTP requests** (GET, POST, PUT, DELETE, etc.) to APIs.
  + The as http part allows you to prefix its functions (like http.get) for clarity.
* ../models/coffee\_model.dart → Imports your **Coffee model class**, which defines how coffee data is structured in your app.  
  This class likely contains a factory constructor like:
* factory Coffee.fromJson(Map<String, dynamic> json) => Coffee(
* id: json['id'],
* name: json['name'],
* price: json['price'],
* );

**🏗️ 2. Class Definition**

class CoffeeApiService {

static const String baseUrl = 'http://localhost:3001/api';

**Explanation:**

* This is a **service class** — it contains all functions related to Coffee API calls.
* static → Means you can call its methods directly using the class name (without creating an object):
* CoffeeApiService.getAllCoffees();
* baseUrl → The root URL of your backend API.  
  All endpoints (like /coffees, /categories) are appended to this.

**☕ 3. Get All Coffees**

static Future<List<Coffee>> getAllCoffees() async {

try {

final response = await http.get(Uri.parse('$baseUrl/coffees'));

**Step-by-step:**

1. **Future<List<Coffee>>** → The function runs asynchronously and returns a list of Coffee objects in the future.
2. **await http.get()** → Sends an HTTP GET request to http://localhost:3001/api/coffees.
3. **Uri.parse()** → Converts the URL string into a URI object (required by the http library).

if (response.statusCode == 200) {

final Map<String, dynamic> data = json.decode(response.body);

final List<dynamic> coffeesJson = data['coffees'];

return coffeesJson.map((json) => Coffee.fromJson(json)).toList();

} else {

throw Exception('Failed to load coffees');

}

**Explanation:**

* response.statusCode == 200 → Success! The server responded OK.
* json.decode(response.body) → Converts JSON response into a Dart Map.
* data['coffees'] → Accesses the “coffees” key in the JSON (assuming response looks like { "coffees": [...] }).
* .map((json) => Coffee.fromJson(json)) → Converts every coffee JSON object into a Coffee model.
* .toList() → Converts that mapped result into a List of Coffee objects.
* If the server didn’t respond with 200, it throws an exception.

} catch (e) {

throw Exception('Failed to connect to API: $e');

}

**Explanation:**  
If something goes wrong (like no internet or server down), the code goes into the catch block and throws an error.

**🔍 4. Get Coffee by ID**

static Future<Coffee> getCoffeeById(int id) async {

try {

final response = await http.get(Uri.parse('$baseUrl/coffees/$id'));

* Here, the URL becomes something like http://localhost:3001/api/coffees/3.
* It fetches a **single coffee** based on its id.

if (response.statusCode == 200) {

final Map<String, dynamic> coffeeJson = json.decode(response.body);

return Coffee.fromJson(coffeeJson);

} else {

throw Exception('Failed to load coffee');

}

* Converts JSON response into a single Coffee object.

**📂 5. Get Coffees by Category**

static Future<List<Coffee>> getCoffeesByCategory(String category) async {

try {

final response = await http.get(Uri.parse('$baseUrl/coffees/category/$category'));

* Example URL:  
  http://localhost:3001/api/coffees/category/Espresso
* It fetches all coffees under a specific category (like “Latte” or “Mocha”).

if (response.statusCode == 200) {

final Map<String, dynamic> data = json.decode(response.body);

final List<dynamic> coffeesJson = data['coffees'];

return coffeesJson.map((json) => Coffee.fromJson(json)).toList();

} else {

throw Exception('Failed to load coffees by category');

}

* Same logic as before — parse JSON, map to model list, and return.

**🏷️ 6. Get All Categories**

static Future<List<String>> getCategories() async {

try {

final response = await http.get(Uri.parse('$baseUrl/categories'));

* Fetches **a list of all available categories** from your backend API.
* Example response:
* { "categories": ["Espresso", "Latte", "Cappuccino"] }

if (response.statusCode == 200) {

final Map<String, dynamic> data = json.decode(response.body);

final List<dynamic> categoriesJson = data['categories'];

return categoriesJson.map((category) => category.toString()).toList();

} else {

throw Exception('Failed to load categories');

}

* Converts JSON array into a List<String> in Dart.

**🧠 Summary (Like a Teacher Recap)**

| **Method** | **Purpose** | **Return Type** | **API URL Example** |
| --- | --- | --- | --- |
| getAllCoffees() | Get all coffee items | List<Coffee> | /coffees |
| getCoffeeById(id) | Get coffee details by ID | Coffee | /coffees/3 |
| getCoffeesByCategory(category) | Filter coffees by category | List<Coffee> | /coffees/category/Espresso |
| getCategories() | Get all coffee categories | List<String> | /categories |

**💡 How You’d Use It in Your Flutter App:**

final coffees = await CoffeeApiService.getAllCoffees();

final espressoList = await CoffeeApiService.getCoffeesByCategory('Espresso');

final singleCoffee = await CoffeeApiService.getCoffeeById(2);

Each method returns **ready-to-use Dart objects**, so your UI can easily display the data.