

Working with REST APIs

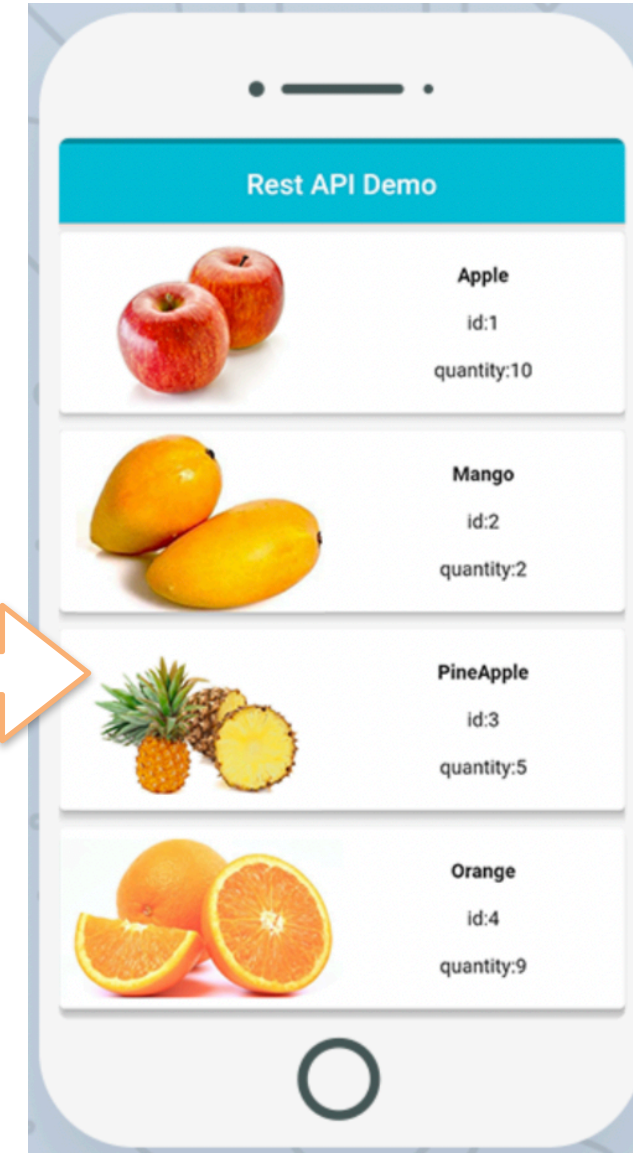
Contents

- Introduction
- JSON overview
- REST API overview
- API Testing
- Implementing Rest API

Introduction

- Data, data and data. Everywhere we see in today's world is just data. So how can we get data in our app?
- We have a lot of ways to show data in our app:
 1. From a database
 2. From public APIs

```
{
  "id": 1,
  "title": "Apple",
  "imageUrl": "https://produits.bienmanger.com/36433-0w470h470_Organic_Apples_Fuj.jpg",
  "quantity": 10
},
{
  "id": 2,
  "title": "Mango",
  "imageUrl": "https://images-na.ssl-images-amazon.com/images/I/31KSJIUe16L._SX450_.jpg",
  "quantity": 2
},
{
  "id": 3,
  "title": "PineApple",
  "imageUrl": "https://www.healthifyme.com/blog/wp-content/uploads/2015/12/shutterstock_1670265607-1.jpg",
  "quantity": 5
},
{
  "id": 4,
  "title": "Orange",
  "imageUrl": "https://i.ndtvimg.com/i/2016-08/orange_625x350_51471855916.jpg",
  "quantity": 9
}
```



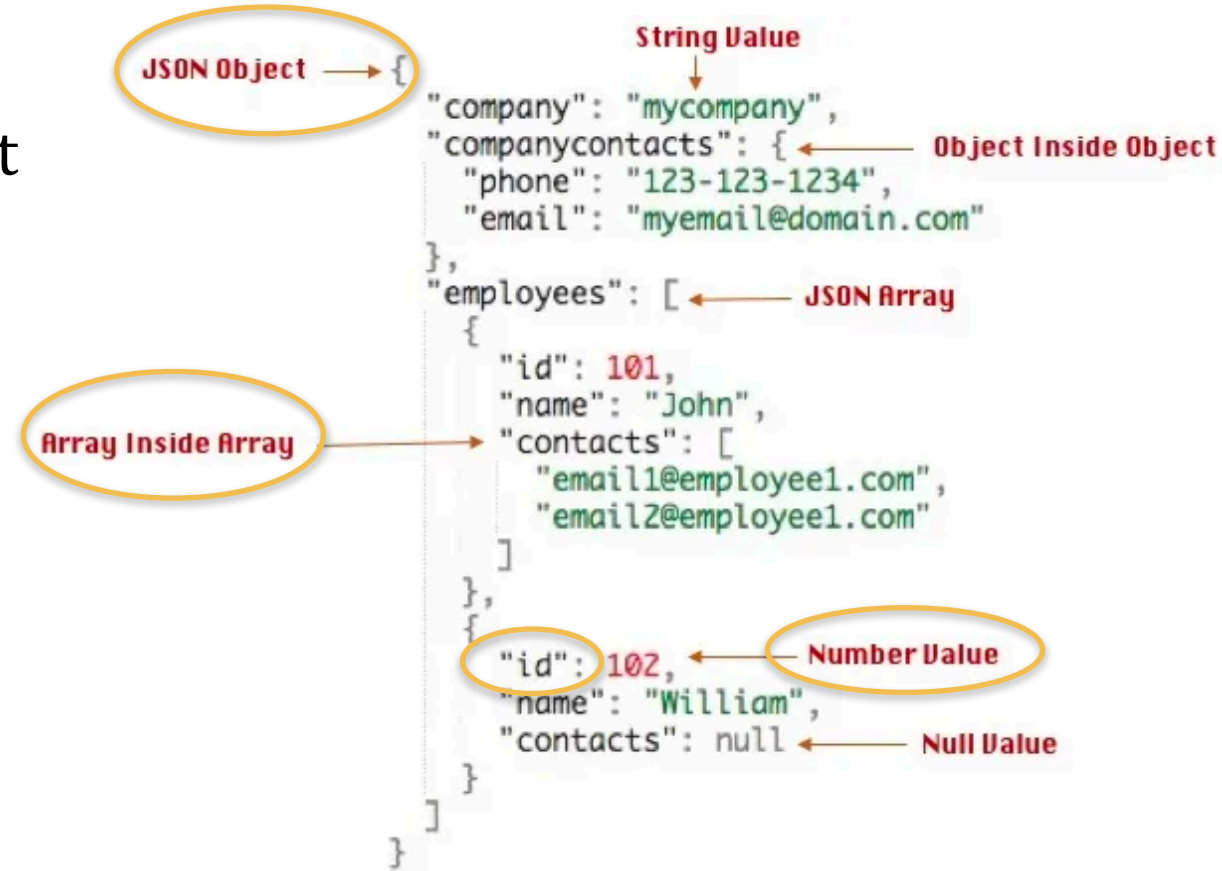
Working with APIs

- The most popularly used form is from Database or Public APIs.
- Let's see how we can integrate, fetch data from a API and use it in your flutter app!



JSON Overview (1)

- JSON represents data in two ways:
 - 1.Object: a collection of name-value (or key-value) pairs. An object is defined within left ({} and right {}) braces. Each name-value pair begins with the name, followed by a colon, followed by the value. Name-value pairs are comma separated.
 - 2.Array: an ordered collection of values. An array is defined within left ([]) and right (]) brackets. Items in the array are comma separated.












JSON Overview (2)

- This is an example of a JSON file that data for trainees resource.

```
[
  {
    "name": "lam",
    "email": "lam@gmail.com",
    "phone": "0974111444",
    "gender": "Female",
    "id": "2"
  },
  {
    "name": "khanh",
    "email": "khanh@gmail.com",
    "phone": "0123456789",
    "gender": "1",
    "id": "5"
  },
  {
    "name": "Le Hong Dang",
    "email": "giasutinhoc.vn@gmail.com",
    "phone": "123",
    "gender": "Male",
    "id": "17"
  }
]
```

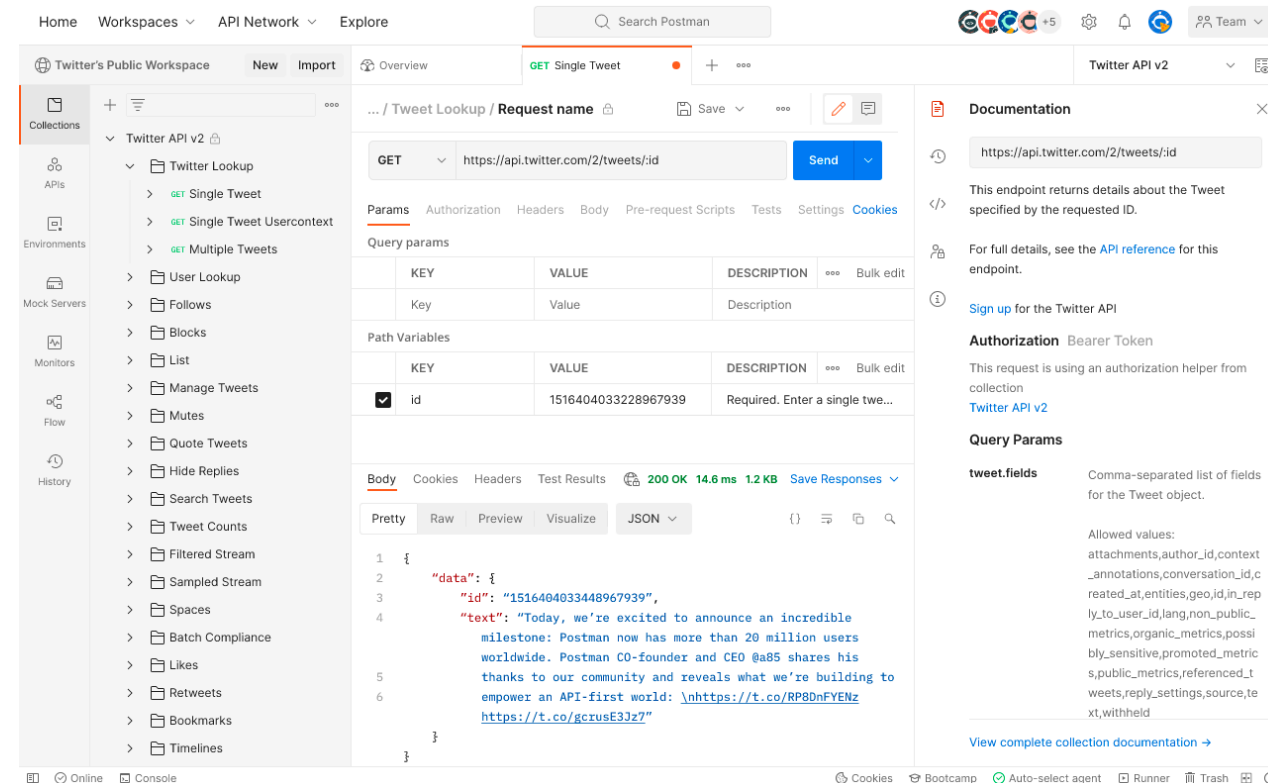
REST API Overview

- API (Application Programming Interface)
- For example, the Task REST API below

Task		Method	Path
Create a new task			/tasks
Delete an existing task			/tasks/{id}
Get a specific task			/tasks/{id}
Search for tasks			/tasks
Update an existing task			/tasks/{id}

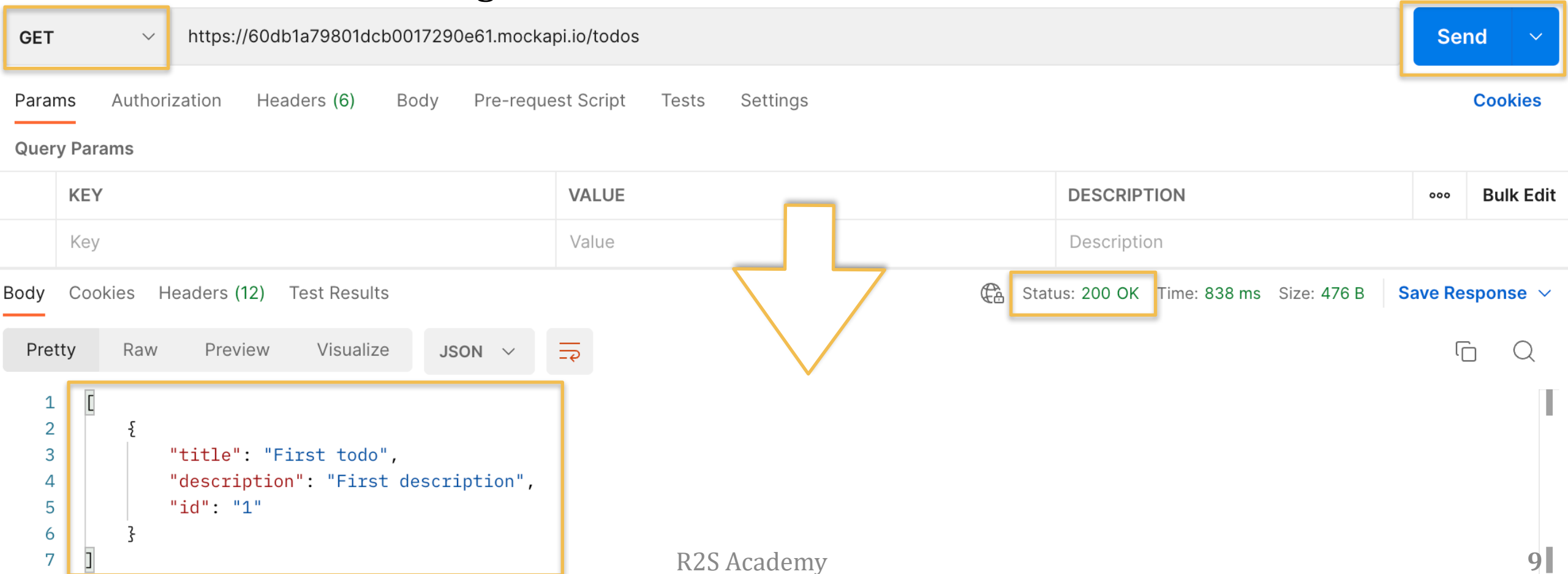
API Testing using Postman

- The Postman app: <https://www.postman.com/downloads/>
- How to use for API Testing
 1. Working with GET requests
 2. Working with POST requests
 3. Working with PUT requests
 4. Working with DELETE requests



Working with GET Requests (1)

- **Get** requests are used to retrieve information from the given URL. There will be no changes done to the endpoint.
- We will use the following **URL** `https://60db1a79801dcb0017290e61.mockapi.io/todos`



The screenshot shows a REST client interface with the following components:

- Request Bar:** Method `GET` and URL `https://60db1a79801dcb0017290e61.mockapi.io/todos`. A `Send` button is on the right.
- Tabs:** Params, Authorization, Headers (6), Body, Pre-request Script, Tests, Settings. A `Cookies` link is also present.
- Query Params Table:**

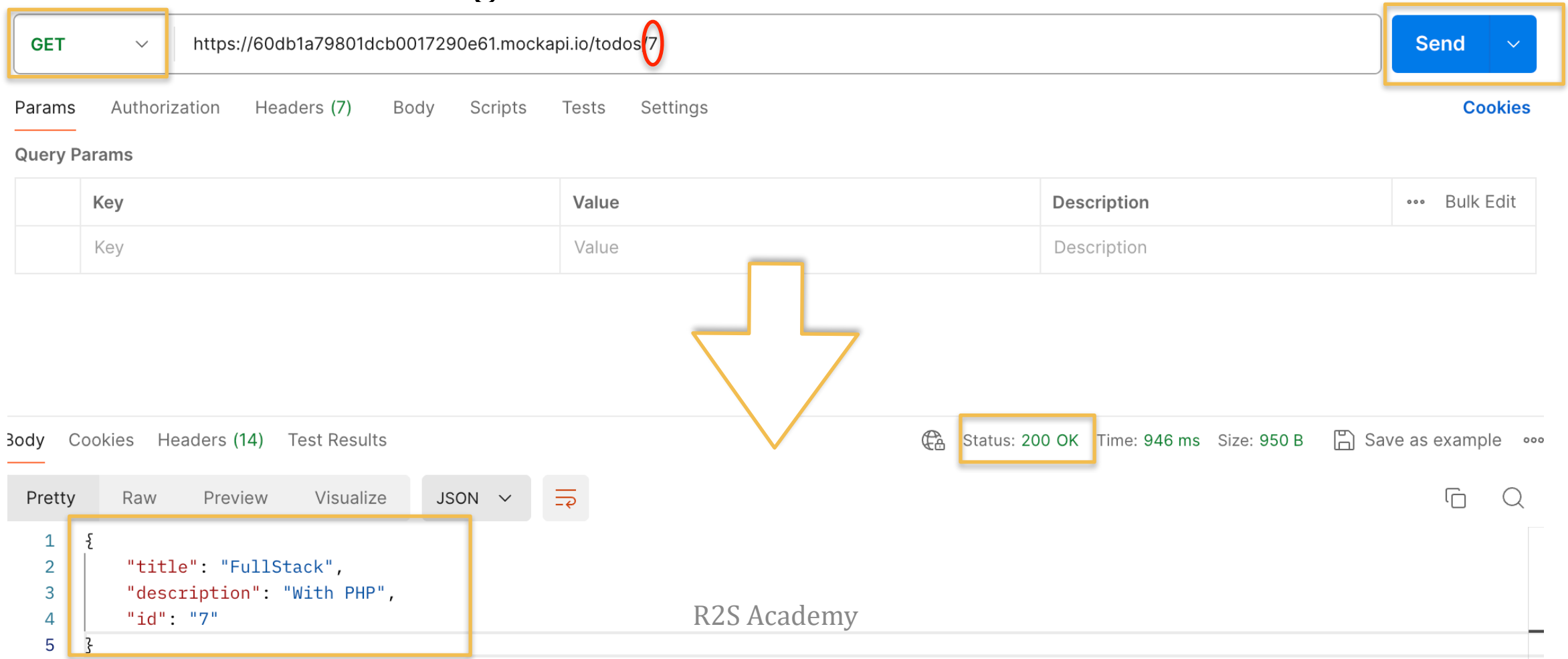
KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		
- Response Section:**
 - Tabs:** Body, Cookies, Headers (12), Test Results.
 - Format:** Pretty, Raw, Preview, Visualize. A dropdown menu is set to `JSON`.
 - Status:** `Status: 200 OK` (highlighted with an orange box).
 - Metadata:** Time: 838 ms, Size: 476 B.
 - Action:** `Save Response` with a dropdown arrow.
- Response Body:** A JSON object (highlighted with an orange box):

```
{  "title": "First todo",  "description": "First description",  "id": "1"}
```

A large orange arrow points from the URL bar area down towards the response body.

Working with GET Requests (2)

- **Get** requests are used to retrieve information from the given URL. There will be no changes done to the endpoint.
- We will use the following **URL** <https://60db1a79801dcb0017290e61.mockapi.io/todos/7>



The screenshot shows a REST client interface. At the top, a dropdown menu is set to 'GET' and the URL 'https://60db1a79801dcb0017290e61.mockapi.io/todos/7' is entered. A 'Send' button is to the right. Below the URL bar are tabs for 'Params', 'Authorization', 'Headers (7)', 'Body', 'Scripts', 'Tests', and 'Settings'. The 'Params' tab is active, showing a table with columns 'Key', 'Value', and 'Description'. Below this is a section for 'Query Params'. A large orange arrow points from the URL bar to the 'Status: 200 OK' box. The 'Body' tab is active, showing a JSON response in 'Pretty' format. The response is a JSON object with 'title', 'description', and 'id' fields. The status bar at the bottom shows 'Status: 200 OK', 'Time: 946 ms', and 'Size: 950 B'. There are also buttons for 'Save as example' and a search icon.

GET ▼ https://60db1a79801dcb0017290e61.mockapi.io/todos/7 Send ▼

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (14) Test Results

Status: 200 OK Time: 946 ms Size: 950 B Save as example ...

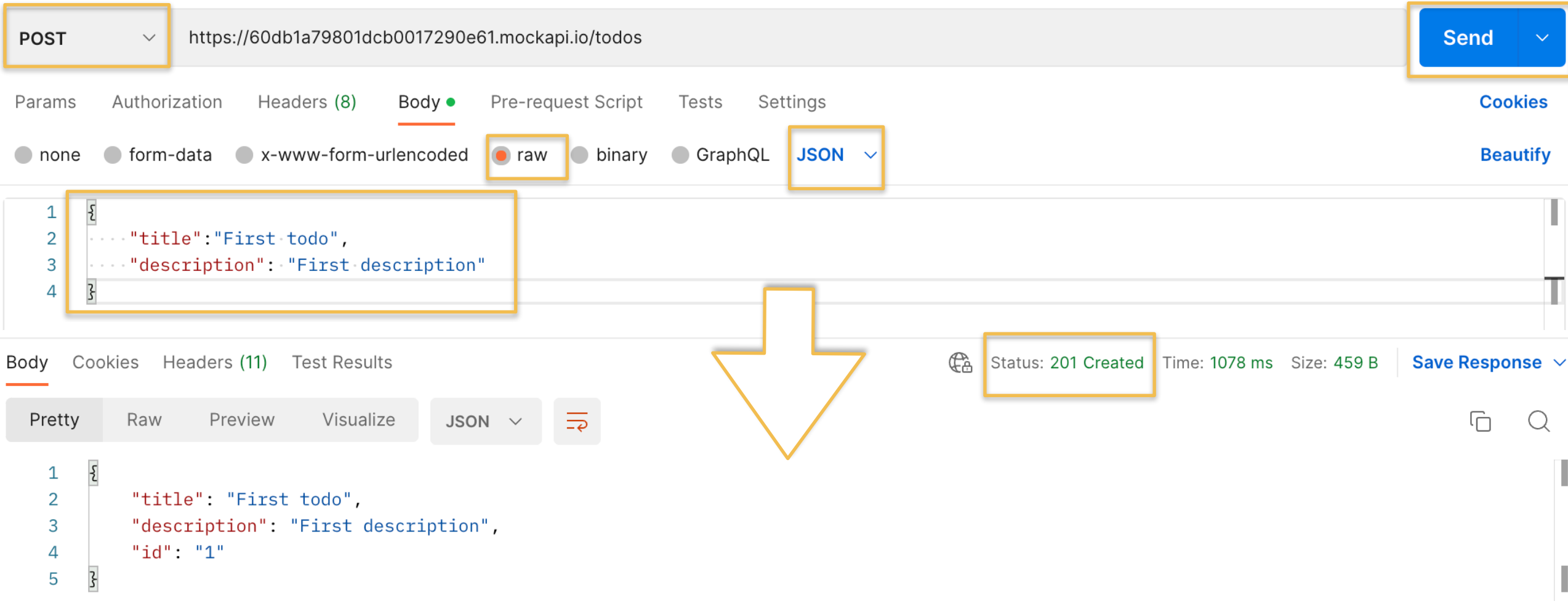
Pretty Raw Preview Visualize JSON ▼ ≡

```
1 {
2   "title": "FullStack",
3   "description": "With PHP",
4   "id": "7"
5 }
```

R2S Academy

Working with POST Requests

- **Post** requests are different from Get request as there is data manipulation with the user adding data to the endpoint.
- We will use the following **URL** `https://60db1a79801dcb0017290e61.mockapi.io/todos`



The screenshot displays a REST client interface with the following components:

- Request Method:** A dropdown menu set to **POST**.
- URL:** `https://60db1a79801dcb0017290e61.mockapi.io/todos`
- Body Tab:** The **Body** tab is selected, showing a JSON payload:

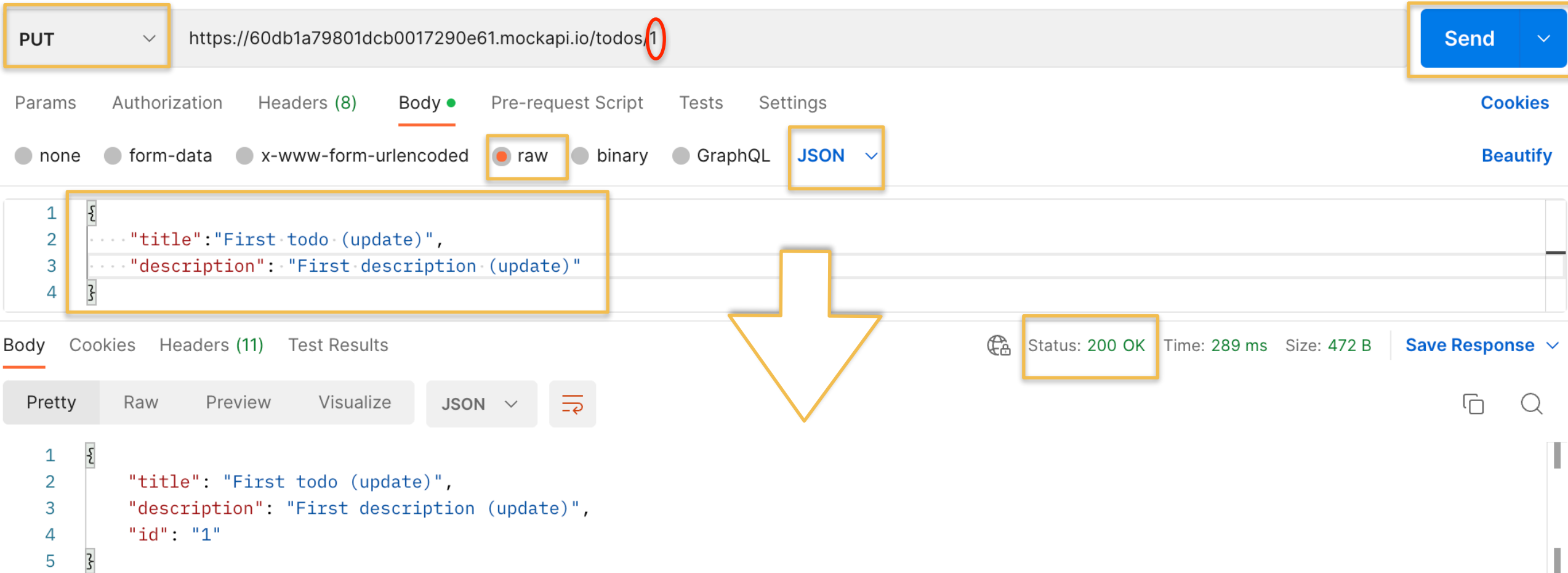
```
{
  "title": "First todo",
  "description": "First description"
}
```
- Format:** A dropdown menu set to **JSON**.
- Send Button:** A blue button labeled **Send**.
- Response Section:** The **Body** tab is selected, showing the response:

```
{
  "title": "First todo",
  "description": "First description",
  "id": "1"
}
```
- Status:** A box indicating **Status: 201 Created**.
- Time and Size:** **Time: 1078 ms** and **Size: 459 B**.
- Save Response:** A blue button labeled **Save Response**.

A large yellow arrow points from the request body to the response body.

Working with PUT Requests

- **PUT** is used to send data to a server to update a resource.
- We will use the following **URL** `https://60db1a79801dcb0017290e61.mockapi.io/todos/1`



The screenshot shows a REST client interface with the following components:

- Method:** A dropdown menu set to **PUT**.
- URL:** `https://60db1a79801dcb0017290e61.mockapi.io/todos/1`. The `1` at the end is circled in red.
- Send Button:** A blue button labeled **Send** with a dropdown arrow.
- Body Tab:** The **Body** tab is selected, showing a JSON payload:

```
{
  "title": "First todo (update)",
  "description": "First description (update)"
}
```

. The **raw** radio button is selected, and the **JSON** format is chosen from a dropdown.
- Response:** The **Body** tab shows the response:

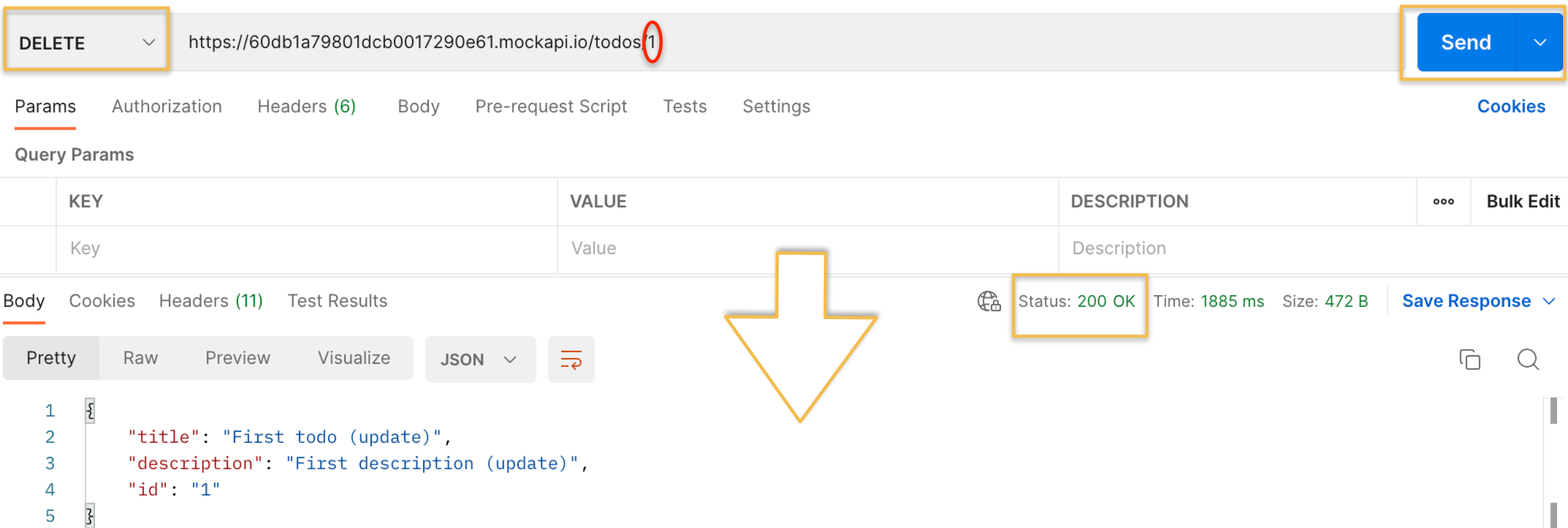
```
{
  "title": "First todo (update)",
  "description": "First description (update)",
  "id": "1"
}
```

. The **JSON** format is selected for the response.
- Status Bar:** Shows **Status: 200 OK**, **Time: 289 ms**, and **Size: 472 B**. A **Save Response** button is also present.

A large yellow arrow points from the request body to the response body.

Working with DELETE Requests

- The **DELETE** method deletes the specified resource.
- We will use the following URL `https://60db1a79801dcb0017290e61.mockapi.io/todos/1`



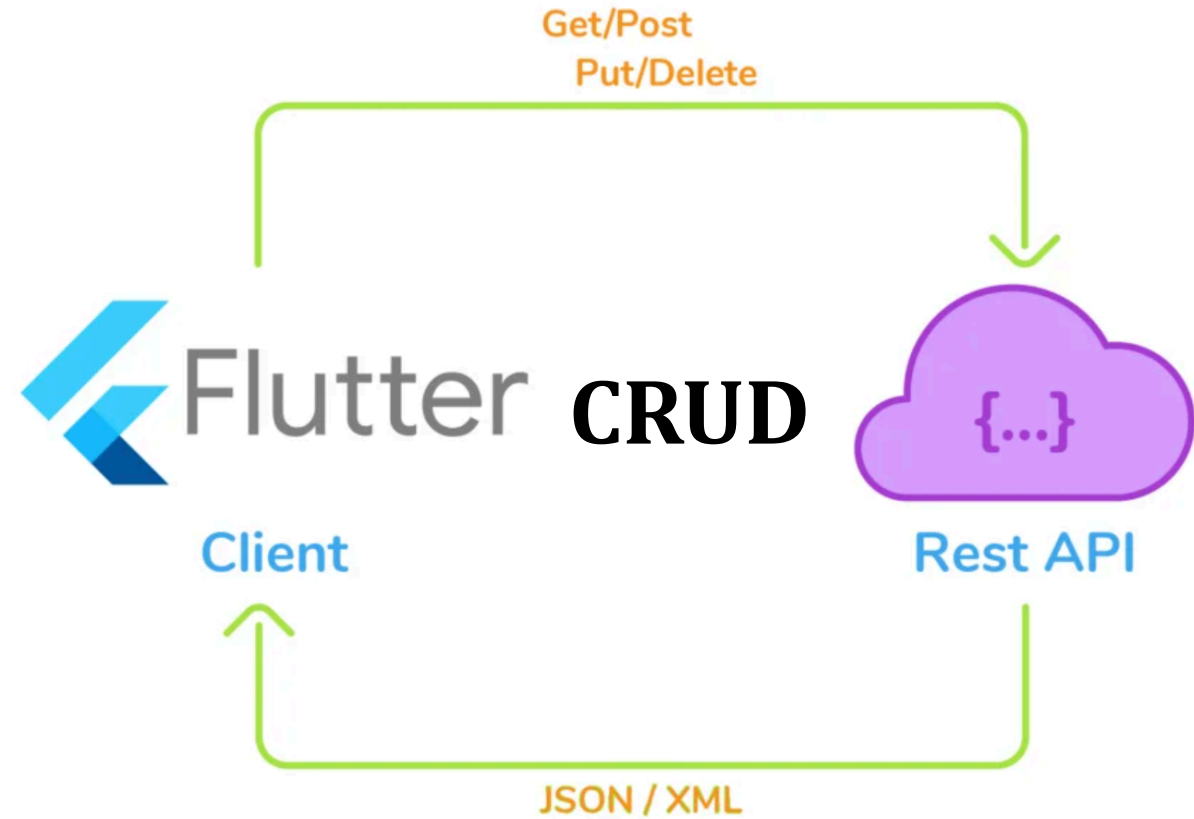
The screenshot shows a REST client interface with a DELETE request and its response. The request URL is `https://60db1a79801dcb0017290e61.mockapi.io/todos/1`. The response status is 200 OK, and the response body is a JSON object:

```
{
  "title": "First todo (update)",
  "description": "First description (update)",
  "id": "1"
}
```

A large yellow arrow points from the response status to the JSON body.

Implementing Rest API

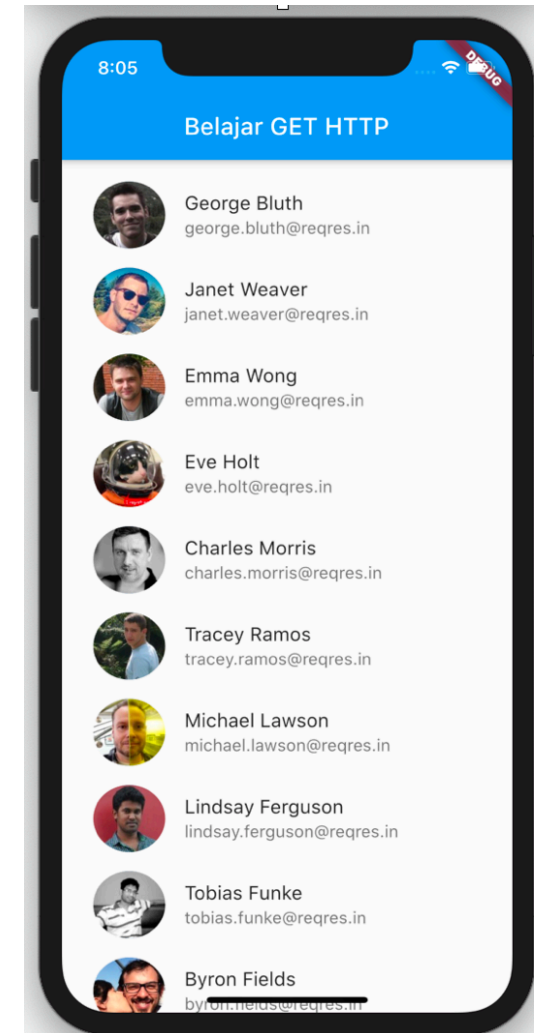
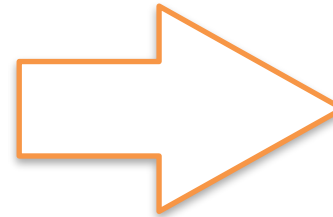
1. **Get** data from the internet
2. **Create** data to the internet ~ POST
3. **Update** data over the internet
4. **Delete** data on the internet



Get data (1)

1. Add the **http** package.
2. Make a network **request** using the http package.
3. Convert the **response** into a custom Dart object.
4. **Display** the data

```
"data": [  
  {  
    "id": 1,  
    "email": "george.bluth@reqres.in",  
    "first_name": "George",  
    "last_name": "Bluth",  
    "avatar": "https://reqres.in/img/faces/1-image.jpg"  
  },  
  {  
    "id": 2,  
    "email": "janet.weaver@reqres.in",  
    "first_name": "Janet",  
    "last_name": "Weaver",  
    "avatar": "https://reqres.in/img/faces/2-image.jpg"  
  },  
  {  
    "id": 3,  
    "email": "emma.wong@reqres.in",  
    "first_name": "Emma",  
    "last_name": "Wong",  
    "avatar": "https://reqres.in/img/faces/3-image.jpg"  
  },  
  {  
    "id": 4,  
    "email": "eve.holt@reqres.in",  
    "first_name": "Eve",  
    "last_name": "Holt",  
    "avatar": "https://reqres.in/img/faces/4-image.jpg"  
  }  
]
```



Get data (2)

- Add the http package
 1. Add it to the dependencies section of the pubspec.yaml file. You can find the latest version of the [http package](#) the pub.dev.

```
dependencies:  
  http: <latest_version>
```

2. Import the http package.

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

Get data (3)

```
Future<List<Todo>> _fetchTodos() async {  
    final baseUrl = Uri.parse('https://60db1a79801dcb0017290e61.mockapi.io/todos');  
  
    // Make a network request using the http.get() method.  
    final response = await http.get(baseUrl);  
  
    if (response.statusCode == 200) {  
        // Convert the response into a object  
        return _parseTodos(response.body);  
    }  
  
    throw Exception('Failed to load Todo');  
}
```

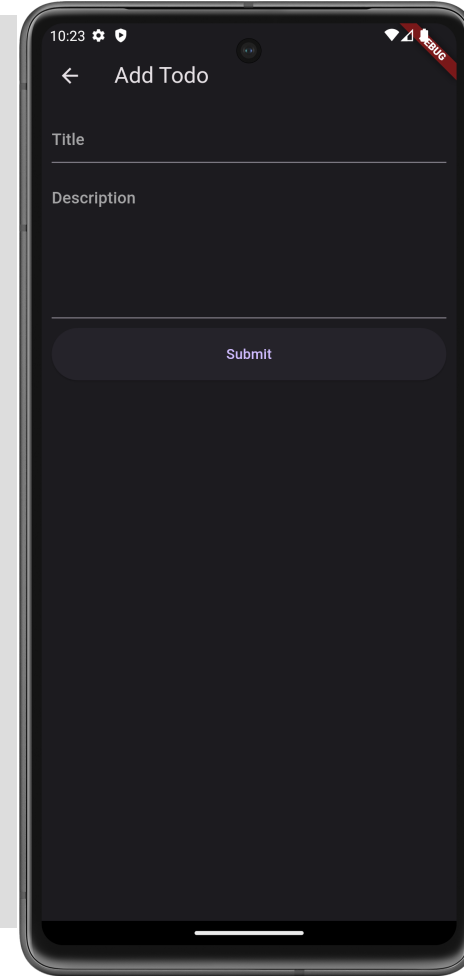
Get data (4) - Display the data

- **FutureBuilder** will help you to execute some **asynchronous** function and based on that **function's result** your **UI will update**.
- FutureBuilder - Parameters
 - **future**: assigns the Future **value** that will be delivered asynchronously
 - **builder**: returns a **Widget**
 - **snapshot**: will **hold the data** once it is delivered

```
FutureBuilder(  
  future: _futureTodos,  
  builder: (context, snapshot) {  
    if (snapshot.hasError) {  
      return Text('Retrieve Failed ${snapshot.error}');  
    } else if (snapshot.hasData) {  
      final List<Todo> todos = snapshot.data!;  
      return ListView.builder(  
        itemCount: todos.length,  
        itemBuilder: (context, index) => Card(  
          child: ListTile(  
            title: Text('${todos[index].title}'),  
            subtitle: Text('${todos[index].description}'),  
          )),  
      ) else {  
        return const CircularProgressIndicator();  
      }  
    },  
  ),  
)
```

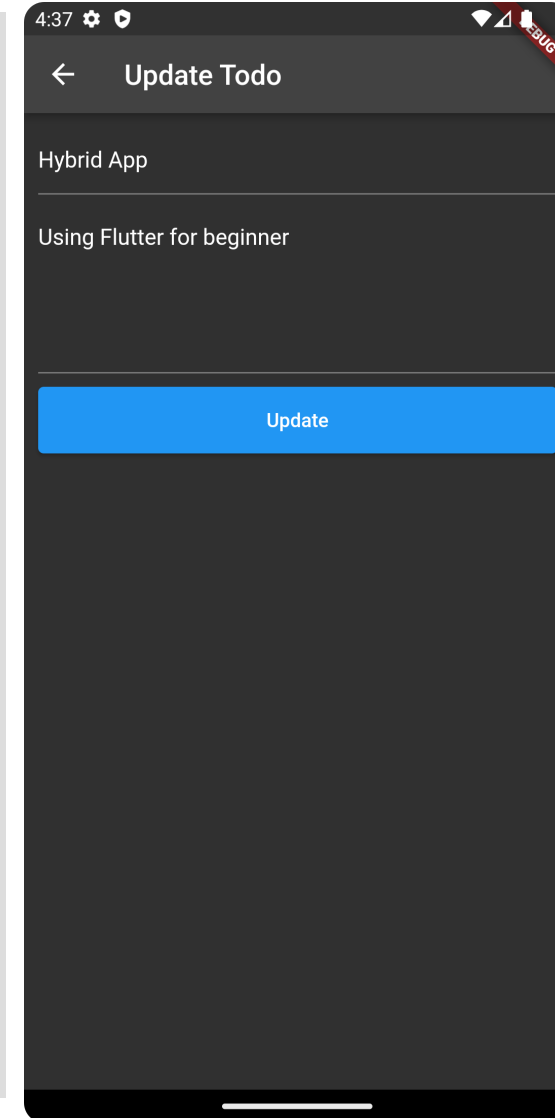
Create data

```
Future<void> submit(Todo todo) async {  
    // Get the data from form  
    final body = {'title': todo.title, "description": toto.description};  
  
    // Submit data to the server  
    final baseUrl = "https://60db1a79801dcb0017290e61.mockapi.io/todos";  
    final uri = Uri.parse(baseUrl);  
    final response = await http.post(uri,  
        headers: <String, String>{  
            'Content-Type': 'application/json; charset=UTF-8',  
        },  
        body: jsonEncode(body));  
}
```



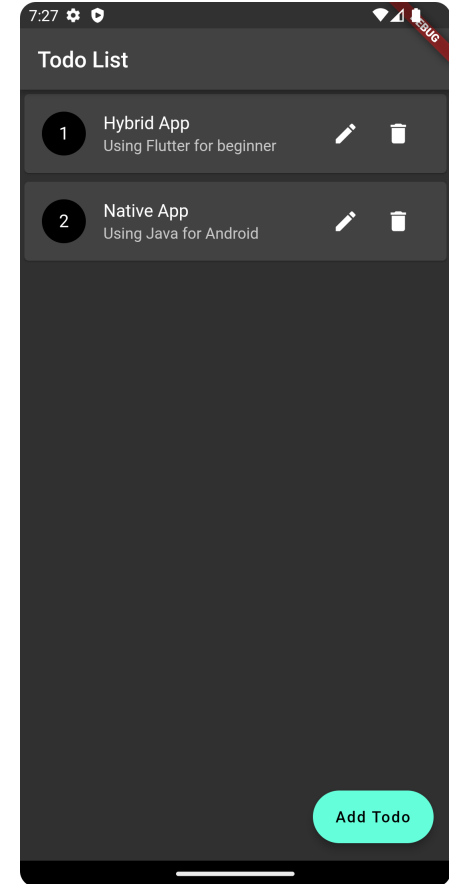
Update data

```
Future<void> update(Todo todo) async {  
  final id = todo!.id!;  
  final url = "https://60db1a79801dcb0017290e61.mockapi.io/todos/$id";  
  final uri = Uri.parse(url);  
  
  final body = {  
    "title": todo.title,  
    "description": todo.description  
  };  
  
  final response = await http.put(uri,  
    headers: <String, String>{  
      'Content-Type': 'application/json; charset=UTF-8',  
    },  
    body: jsonEncode(body));  
}
```



Delete data

```
Future<void> deleteById(String id) async {  
    // Delete the item  
    final url = "https://60db1a79801dcb0017290e61.mockapi.io/todos/$id";  
    final uri = Uri.parse(url);  
  
    final response = await http.delete(uri);  
}
```



*Keeping up those **inspiration** and the **enthusiasm** in the **learning path**.
Let confidence to bring it into **your career path** for getting gain the **success**
as your expectation.*

Thank you

Contact

- Name: R2S Academy
- Email: daotao@r2s.edu.vn
- Phone/Zalo: 0919 365 363
- FB: <https://www.facebook.com/r2s.tuyendung>
- Website: www.r2s.edu.vn

Questions and Answers