# Documenting a REST API

You have hopefully noticed that a great advantage of architectures that rely on REST is that client and server code can be developed independently. For this to work, however, we must have a clear and unambiguous description of the API, including methods, parameters, return values, and reaction to errors. With such a description, two parties should truly be able to develop the two “ends” of an application independently and eventually connect the two, in a successful way.

There are tools we can use for this, but generally, there is no “one correct way” of doing it.

In this document, we suggest a simple no-frills strategy that will be sufficient for everything you do this semester.

Use the examples below as a template for how you document your own API’s

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method** | **URL** | **Request Body (JSON)** | **Response (JSON)** | **Error (e)** |
| **GET** | /api/users |  | [user, user, ...](1) |  |
| **GET** | /api/users/{id} |  | user (1) | (e1) |
| **POST** | /api/users | user(1) without id |  | (e2) |
| **UPDATE** | /api/users/{id} | user(1) without id | user (1) |  |

#### Request Body and Response Formats

(1) User format (don’t provide ID, for POST)

{

"id": Number,  
 "age": Number,  
 "name": String,  
 "gender": String [“male” | “Female”],  
 "email": String (email)  
 }

#### Errors

(e) All errors are reported using this format (with the HTTP-status code matching the number)

{ status : statusCode, "msg": "Explains the problem" }

* (e1) : { status : 404, "msg": "No content found for this request" }
* (e2) : { status : 400, "msg": "Field ‘xxx’ is required" } (for example, no name provided)