**10 REST API BEST PRACTICES**



Photo by [Christopher Gower](https://unsplash.com/@cgower?utm_content=creditCopyText&utm_medium=referral&utm_source=unsplash) on [Unsplash](https://unsplash.com/photos/a-macbook-with-lines-of-code-on-its-screen-on-a-busy-desk-m_HRfLhgABo?utm_content=creditCopyText&utm_medium=referral&utm_source=unsplash)

**1. Use descriptive and meaningful resource names-**

*Instead of****generic****or****ambiguous****names, choose resource names that accurately represent the entities they represent.*

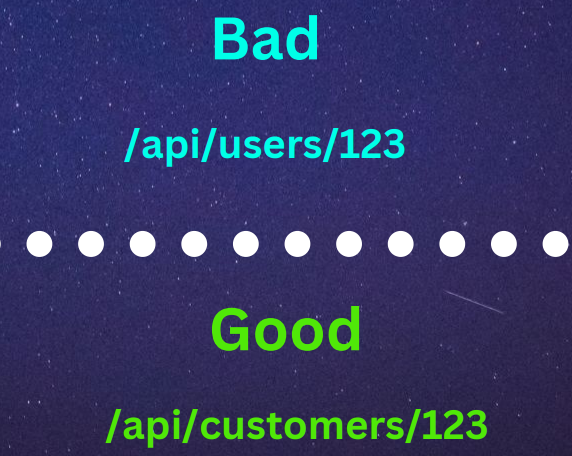


Image generated by the author

**2. Use HTTP methods correctly-**

*Use the appropriate****HTTP****methods (****GET, POST, PUT, DELETE, PATCH, etc.****) for different operations.*



Image generated by the author

**3. Version your APIs-**

*Use****versioning****to ensure backward compatibility and allow for future enhancements without breaking existing clients.*

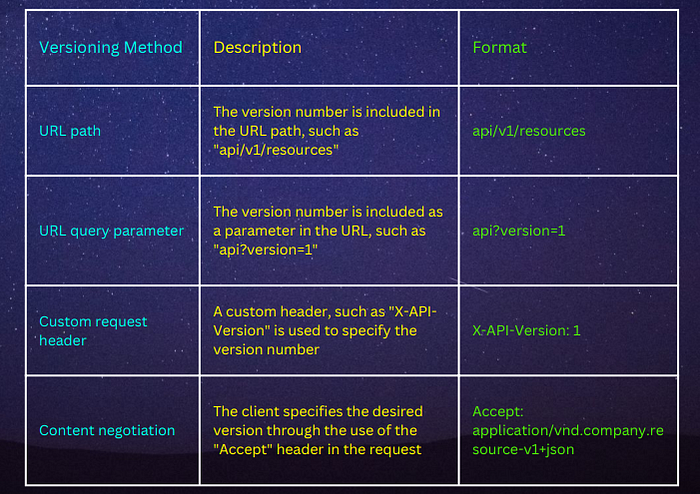


Image generated by the author

**4. Use HTTP status codes correctly-**

*Return the appropriate****HTTP status codes****to indicate the****success****or****failure****of an API request.*

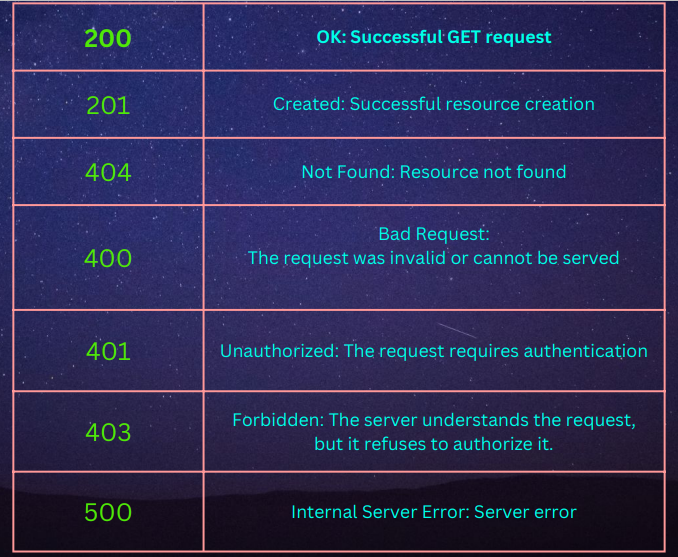


Image generated by the author

**5. Pick your JSON field naming convention (and stick to it)-**

***JSON****standard doesn’t impose a field naming convention, but it’s a best practice to pick one and stick with it.*

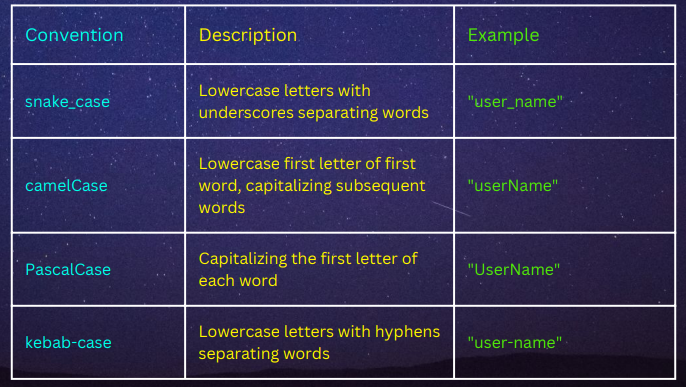


Image generated by the author

**6. Use consistent error messages-**

*In most cases,****HTTP status codes****are not enough to explain what went wrong.  
To help your API consumers, include a structured****JSON****error message.  
The response should include the following information-****Error code:****A machine-readable error code that identifies the specific error condition.****Error message:****A human-readable message that provides a detailed explanation of the error.****Error context:****Additional information related to the error, such as the request ID, the request parameters that caused the error, or the field(s) in the request that caused the error.****Error links:****URLs to resources or documentation that provide additional information about the error and how it can be resolved.****Timestamp:****The time when the error occurred.*

**7. Use query parameters for filtering, sorting, and searching-**

***Query parameters****allow you to provide additional information in the URL of an HTTP request to control the response returned by the server.*

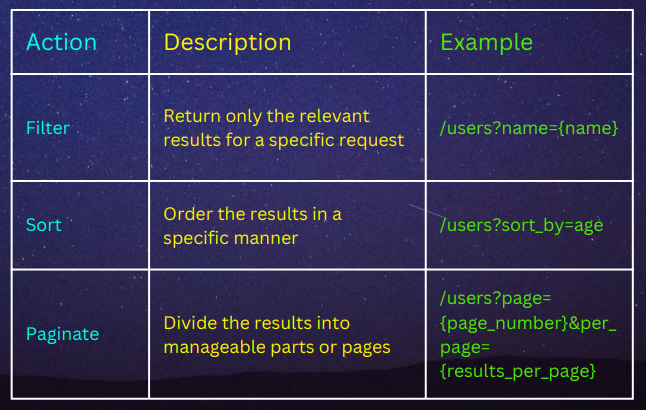


Image generated by the author

**8. Implement authentication and authorization-**

*Secure your APIs by implementing proper authentication and authorization mechanisms.*

***=>USE API KEYS, TOKENS, OR OAUTH 2.0 FOR AUTHENTICATION***

***=>APPLY ROLE-BASED ACCESS CONTROL (RBAC) FOR AUTHORIZATION***

**9. Do not maintain state-**

*A REST API should not maintain a state on the server. That’s the responsibility of the client.*

*This is important because it allows for the API to be****cacheable****,****scalable****, and****decoupled****from the client.*

*For example, an e-commerce API might use cookies to maintain the state of a shopping cart. However, such an approach violates key the key principle of RESTful APIs — they need to be stateless.*

**10. Document your APIs-**

*Provide comprehensive documentation for your APIs, including endpoint details, request/response examples, and usage guidelines .*

***=>SWAGGER/OPENAPI DOCUMENTATION***

***=>MARKDOWN-BASED DOCUMENTATION (E.G., USING TOOLS LIKE SWAGGER UI OR REDOC)***