Chapter 05 - Documenting Your API

Signal of Trust

- Documentation & features such as Software Development Kits (SDK) improve the Developer Experience (DX) for your Technical users.
- Consumers might Build an Integration: code that calls your code repeatedly.
- Data Scientist user might Schedule an Extract, Transform, Load (ETL) process
- Current release history + clear versioning strategy = Good Signals
- Questions that user might ask himself/you to decide if their going to use your product or not?
  - Can I count on you?
  - Will your API be available for me a month from now? What about a year from now?
  - Can I trust your data?
  - Do the definitions match the contents?
  - What level of quality is your data?

Making Great API Docs

Core Features

- Without these core features it is less likely for your API to get used by many users
- allow user to understand the value of API + get the address and necessary authorization information + easily test the API + Terms of service + Feedback mechanism $\xrightarrow{\text{lead to}}$ API get used appropriately + user know where to find help

Getting Started

- Overview/Introduction that explain the purpose of the API
- How consumer can interact with it.
- Provide the address of the API
- Security requirements
- Instruction for requesting users IDs or API keys

OAS file

- The OpenAPI Specification (OAS) file is a machine-readable file that defines:
    - The connection instructions
    - Endpoint definitions

Terms of Service

- Explain the allowed usage of the API:
    - Rate limits
    - Other restrictions

Feedback Mechanism

- Methods for user to reach out and ask questions:
    - Support email address
    - Contact forms
    - GitHub issu trackers

SDKs

- Make your user life so much easier by:
    - Enforce responsible usage of the API
    - Reducing some common problems such as overuse and etc.

Extra Features

Sample Program Code

- Example snippets of code in popular software language that demonstrate how to properly use the API

Interactive Documentation

- Allow users to submit sample API calls to a development environment to get hands-on experience using the API
- Swagger UI is used to generate interactive documentation for FastAPI projects.
- It support test API call, but it won't store the result

Sandbox Environment

- Beyond the Interactive Documentation, Good play grounds
- Enable developers to submit multiple API calls in an environment that saves the sample data.
- Sandbox environment keeps track of previous API calls to allow you to test multiple tasks together

Additional Features

- Creating a Postman collection
  - Which include example requests and tests that can be run with the Postman API testing tool.

> **This things that we discussed often need a dedicated developer relations (devrel) staff to handle. before investing resources think about the value that it provide for you, cause handling all these things will cost you a lot!**

> **Reducing the time to hello world (TTHW) is a way to make customers more likely to use your product!**

Fantasy League API Documentation Example

- Fantasy Sports API - Yahoo
- Fantasy Football: MFL Developers Program
- Sleeper API

Viewing Your API's Built-in Documentation

```
fastapi dev main.py
```

- This tells FastAPI to automatically reload the application each time you make any changes to the program code.

Swagger UI

- Generate Documentation by itself from OpenAPI Specification (OAS) file that is named `openapi.json` which is generated by FastAPI by itself from the `main.py` code

- Allow you to test your API (Redoc don't have this option)
- Add `/doc` to the end of your API to view the Swagger UI interactive API documentation
- `curl` section is a command-line statement that could be used to call this API.
  - Helpful to understand the exact HTTP request that is constructed by the parameters
  - cURL is a common command-line utility used to make HTTP requests to web applications and APIs
- Request URL section: URL without the HTTP verb
- Server Response section
- Response Body section: shows the JSON data that was returned by the API.
- Response Headers section: display HTTP headers, which are additional metadata that API sent along with the response body.
  - You would not see this if you were calling the API directly in your browser

Redoc

- Add `/redoc` to the end of your API URL to view it
- Advantage of Redoc over swagger is that, it have a better layout in mobile than swagger, which is not that important, but keep it in mind

> **Difference between Redoc and Swagger is that Redoc don't provide interactive option**

Working with Your OpenAPI Specification File

- OAS file is not just a simple way to generate documentation, it is a powerful API definition standard that allows many other tools to interact with your API.
  - Website list code generators
  - Data validators
  - SDK generators
  - Mock servers
  - etc.
- Add `/openapi.json` to the end of your API URL

> **Arazzo Specification is out there too!**

It can assist generative AI applications that are based on LLMs

4 main parts of OAS file:

1. `string`object, OpenAPI specification version
2. `Info` object, provides the metadata about the API.
   `title`, `version` = required fields
   Optional fields: `summary`, `description`, `termsOfService`, `contact`, `license`
3. List of `paths`, which are relative URLs for API endpoints.
   the HTTP verbs contain ed one level down inside the `path` object
   1. information that displayed in Swagger UI
   2. Operation identifier; not display in Swagger UI, can be used in generative AI applications
   3. Reference to schemas that are defined in `Components` section $\xrightarrow{\text{why?}}$ allow data structures to be defined a single time in the OAS and then get referenced in different parts of file
4. Can contain wide range of reusable items that can referenced in other parts of OAS.
   In this example it contain schemas, data structures used by the API.

Adding Details to the OAS info Object

- Current info is to generic, need to add content for user to understand your API better.
  How? by changing the `FastAPI()`constructor function!

1. API Descriptions:

- Add `api_description` variable contain `"""` multiline string
- Pass the `api_description` to application constructor, `description=`
- `title`
- `version`

2. Tags:

- `tag=[""]` to categorize things

3. More Details To Individual Endpoints:

- Add a summary: This summarizes the path and will be displayed on the operation title by Swagger UI.
- Add a description: This gives additional detail about the path and will be displayed below the tile by Swagger UI.
- Add a description to the 200 response: This replaces the default "Successful response" with a clearer description.
- Modify the `operationID`: This standardizes the `operationID` value, which will be used by a variety of tools that are using the OAS file.

4. Add Parameter Description:

- replaces the default values of the parameters with the `Query()` statement and a description

5. Regression-Testing Your API

Running your existing tests to make sure you didn't break anything.

Additional Resources:

- Chapter 7 of Principles of Web API Design
- Designing Web APIs
- NordicAPIs blog
  - Developer Experience Subject
  - API Documentation Subject
- Developer Experience: The Metrics That Matter Most | Moesif Blog
- Using ESPN's new Fantasy API (v3) | Steven Morse
- GitHub - joeyagreco/leeger: Instant stats for any fantasy football league.
- Installation | Swagger Docs
- OpenAPI Reference | Speakeasy
- OpenAPI Specification v3.1.0
- OpenAPI.Tools - an Open Source list of great tools for OpenAPI.
- GitHub - Redocly/redoc: 📘 OpenAPI/Swagger-generated API Reference Documentation

- API Documentation & Design Tools for Teams | Swagger