How to document REST APIs

Python Ecosystem

https://twitter.com/jorlugaqui
https://medium.com/@jorlugaqui
https://www.linkedin.com/in/jorlugaqui/

What I have noticed

Too much code, too few docs.

What I have noticed

We got lazy at writing and reading specifications.

What I have noticed

Who said that Agile means no documentation?

What I have noticed

“Time to market” is the evil for docs.

When there are not docs…

We are inflicting pain ourselves.

Remember the benefits.

“It force us to architect the solution first.”

Docs in Python Projects

Sphinx and Read the Docs:

https://docs.readthedocs.io/en/stable/intro/getting-started-with-sphinx.html

Docs in Python Projects

https://realpython.com/documenting-python-code/

OpenAPI Specification

https://www.openapis.org/

OpenAPI Specification

Previously known as Swagger Specification.

OpenAPI Specification

It is a standard that describes, produces, consumes and, visualizes RESTFul APIs web services.

OpenAPI Specification

Code and documentation keep sync.

Where can I check it?

- http://spec.openapis.org/oas/v3.0.3
- https://github.com/OAI/OpenAPI-Specification/
- https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.3.md
- https://github.com/OAI/OpenAPI-Specification/blob/master/IMPLEMENTATIONS.md#implementations

Swagger related tools

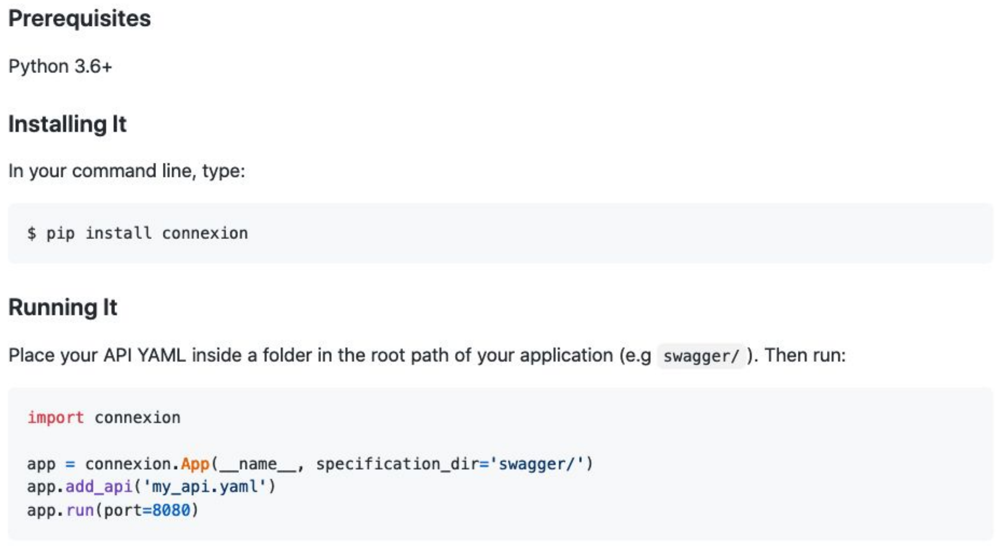
https://swagger.io/

https://swagger.io/tools/swagger-editor/

Python Ecosystem

- connexion
- flasgger
- Flask RESTX (previously Flask RestPlus)
- drf-yasg (Django)
- DRF schemas (Django)
- Fast API

connexion



Prerequisites

Python 3.6+

Installing It

In your command line, type:

```
$ pip install connexion
```

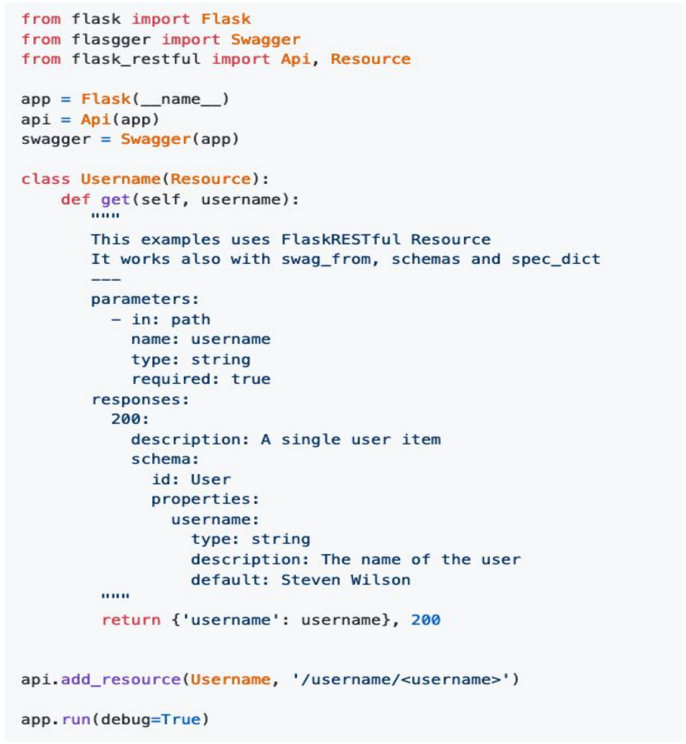
Running It

Place your API YAML inside a folder in the root path of your application (e.g `swagger/`). Then run:

```
import connexion

app = connexion.App(__name__, specification_dir='swagger/')
app.add_api('my_api.yaml')
app.run(port=8080)
```

https://medium.com/@hmajid2301/implementing-a-simple-rest-api-using-openapi-flask-connexions-1bdd01ca916

flasgger

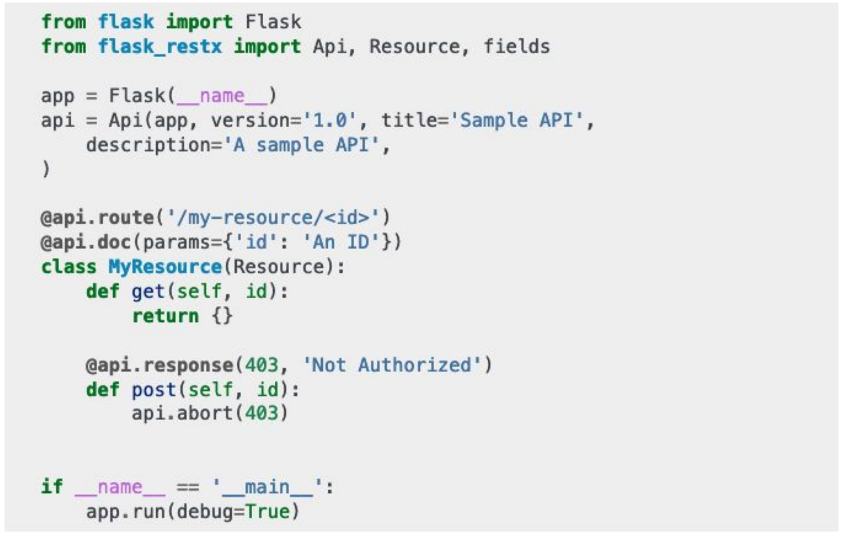


```python
from flask import Flask
from flasgger import Swagger
from flask_restful import Api, Resource

app = Flask(__name__)
api = Api(app)
swagger = Swagger(app)

class Username(Resource):
    def get(self, username):
       """
       This examples uses FlaskRESTful Resource
       It works also with swag_from, schemas and spec_dict
       ---
       parameters:
         - in: path
           name: username
           type: string
           required: true
       responses:
         200:
           description: A single user item
           schema:
             id: User
             properties:
               username:
                 type: string
                 description: The name of the user
                 default: Steven Wilson
        """
        return {'username': username}, 200


api.add_resource(Username, '/username/<username>')

app.run(debug=True)
```

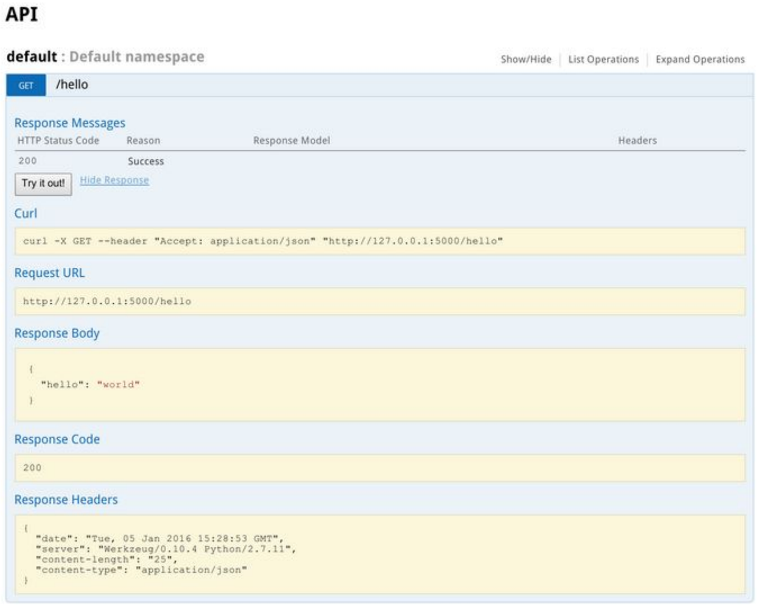
https://github.com/flasgger/flasgger

https://github.com/jorlugaqui/ahm-python

flask-restx



```python
from flask import Flask
from flask_restx import Api, Resource, fields

app = Flask(__name__)
api = Api(app, version='1.0', title='Sample API',
    description='A sample API',
)

@api.route('/my-resource/<id>')
@api.doc(params={'id': 'An ID'})
class MyResource(Resource):
    def get(self, id):
        return {}

    @api.response(403, 'Not Authorized')
    def post(self, id):
        api.abort(403)


if __name__ == '__main__':
    app.run(debug=True)
```



Response Messages

HTTP Status Code	Reason	Response Model	Headers
200	Success		

Curl

```
curl -X GET --header "Accept: application/json" "http://127.0.0.1:5000/hello"
```

Request URL

```
http://127.0.0.1:5000/hello
```

Response Body

```
{
  "hello": "world"
}
```

Response Code

```
200
```

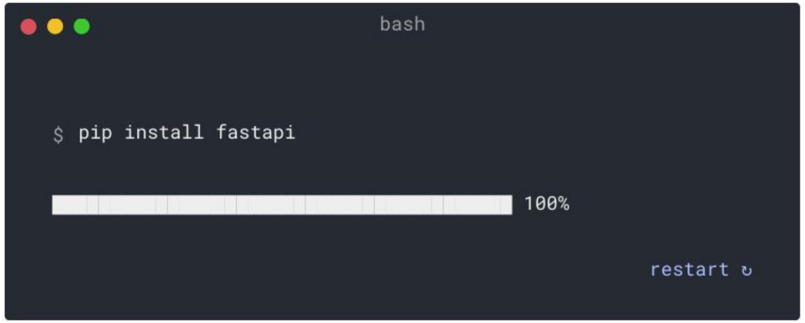
Response Headers

```
{
  "date": "Tue, 05 Jan 2016 15:28:53 GMT",
  "server": "Werkzeug/0.10.4 Python/2.7.11",
  "content-length": "25",
  "content-type": "application/json"
}
```

https://towardsdatascience.com/working-with-apis-using-flask-flask-restplus-and-swagger-ui-7cf447deda7f

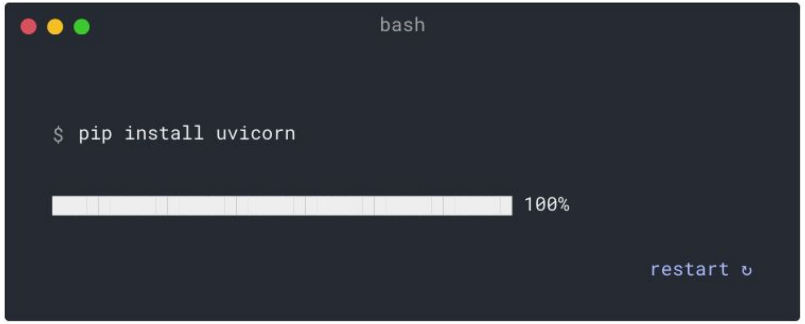
fast-api

Installation

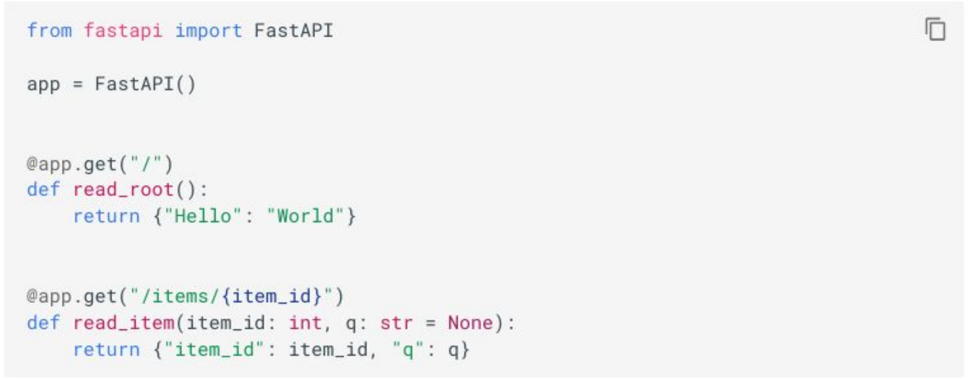


```
$ pip install fastapi

100%

restart ↻
```

You will also need an ASGI server, for production such as Uvicorn [↪] or Hypercorn [↪].



```
$ pip install uvicorn

100%

restart ↻
```



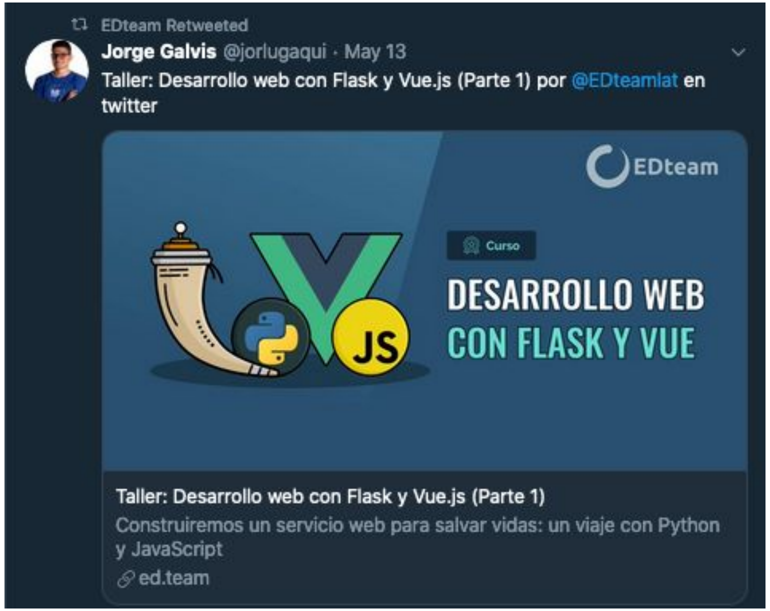
```python
from fastapi import FastAPI

app = FastAPI()


@app.get("/")
def read_root():
    return {"Hello": "World"}


@app.get("/items/{item_id}")
def read_item(item_id: int, q: str = None):
    return {"item_id": item_id, "q": q}
```

https://fastapi.tiangolo.com/

Conclusions

“Documentation is a love letter that you write to your future self.”
-- Damian Conway

I recorded a workshop



EDteam Retweeted

Jorge Galvis @jorlugaqui · May 13

Taller: Desarrollo web con Flask y Vue.js (Parte 1) por @EDteamlat en twitter

Taller: Desarrollo web con Flask y Vue.js (Parte 1)

Construiremos un servicio web para salvar vidas: un viaje con Python y JavaScript

ed.team

https://www.youtube.com/watch?v=JsPylo2sSz0&feature=emb_logo

https://ed.team/cursos/flask-js

How to document REST APIs

Thanks!