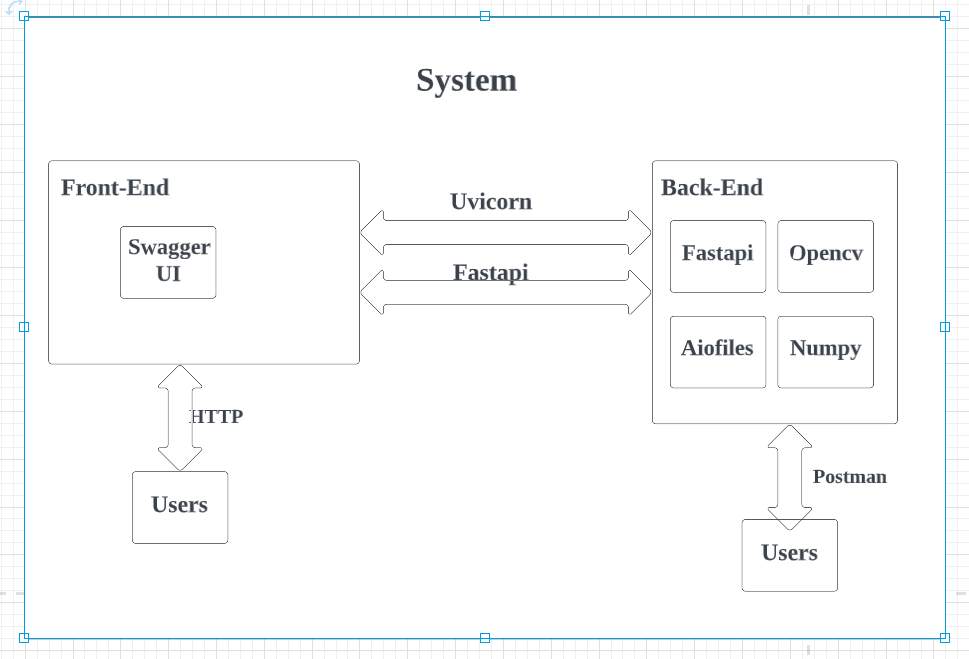
# Individual project

## Design component:

### Overall architecture:



Picture 1 System architecture

The system contains two user interface. The first one is Swagger UI. The second one is postman. Users can upload their images to manipulate via http url or postman get/post methods.

### Web UI specifications:

The system can let users manipulate their image by two ways. First, users can upload their image and the system will save the image at disk. Users will get a number of their image. They can choose the way they want and get the result. The organization of it can read from Picture 2. Users should upload their image and get the number of the upload picture. Then tell the system the number of the picture and the way they choose. System will return the result to users.

Another way is users send their image to system and system will return the result immediately. System will not save the image data. The logic should be like Picture 3. It looks like parallel method and won’t interact with each others.

### API specifications:

The fastapi framework combines the UI and API together. When we implemented the back-end logic, fastapi helps me make the swagger UI automatically and it can interact with postman. Same as user specifications. All the user interact way can happen in a same way in Postman. The example shows in later chapter.

### The endpoint descriptions:

The url rule: Postfix with methods name. Plus ? with parameter & input value.

For example: <http://127.0.0.1:8000/upload_file>

http://127.0.0.1:8000/Resize\_X\_Y?number=2&X=600&Y=500

http://127.0.0.1:8000/api\_Saturate?factor=0.5



Picture 2 One post image and get result by different get methods



Picture 3 Post image and get the result immediately

### Components and Connectors:

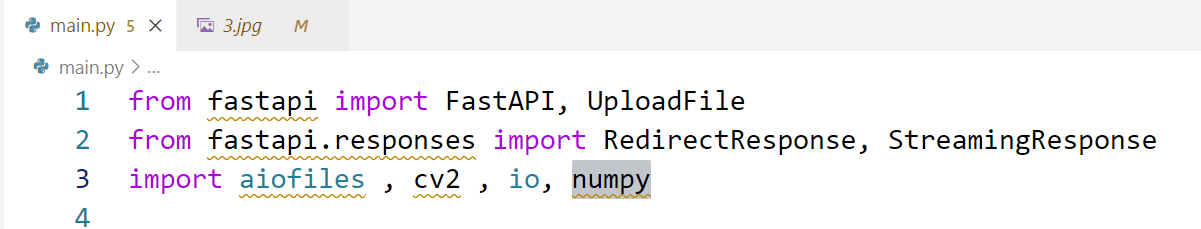
Picture 4 System Organization

### The organization is quite simple. Just a python file with all logic and run it. The FastAPI will read all the get/post methods in main.py.

## Code component:

FastAPI provides an automatically creating swagger UI when developers implement back-end logic. It is a little inconvenient for developers to change the front-end like use css or js class.

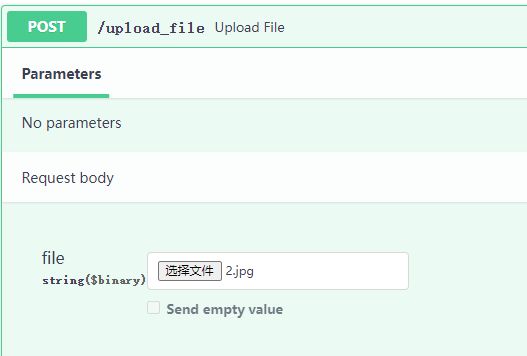
Toolkits:



Picture 5 Tools kit

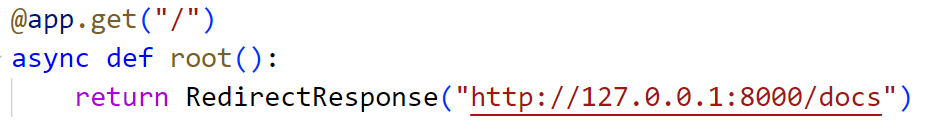
This is main tools I used to implement the project.

FastAPI is the framework I use.



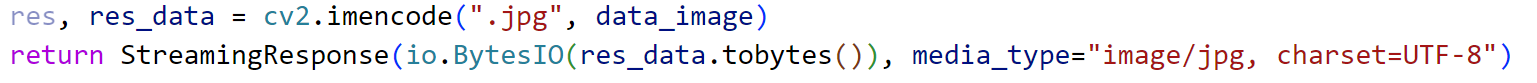
Picture 6 UploadFile

UploadFile is a tool within the FastAPI. Import it for upload file in web page.



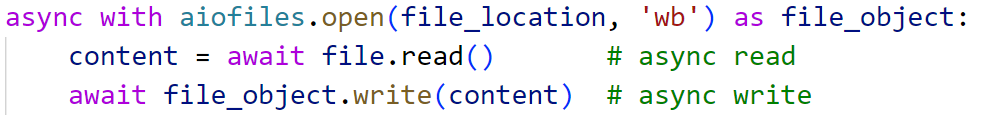
Picture 7 RedirectResponse to root page

RedirectResponse is a tool within the FastAPI. We can use it to redirect the endpoint.



Picture 8 StreamingResponse to return image data

StreamingResponse is a tool within the FastAPI. We use it for return image data to users. More specific, we need to combine with io to return the image bytes to users.

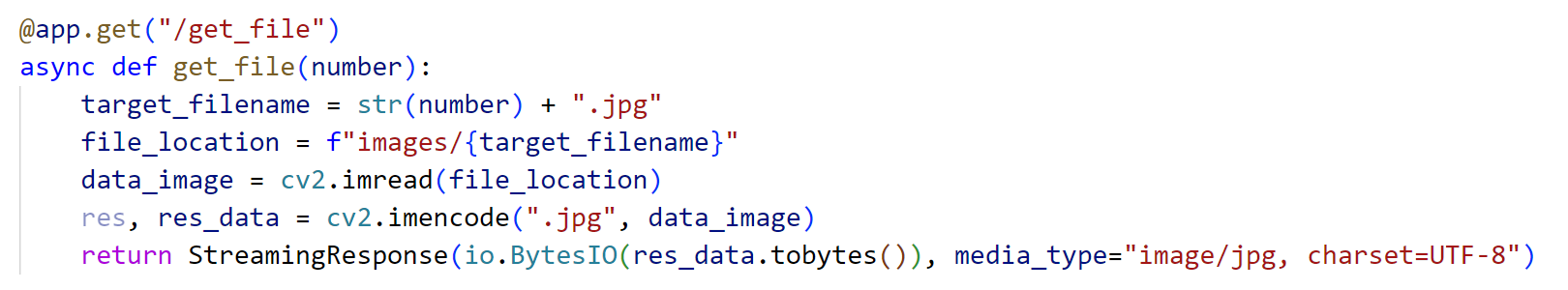


Picture 9 aiofiles read file and write file

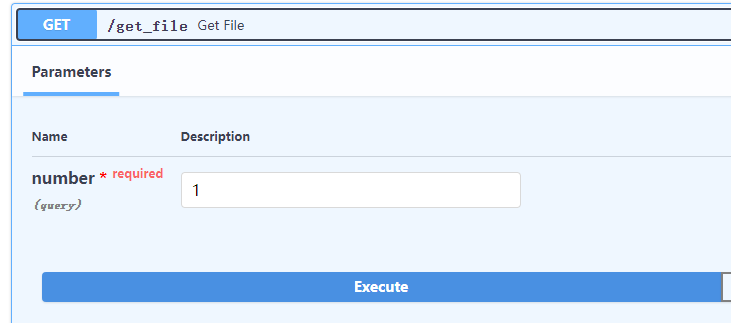
Aiofiles is an external tool for reading and writing image files. Use it for read the upload image and write the image into disk. The original methods of Read/Write are incompatible. It supports asynchronous ways of reading and writing files.

Cv2 is a commonly tool for modifying images.

### Get method example:



Picture 10 Get method example



Picture 11 Get method Web UI

FastAPI provides a combined way of swagger UI and API logic. We just need to declare there is a get method. It would generate a swagger UI page. When we click every different module, we would redirect to different url. Although they look the same as each other.

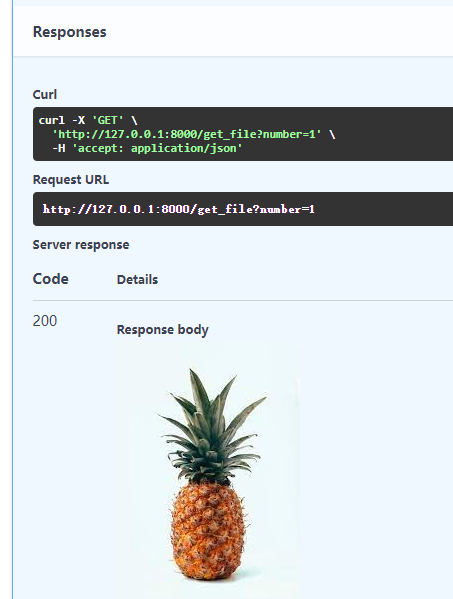
For example:

Get file url: <http://127.0.0.1:8000/docs#/default/get_file_get_file_get>

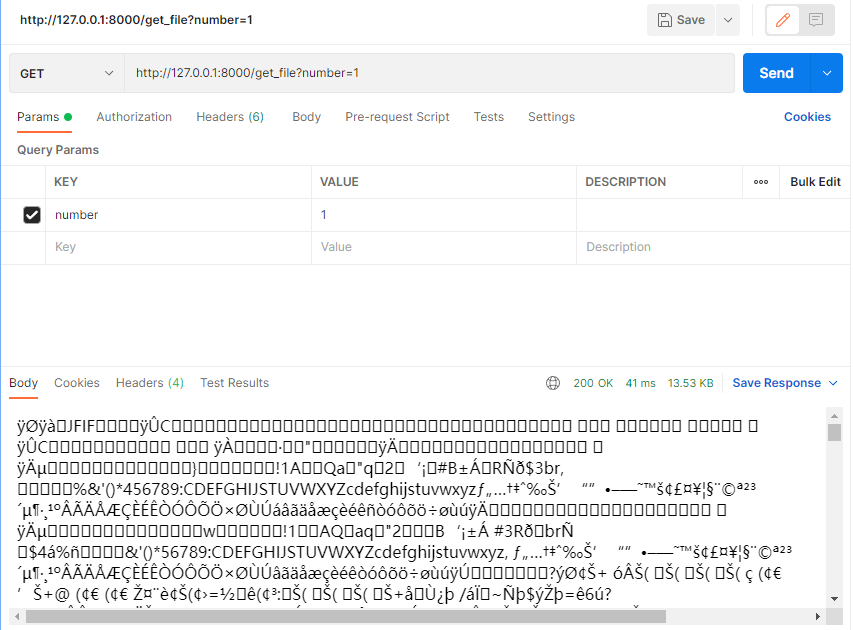
Image flip url: <http://127.0.0.1:8000/docs#/default/image_Flip_Flip_get>

When you click different module, it will jump into different url. Postman will get data from these different url.

Here are two accessing image ways. One is provide the image number and them the page would display the image data. Another ways is use Postman get method. The Postman would display bytes. Just save the data as .jpg and it will show the image.



Picture 12 Get method Web Response example

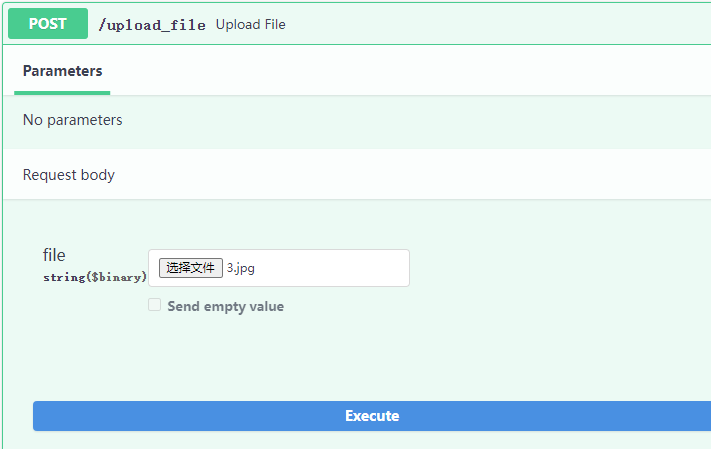


Picture 13 Get method Postman example

### Post method example:

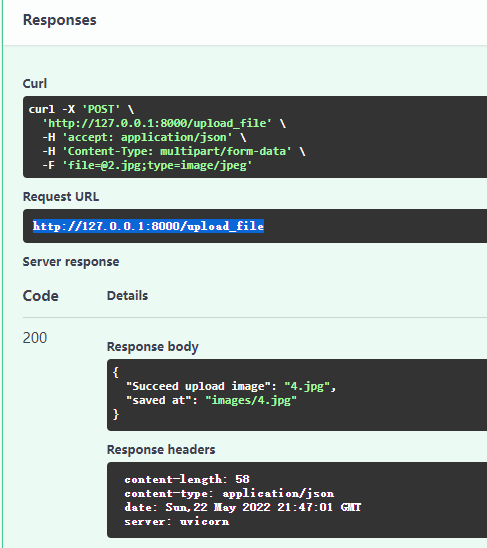


Picture 14 Post method example

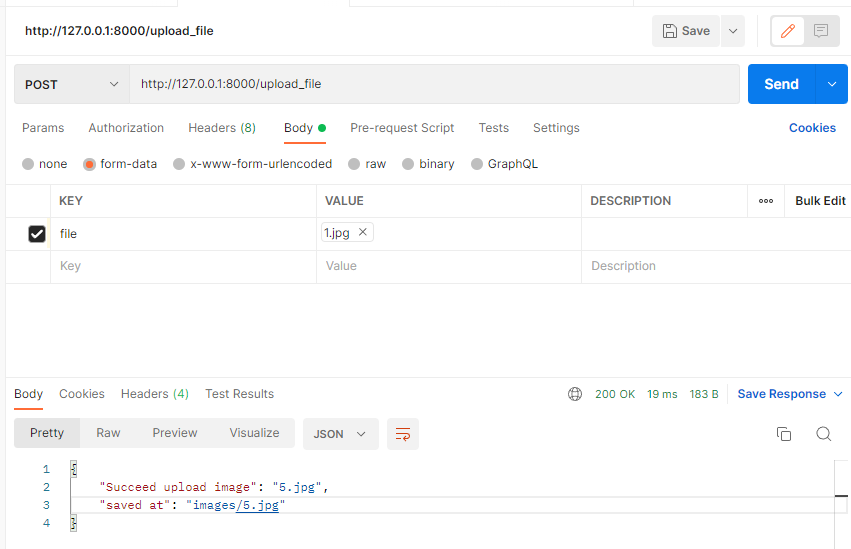


Picture 15 Post method Web UI

Same as get method. We can upload the image by web UI or postman. The post url shows in every module.



Picture 16 Post method Web Response example



Picture 17 Post method Postman example