ISSN: 2230-9926

Available online at http://www.journalijdr.com

IJDR

International Journal of Development Research
Vol. 10, Issue, 05, pp. 35551-35558, May, 2020
https://doi.org/10.37118/ijdr.18700.05.2020

RESEARCH ARTICLE OPEN ACCESS

DIGITAL IMAGE CONSUMPTION WITH REST API

[1]Deyveson Willian Gomes Rodrigues, [1]Leandro Oliveira Negreiros, [1]Bruno Pereira Gonçalves, [1]Manfrine Silva Santos, [1]Jean Mark Lobo de Oliveira and [2,*]David Barbosa de Alencar

[1]Academic Department, University Center FAMETRO, Amazon-Brazil
[2]Institute of Technology and Education Galileo of Amazon (ITEGAM), Brazil

ARTICLE INFO

Article History:
Received 02nd February, 2020
Received in revised form
06th March, 2020
Accepted 28th April, 2020
Published online 25th May, 2020

Key Words:
API Rest; Image; Pillow; Python; Streaming; Response Time.

Corresponding author:
David Barbosa de Alencar

ABSTRACT

The compression for sending images with REST API to provide a better response time. Applied to bibliographic research to understand the concepts included in the work, descriptive research to clarify the methods used and exploratory research to develop a REST API based on another product. With the above, it was possible to evaluate the performance of the REST API for the transmission of images. With the application of the image transmission tool they were able to decrease the transmission time by approximately 98.6%, before in 217 milliseconds that on a large scale would become long for the user. With the resources presented, cognitive research is relevant to express the improvement in sending and receiving images, leveraging education, human values and ethics.

Citation: ***Deyveson Willian Gomes Rodrigues, Leandro Oliveira Negreiros, Bruno Pereira Gonçalves, Manfrine Silva Santos, Jean Mark Lobo de Oliveira and David Barbosa de Alencar.*** **2020.** "Digital image consumption with rest api", *International Journal of Development Research*, 10, (05), 35551-35558.

INTRODUCTION

Large images reduce the speed of web pages, which provides a bad user experience. Optimizing images is reducing the file, so it can help improve the performance of an electronic website. A problem arises when formatting the images, which may reduce the quality (it may make the visitor not have a good user experience). There are techniques that allow you to reduce the size of the image file and keep them with good resolution. A REST API, it is documentation of established standards and routines, enabling applications to use these features, without knowing the software. APIs allow the exchange of information between applications. In other words, the communication between users and applications, to be consumed by front-end applications (client-side), and sending them via HTTP requests, improving the usability associated with the performance generated by the optimization in the general process of sending images, being low or high resolution without affecting the response time, providing the parties involved with satisfactory results in relation to the performance of the application. The alternative would be the implementation of a tool that reduces the size of the image file and thus transmits data between users and applications, through a technological tool improving the sending time, being able to transmit low or high resolution images without affecting the response time.

THEORETICAL REFERENCE

Docker, Container Manager: A standard software unit is based on a container that packages all its dependencies and the code and so that the application's tour is fast and reliable from one computing environment to another (SILVA, 2016). A Docker container image is a lightweight, executable, standalone software package that has everything needed for an application to run: system tools, libraries, code system settings, and runtime. In the case that the container images become runtime containers, the Docker Engine executes the images and makes them containers. It is available for Windows and Linux based applications, the container will always run the software in the same way, regardless of the infrastructure. the software is isolated from its environment by means of containers and its operation is guaranteed in a uniform manner, even though the differences, for example, between preparation and development.

Docker containers are run by the Docker Engine the containers are set to an industry standard so they can be portable in any environment. Containers share the kernel and do not require an operating system per application, increasing server efficiency and reducing server and licensing costs. Docker provides the industry's strongest standard isolation capabilities and applications are more secure. In the project it was used to facilitate the deployment of the application in any environment, whether in production or development.

Python, High Level Programming Language: Python was developed in the 1990s to succeed the ABC language. Python versions are also compatible with the GPL. It is an interpreted and general-purpose programming language, which allows systems to be integrated more effectively, its object-oriented, high-level approach with dynamic semantics (BANIN, 2018). Python's simplicity reduces the maintenance of a program by writing logical and clear code for large and small scale projects, supports modules, packages, modularization and code reuse. It is one of the languages that has grown the most due to its ability to support other languages and the compatibility operating on the main Windows and Linux operating systems. Programs like Instagram, Reddit and Dropbox are encoded in the Python language. Python is even the most popular language for data analysis and has conquered the scientific community because it supports structured programming, object-oriented coding, a large community to answer questions, a growing number of data analysis libraries and scalability. In the project used to write the business rule functions that include image compression and persistence in the database.

Flask, Framework Web Written in Python: It is a small framework a lightweight web application framework written in Python and based on the WSGI Werkzeug library and the Jinja2 library (FLASK, 2018). It started out as a simple wrapper around Werkzeug and Jinja and became one of the most popular Python Web application frameworks, because it maintains a simple but extensible core. However, it does not have a form validation, database abstraction layer or component where third party libraries exist to perform functionality. It is available under the terms of the BSD License. It was designed to facilitate and speed up the introduction, with the ability to expand to complex applications and has the flexibility of the Python programming language and provides a simple model for web development that offers suggestions, but does not impose any dependency or layout of the project. Once importing into Python. It can be used to streamline the construction of web applications, an example of an application developed with Flask is the framework page. There are extensions capable of adding features that are provided by the community that provides libraries to solve Python issues, this simplifies the framework and makes your learning curve smooth. We will use it to write HTTP communication methods like (GET, POST, PUT, DELETE).

Pillow, Python Library for Image Handling: The Python Imaging Library (PIL) adds image processing capabilities to its Python interpreter (MITCHELL, 2019). It is a free Python programming language library that supports many file formats and provides advanced image and graphics processing capabilities with support for opening, manipulating and saving to PNG, TIFF, BMP, EPS and GIF image formats. It is possible to create new file decoders to spread the library in accessible image formats. PIL or Python Image Library is a package of public functions for manipulating images through a script, inserting image processing capabilities into your language interpreter. An image can represent one or more data lists. The Library grants to store several lists in an image, since all have the equivalent depth and dimensions. Ideally, for the red, green, blue and alpha transparency values a PNG image can have 'A', 'B', 'G' and 'R' bands. Many operations operate on each band separately, for example, histograms. It is often useful to think of each pixel as having a value per band. It is being used to reduce the resolution of the images, compress and stream the image.

Mongodb, Open Source Nosql Database: It is a distributed database software oriented to free general-purpose documents, open source and multiplatform, written in the C ++ language (HOWA; MEMBREY; PLUGGE, 2019). MongoDB uses JSON-like documents with schematics is developed for developers of modern applications and for the cloud era, which stores data in JSON documents with dynamic schema, this allows records without worrying about the data structure, such as the types of fields to store values or number of fields. It is a database that can be run on Windows, Linux and Mac etc. It supports most popular programming languages, such as C #, Java, PHP, JavaScript, NodeJS, Python and others. Its attributes allow applications to model information in a more natural way, so data can be housed in different hierarchies and continue to be easy to find and indexable. It is designed for modern application developers and the cloud era. It is being used to save the image stream.

Swagger, A Tool That Allows Creating Documentation for APIS: It is a mechanism that makes it possible to create API documentation through 3 ways, allows you to write service specifications manually and publish them on your own server or on third parties (DOGLIO, 2015). Automatically, concurrently grants to create the documentation and the API. Codegen, is an application that converts the Swagger annotations included in the source code of the REST APIs into documentation. When consuming an existing API, we need to know the available features and details on how to invoke them: resources, URIs, methods, Content-Types and other information. Currently, it is common for the business market to use REST APIs for application integration, to provide new services or to consume third-party services, Swagger is an open source software framework supported by tools that helps developers to consume RESTful web services, create, design and document. The Swagger toolkit includes code generation, support for automated documentation, UI tool and test case generation. We will use it to visualize the endPoints with the visual documentation, it facilitates the implementation of the backend and the consumption on the client side.

Mongo Express: It's an Administrative Interface for Mongodb: Mongo Express comes with a “config-default.js file”. It supports basic authentication, which addresses the base64-encoded payload (HOLMES, 2016). Basic authentication is configured, the HTTP Authorization: Basic <payload> request header must be passed to access Mongo Express web components. The Mongo Express package supports database authentication credentials to be passed through environment variables. Passing credentials through environment variables can result in information leakage, if the host is deployed using containers or virtual machines. Basic authentication is used to protect Mongo Express environment variables and web administrative panels and is used during credential storage to make it possible to configure back-end

connections to the main database. That we will use to manage the information in the database and visualize the data persisted in the database.

MATERIALS AND METHODS

Bibliographic search will be used to search for content to build knowledge about API Rest and Python in books, articles and magazines. Descriptive research will be applied to clarify the methods used so that the reader can understand the subject. The exploratory research method was used to implement a new Python-based Rest API based on a product that already exists, providing the basis for creating a new product.

Docker is a Container manager using to facilitate the application deployment in any environment. Python is a programming language being used to write business rule functions. Flask is a framework written in Python that we will use to write HTTP communication methods. Pillow is a Python library for manipulating images, it is being used to reduce the resolution of images, compress and stream the image. MongoDB is a database being used to store streaming images. Swagger is a tool that allows you to create documentation for APIs, we will use it to view endPoints, with visual documentation. Mongo Express is a web-based administrative interface for MongoDB, which will be used to manage database information and view persistent data in the database.

RESULTS AND DISCUSSION

The information stored and processed daily is the main means of information and decision making for organizations and people, but among this information, countless images of different sizes and formats are passed daily, corresponding to the loss of performance in applications. According to the largest search engine, the speed of your site is one of the main points taken into account for distribution in search results. Performance when loading a website is one of the main points, but often overlooked. Sites with performance issues taking 10 seconds or more to render are negatively affected according to the Googlers themselves. In practice, slow sites take longer to be indexed by the Google robot (crawler). And it is not a good strategy, it affects the number of pages indexed by the search engine. Akamai together with Forrester Consulting conducted a survey of 1,048 consumers online and obtained the results: Of consumers 40% will not wait 3 seconds for the page to load. After that time they will leave the site. Consumers 52% say that speed is linked to loyalty to a website. 47% of consumers expect the page to load in 2 seconds or less. Another factor is how much a slow page can take the focus away from the consumer selling process. The applications with the image download scenario, use the consumption of images on a large scale with several different application requests, the cost for this operation would be higher on the server and client side providing problems such as display time of images on desktop devices and getting worse on mobile because it has an audience that has a preference for real-time applications, compared to sending an image with a size of 778 KB the response time is 217 milliseconds, knowing that a large-scale system would not be supplied, we propose the development of a tool that optimizes this process, so through this we will propose the construction of this API to have a comparison of the method currently used. One of the main causes of slowness on your site is the application of heavy images, in some cases heavy images can make your site 3 times slower, within this environment we

Fig. 1: Folder structure Source: Authors, (2020)

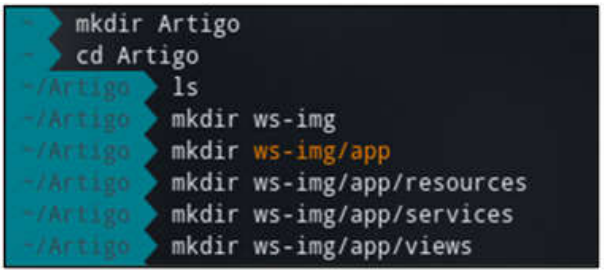

Fig. 2: Linux commands to create the folder structure. Source: Authors, (2020)

Fig. 3: Command to enter the folder Source: Authors, (2020).

Fig. 4: Installing python 3.8 Source: Authors, (2020).

Fig. 5: Installation of used libraries. Source: Authors, (2020).

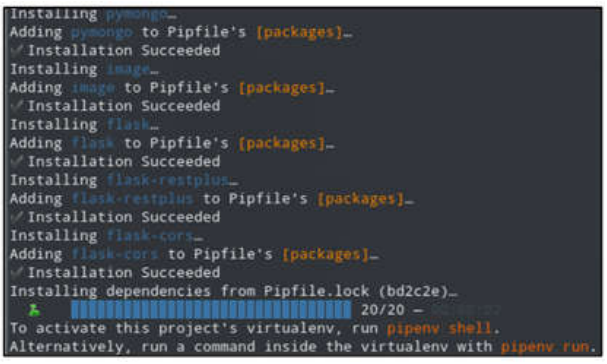

Fig. 6. Libraries successfully installed Source: Authors, (2020).

will start to develop the API, a folder for the project must be created, inside from the project folder create the ws-img folder and within it create the folder structure (Fig. 1). We can see the Linux commands to create the folder structures (Fig. 2).You must enter the folder "ws-img" (Fig. 3). In order to start the installation of python 3.8, execute the command (Fig. 4). After installation to install python, execute the commands (Fig. 5).

```yaml
version: '3.1'

services:

  mongo:
    image: mongo:4.0.4
    ports:
      - "27017:27017"
    restart: always

  mongo-express:
  image: mongo-express
  restart: unless-stopped
  ports:
    - 8081:8081
  environment:
    ME_CONFIG_MONGODB_SERVER: 192.168.101.126
```

Fig. 7: Mongodb configuration files Source: Authors, (2020).

```
[sys05@sys05-pc Artigo]$ docker-compose ps
        Name                      Command               State          Ports
-------------------------------------------------------------------------------------
artigo_mongo-express_1   tini -- /docker-entrypoint ...   Up      0.0.0.0:8081->8081/tcp
artigo_mongo_1           docker-entrypoint.sh mongod      Up      0.0.0.0:27017->27017/tcp
artigo_py-image_1        /bin/sh -c pipenv install  ...   Up      0.0.0.0:5555->5555/tcp
```

Fig. 8: Containers that are in use Source: Authors, (2020)

```python
import os
import sys

from flask import Flask
from flask_cors import CORS
from flask_restplus import Api
from werkzeug.contrib.fixers import ProxyFix

app = Flask(__name__)
profile = os.environ['PROFILE']
properties = {}
resources = os.getcwd() + '/app/resources'
sys.path.append(resources)

try:
    app.config.from_object('config_{}'.format(profile))
except FileNotFoundError:
    print('Profile não existe')
    sys.exit(4)

api = Api(app, version='1.0.1', title='Py-Image API', description='Image Controller')
app.wsgi_app = ProxyFix(app.wsgi_app)
ns = api.namespace('image', description='All operations  image')
CORS(app)

from .views import image_view
```

Fig. 9. Configuration of the flask framework. Source: Authors, (2020).

```python
from flask import send_file
from flask_restplus import Resource

from app.py_image import app, ns, api


@ns.route('/version')
class Version(Resource):
    def get(self):
        """
            Version project.
        """
        return api.version
```

Fig. 10: Route file. Source: Authors, (2020)

```python
from app.py_image import app

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5555, debug=True)
```

Fig. 11: Configuration commands Source: Authors, (2020)

```
[sys05@sys05-pc ws-img]$ PROFILE=develop python ws-img.py
 * Serving Flask app "app.py_image" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:5555/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 216-582-140
```

Fig. 12: Starting the application server Source: Authors, (2020)

Fig. 13: Route running in the browser Source: Authors, (2020)

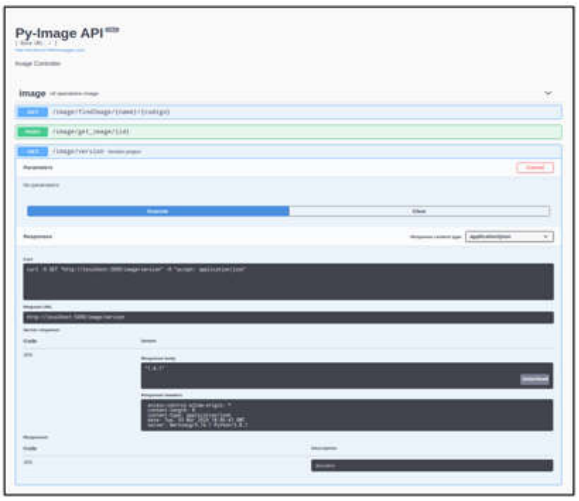

Fig. 14: Route running through the swagger Source: Authors, (2020)

To start the installation of the libraries that will be used in the project, we can see the example of the libraries that were successfully installed in the project in (Fig. 6). The “Pipfile” file will contain the libraries that are being used in the project. You must start by uploading the container with mongodb, inside the folder create the file “docker-compose.yml” (Fig. 7). Upload the container with the mongodb, via terminal accessing the mongo's folder, executing the command “docker-compose up -d”, to validate that the container is working correctly “docker-compose ps”. After starting the container with mongodb, via the terminal, navigate to the mongo's folder, type “docker-compose up -d”, to check if the container is working correctly “docker-compose ps”.

```python
@ns.route('/get_image/<id>')
class ImageDefault(Resource):

    def get(self, id):
        if id == '1':
            filename = app.config['DIRETORIO']+'ok.jpg'
        else:
            filename = app.config['DIRETORIO']+'error.jpg'
        return send_file(filename, mimetype='image/jpg')
```

Fig. 15: Route to obtain (GET) image Source: Authors, (2020).

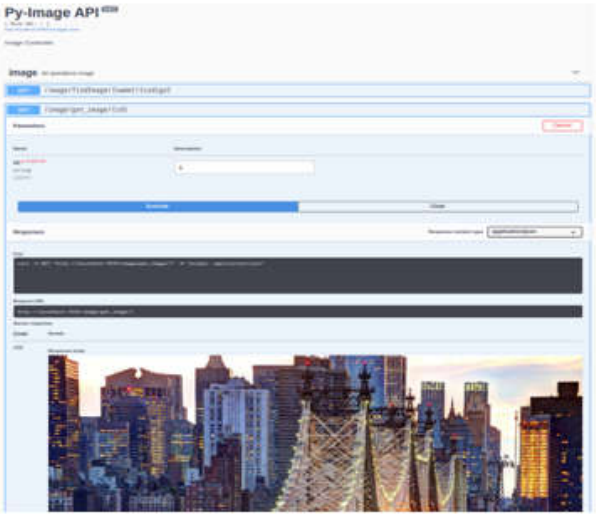

Fig. 16. Return route you get (GET) Image Source: Authors, (2020).

```python
@ns.route('/findImage/<name>/<codigo>')
class Compact(Resource):
    def get(self, name, codigo):

        response = imageService.compactImage(name, codigo)

        return response
```

Fig. 17. Route that calculates the image height Source: Authors, (2020).

```python
def compactImage(name, codigo):
    data = []
    response = {}

    data.append(buscaImg(int(codigo), name))

    if data[0] != None:

        return jsonify(data)

    elif data[0] == None:


        response = compactTransformImg(int(codigo), name)
        jsonify(response)

    else:
        response["menssage"] = "Imagem não existe no servidor"

    return jsonify(response)
```

Fig. 18: Connection function and image compactor Source: Authors, (2020)

```python
def buscaImg(codigo, name):
    query1 = {"_id": codigo, "name": name}

    response = mycol.find_one(query1)

    return response
```

Fig. 19. Consultation with mongoDB Source: Authors, (2020).

```python
def compactTransformImg(codigo, name):
    """
        Função para compacta, redimensionar a imagem, tranforma em base64 e salvar no
            banco de dados.
        :argument:
           codigo, name
        :return:
            Json contendo um array com _id, base64 e name.
    """

    arquivo = app.config['DIRETORIO'] + name + "/" + str(codigo) + ".JPG"

    basewidth = 400
    img = Image.open(arquivo)

    wpercent = (basewidth / float(img.size[0]))
    hsize = int((float(img.size[1]) * float(wpercent)))
    new_img = img.resize((basewidth, hsize), Image.ANTIALIAS)
    new_img.save(app.config['DIRETORIO'] + name + "/newImg" + str(codigo) + ".JPG",
        optimize=True)
    new_arquivo = app.config['DIRETORIO'] + name + "/newImg" + str(codigo) + ".JPG"

    f = open(new_arquivo, 'rb')
    imgCompact = f.read()
    f.close()

    encodedImg = base64.b64encode(imgCompact)

    documento = {"_id": codigo, "name": name,
                 "base64": "{}".format(encodedImg).replace("b'", "").replace("'", "")}
    x = mycol.insert_one(documento)

    os.remove(new_arquivo)

    response = buscaImg(codigo, name)

    return response
```

Fig. 20: Streaming optimization and transformation Source: Authors, (2020).

Fig. 21: Swagger-ui Source: Authors, (2020).

If everything is working, the following message will be informed (Fig. 8). Inside the "py-image.py" file, insert the following lines of code (Fig. 9). In the "image_view.py" file that is inside the views folder, it will be our controller making an analogy to Java, where our routes were, and their respective rules (Fig. 10). Enter the file "ws-img.py" copy the following code snippet (Fig. 11). Following we can test the application, via terminal navigate to the "app" folder, execute the following command "PROFILE = develop python ws-img.py" (Fig. 12).

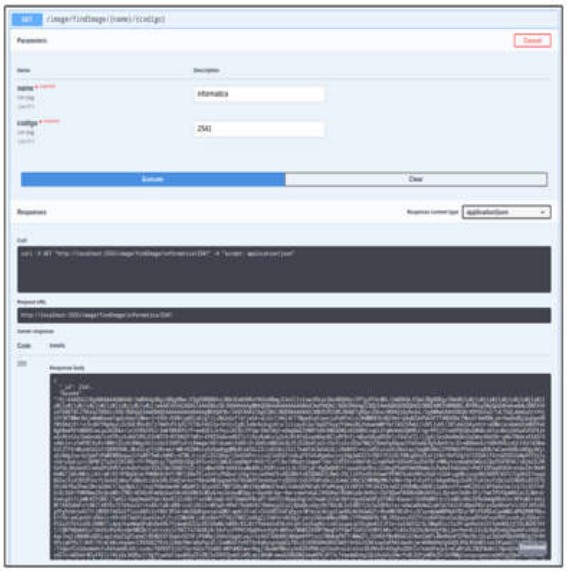

Fig. 22: Route response Source: Authors, (2020).

```
py-image:
    user: root
    build:
      dockerfile: Dockerfile
      context: ./
    ports:
      - 5555:5555
    volumes:
      - ./:/usr/src/app
    links:
      - mongo
    depends_on:
      - mongo
```

Fig. 23: Configuration commands Source: Authors, (2020).

```
FROM python:3.8

WORKDIR /usr/src/app/ws-img

EXPOSE 5555

RUN pip install pipenv

ENTRYPOINT pipenv install --system && \
           PROFILE=develop python ws-img.py
```

Fig. 24: System configuration commands Source: Authors, (2020).

Open the browser and enter the address “localhost: 5000 / version” you can see the answer in the browser (Fig. 13). It is possible to see this same route through Swagger-ui, he is responsible for documenting our API, in your browser type “localhost: 5000 /” we can see the one in more detail (Fig 14). Having the entire API structure ready, it is necessary to implement the functionalities.

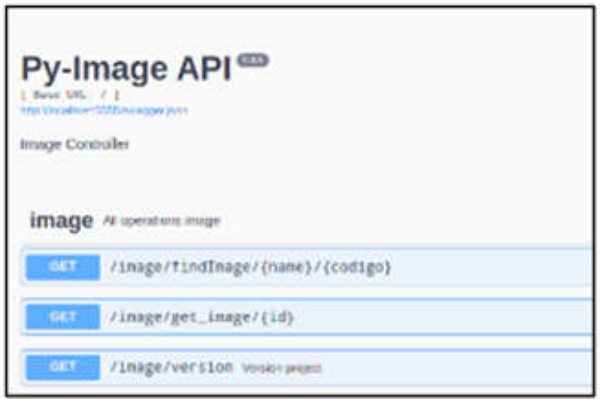

Fig. 25: Swagger-ui Source: Authors, (2020).

Fig. 26: Mongo-Express Source: Authors, (2020).

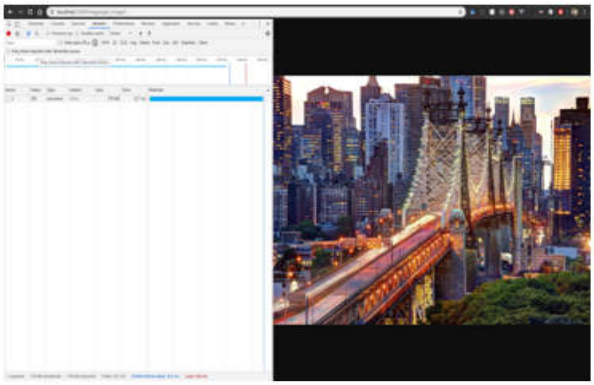

Fig. 27: Size and response time Source: Authors, (2020).

It is necessary to elaborate a route that returns the images in JPG format so that you have the possibility to compare the performance of the presented solution. You need to go to the “image_view” controller to create a new route (Fig. 15). This route is responsible for receiving a request with an 'id' as a parameter, and returns to the image that is in our image repository inside the / baseImage folder, the image “ok.jpg” and if you don't find the image it returns a error image “error.jpg”. Before the test, it is necessary to download the images, available on the google driver, follow the link:

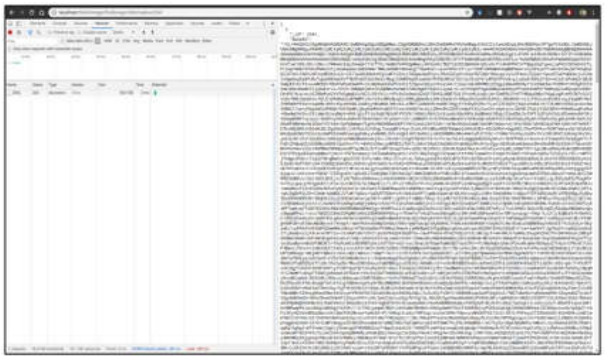

Fig. 28: Response time Source: Authors, (2020).

https://drive.google.com/open?id=1lv-AwF0g DKhRHzi-tfGeGnsI8Z5UCGhS. After downloading, unzip the file in the project folder. Open the browser, type "localhost: 5555", it is necessary to go to the endPoint "/ image / get_img / {id}" type "1" in the ID field, execute if everything happens right there will be a return (Fig. 16). We will work with a predefined width for our images, but being able to create an Array with different sizes, depending on the need of the application, one of the possibilities would be more than one application consuming this service, and need different sizes of the image, it can be a mobile device or a website, so that it is possible to serve both with just one registration at the bank. We will define only one width to calculate the proportional height for the image. Create the "image_view.py" controller and add this new endPoint (Fig. 17). Inside the services folder, open the file "imageService.py", where the connection to mongoDB will be created, and all the business rules of the API. After elaborating the connection function and the "compactImage ()" function, which is being called inside the controller (Fig. 18).

In the same file, the query to mongoDB is made, which is called within the function "compactImage ()", in the previous code snippet the method "buscaImg ()" is called, the method "buscaImg ()" will be implemented (Fig. 19). It will be to elaborate the method that will be responsible for the optimization and transformation of streaming images (Fig. 20).

Use Swagger-ui in the browser "localhost: 5555 /", in the route "/ image / findImage / {name} / {code]", pass by parameter the name of the folder that I keep the image and the code for it (Fig . 21). The route will have a return (Fig. 22).Within services it is necessary to edit the "docker-compose.yml", add the code snippet (Fig. 23). Create the Dockerfile file, and enter the commands (Fig. 24). Run the complete project in the docker, go to the root folder where your docker-compose is, open it in the terminal and type "docker-compose up -d --build". Links: Swagger-ui: localhost: 5555 / (Fig. 25). In Mongo-Express: localhost: 8081 / (Fig. 26). The size and response time of the method used exposed (Fig. 27). We have 217 milliseconds in the response time, with a size of 778 KB, in the counter-proposal the response time and size of an image using the proposed method (Fig. 29). It is only 3 milliseconds in response time and only 45 KB in size, we can define different sizes for different applications, several resolutions and this would not interfere with the application's performance. Great performance has been achieved with minimal cost. The tests were designed with a stable internet, the user traveling through a mobile data network that is constantly interfered, these values doubled more and thus they would have a sensation of speed in the rendering of the images in their application, generating a pleasant experience for the user and consequently making to get him back on the platform.

Conclusion

According to the exposed scenario, the image transmission API proved to be a viable medium for solution and management, storage and distribution of images through Api's, presenting a better performance and thus accelerating the process of loading images on websites and applications mobile, managing to decrease the transmission time by approximately 98.6%, which was previously 217 milliseconds and changed to 3 milliseconds in response time without losing the resolution quality and providing users with a good user experience within the time that is approximately three seconds. Therefore, the tools and programming language used in the development of the research provided the project collaborators with a satisfactory result and could contribute to the community. The means developed and presented are relevant to cognitive research to share knowledge with the scientific community expressing the knowledge of sending and receiving images through a REST API, leveraging education, human values and ethics.

Acknowledgements

First, I would like to thank God. I thank my advisor Bruno Gonçalves for conducting our research work. To all my teachers in the Information System course at the Metropolitan University of Manaus - FAMETRO for the excellence of the technical quality of each one. To all my friends from the undergraduate course who shared the countless challenges we face, always with a collaborative spirit. To my family members who have always been by my side supporting me throughout my career.

REFERENCES

Banin, S. L. 2018. Python 3: Conceitos e aplicações uma abordagem didática (1th ed.). Érica.

Doglio, F. 2015. Pro REST API Development with NODE.js. ISBN-13.

Fredrik. Handbook. Disponível em: <https://pillow.readthedocs.io/en/stable/handbook/index.html>. Acesso em: 23 fev. 2020.

FRESHLAB. Conheça as principais causas de lentidão de seu site. Disponível em: <https://www.freshlab.com.br/web-design/conheca-as-principais-causas-de-lentidao-de-seu-site/>. Acesso em: 08 mar. 2020.

Grinberg, M. 2018. Desenvolvimento web com flash: Desenvolvendo aplicações web com Python (1th ed.). Novatec Editora Ltda.

Holmes, S. 2016. MEAN Definitivo: com Mongo, Express, Angular e Node (1thed.). Novatec Editora Ltda.

Hows, D.; Membrey, P.; Plugge, E. 2015. Introdução ao MongoDB (1th ed.). Novatec Editora Ltda.

Lassance, Raphael. As pessoas gostam de sites rápidos. O Google também. Disponível em: <https://www.linkedin.com/pulse/2-segundos-de-carregamento-e-agora-raphael-lassance/>. Acesso em: 08 mar. 2020.

Matos, David. Ciência E Dados: Por que Cientistas de Dados escolhem Python?. Disponível em: <http://www.cienciaedados.com/por-que-cientistas-de-dados-escolhem-python/>. Acesso em: 27 fev. 2020.

Mitchell, R. 2019. Web Scraping com Python: Coletando mais dados na web moderna (2th ed.). NovatecEditoraLdta.

Mongodb. MongoDB Atlas - the global cloud database. Disponível em: <https://www.mongodb.com/>. Acesso em 23 fev. 2020.

Silva, W. F. 2016. Do básico à orquestração de contêiners: Aprendendo Docker (1th ed.). NovatecEditora Ltda..

Smartbear Software. 2020. API Development for Everyone. Disponível em: <https://swagger.io/>. Acessoem 23 fev.

Yamamoto, Cleber. Oráculo TI: Tecnologia, Informação e Cursos. Disponível em: <https://oraculoti.com.br/2017/03/04/guido-van-rossum-fala-sobre-o-passado-presente-e-futuro-do-python/>. Acessoem 23 fev. 2020.

****