

Entrega 1 - Arquitectura, conclusiones y consideraciones

Grupo 3

C. Camilo Baquero Gómez, Julian Torres, Franklin A. Pinto Carreño.

Desarrollo de Aplicaciones Cloud

Universidad de los Andes, Bogotá, Colombia

c.baquero@uniandes.edu.co, jy.torres@uniandes.edu.co, f.pintoc@uniandes.edu.co

Fecha de presentación: febrero 24 de 2023

GitHub:

GitHub Video:

1. Arquitectura de la aplicación

La aplicación web de compresión de archivos se encuentra implementada bajo el modelo vista controlador. El modelo implementa un conjunto de métodos para crear, modificar, eliminar, consultar, comprimir y descomprimir archivos y tiene acceso directo al motor de persistencia.

La vista está implementada en formato html para los formularios y páginas de presentación en capa web, y para las api rest, se utiliza el formato json, para capturar y responder las peticiones web. El controlador es el intermediario entre el modelo y la vista para interpretar las peticiones y entregar una respuesta a cada petición web realizada por un usuario.

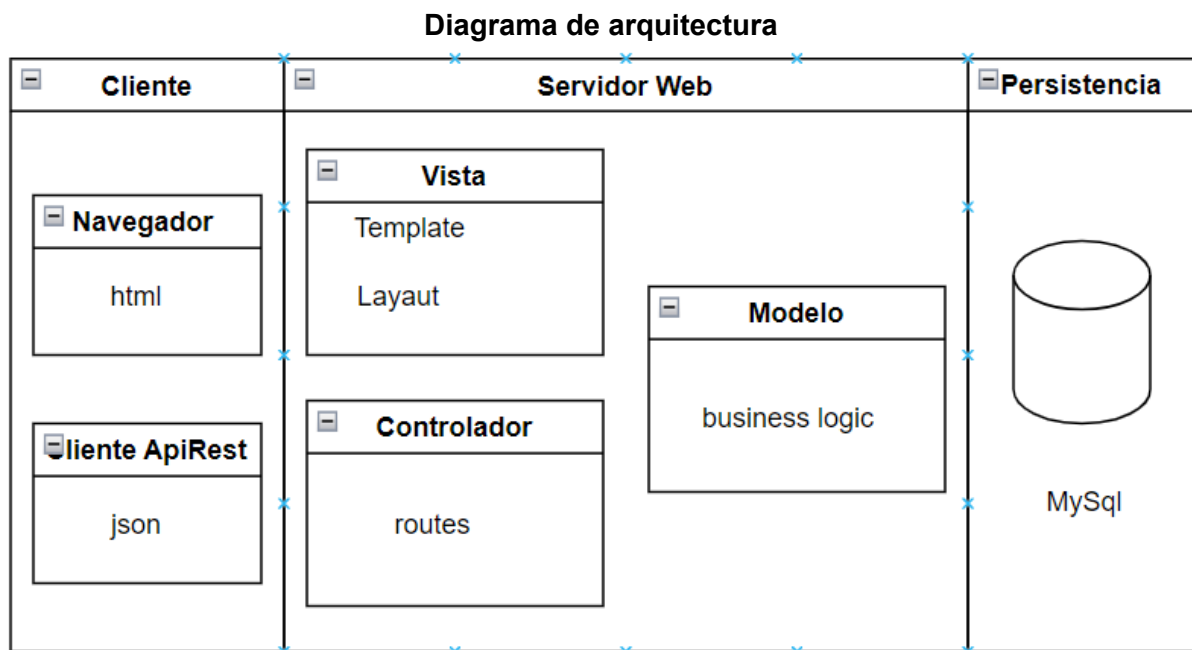
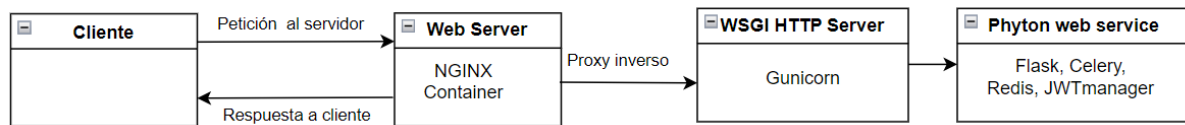


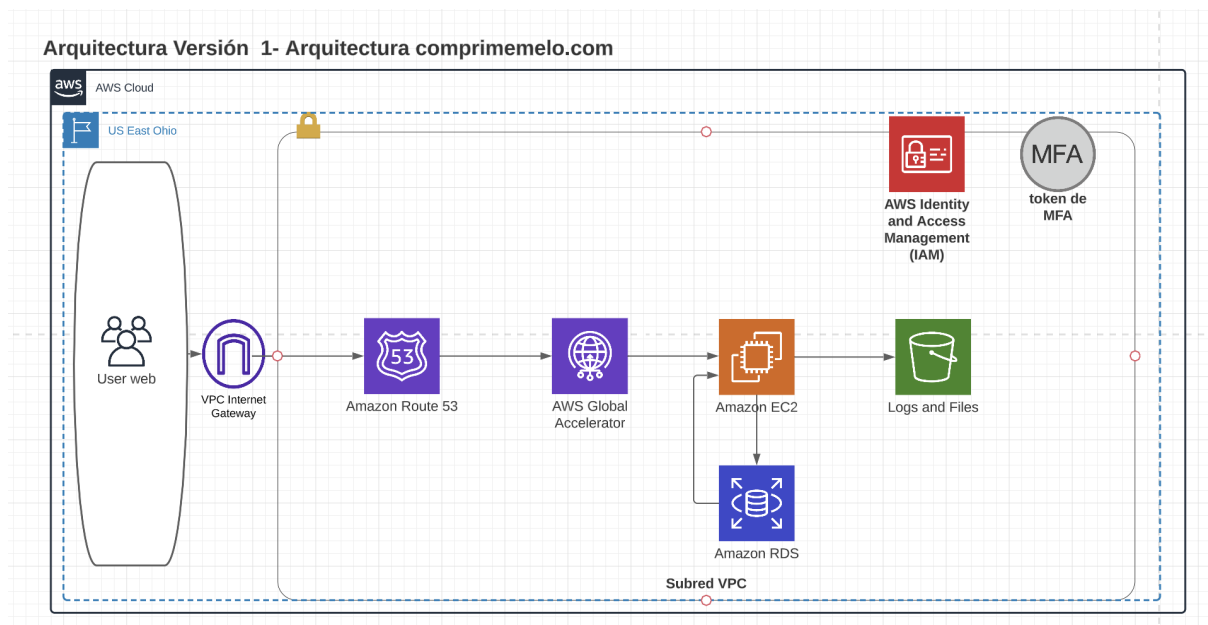
Figura 1. Diagrama de arquitectura aplicación de compresión de archivos

Para el despliegue de la aplicación Flask en ambiente productivo es indispensable usar el servidor HTTP WSGI llamado Gunicorn, el cual nos provee una mejor alternativa cuando se trata de escalar la aplicación.

Diagrama de despliegue



Arquitectura Cloud AWS



La aplicación está corriendo en el dominio www.comprimemelo.com:5000

la cual está configurada con un Route 53 para el dominio y redireccionamiento usando Global Accelerator para mejorar el performance para las edge zones. Posteriormente carga la aplicación de Flask la cual está en una Ec2 t2.small y los archivos y logs son guardados en un s3. La aplicación Flask (EC2) interactúa con la base de datos Mysql Serverless desplegada en el servicio RDS.

2. Validación Requerimientos

1. Desarrollar y exponer un servicio REST (POST) para crear una cuenta de usuario en la aplicación. Para crear una cuenta se deben especificar los campos: usuario, correo electrónico y contraseña. El correo electrónico debe ser único en la plataforma dado que este se usa para la autenticación de los usuarios en la aplicación.

Para crear una cuenta de usuario es posible hacerlo a través de un formulario web ó consumiendo la api rest a través de un cliente http, ejemplo postman..

El siguiente endpoint con método post, permite crear una cuenta de usuario: <http://comprimemelo.com:5000/api/auth/signup>

The screenshot shows the Postman interface for a POST request to `http://comprimemelo.com:5000/api/auth/signup`. The 'Body' tab is selected, showing form data with the following fields:

KEY	VALUE	DESCRIPTION
email	franklin.pintoc@uniandes.edu.co	
first_name	Franklin	
last_name	Pinto	

The response is displayed in JSON format:

```
{  "message": "User created successfully"}
```

o ingresando a la aplicación web en la siguiente url: <http://comprimemelo.com:5000/api/auth/signup> presentará el siguiente formulario para el respectivo diligenciamiento de datos

The screenshot shows a web browser displaying the registration form at `127.0.0.1:5000/auth/signup`. The form is titled 'Registrar Usuario' and includes the following fields:

- Nombre
- Apellido
- Username
- Correo
- Celular
- Nueva contraseña
- Confirmación de contraseña

A 'Regístrate' button is located at the bottom of the form.

2. Desarrollar y exponer un servicio REST (POST) para iniciar sesión en la aplicación web, el usuario provee el correo electrónico/usuario y la contraseña con la que creó la cuenta de usuario en la aplicación. La aplicación retorna un token de sesión si el usuario se autenticó de forma correcta, de lo contrario indica un error de autenticación y no permite utilizar los recursos de la aplicación.

Para iniciar sesión en la aplicación se debe autenticarse ingresando las credenciales de usuario a través de la siguiente url <http://comprimemelo.com:5000/api/auth/login>

Para obtener el token de autenticación de usuario a través de la api rest debe usar el siguiente endpoint con metodo POST: <http://comprimemelo.com:5000/api/auth/login>

POST

http://comprimemelo.com:5000/api/auth/login

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	email	franklin.pintoc@uniandes.edu.co			
<input checked="" type="checkbox"/>	password	pwd			
	Key	Value	Description		

Body

Cookies

Headers (5)

Test Results

200 OK175 ms519 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

2

3

{

"access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.

eyJmcmVzaCI6ZmFsc2UsIm1hdCI6MTY3NzI1NzAyNywiYW50b3R1LnBpbmR1cy51ZHUuY28iLCJuYmYiOiJlcyNTcwMjc5ImV4cCI6MTY3NzI1NzkyN30.vI8nlnY7AHRW_eB8XA7pk5Leu7aqW2XnlhGW3bv6ZxU",

"message": "Login Successful"

}

3. Desarrollar y exponer un servicio REST (GET) para listar todas las tareas de conversión de un usuario, el servicio entrega el identificador de la tarea, el nombre y la extensión del archivo original, a qué extensión desea convertir y si está disponible o no. El usuario debe proveer el token de autenticación para realizar dicha operación.

Comprimemelo.com / Task List

GET

http://comprimemelo.com:5000/api/tasks

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Type

Bearer Token

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, variables are masked.

The authorization header will be automatically generated when you send the request.
[Learn more about authorization](#)

Token

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJmcmVzaCI6ZmFsc2UsIm1hdCI6MTY3NzI1NzAyNywiYW50b3R1LnBpbmR1cy51ZHUuY28iLCJuYmYiOiJlcyNTcwMjc5ImV4cCI6MTY3NzI1NzkyN30.vI8nlnY7AHRW_eB8XA7pk5Leu7aqW2XnlhGW3bv6ZxU

Body

Cookies

Headers (5)

Test Results

Status: 200 OK

Pretty

Raw

Preview

Visualize

JSON

1

2

3

"{

"tasks": [

{"id": "1", "original_file": "pdf", "format": "rar", "status": "processed"}

]

}

4. Desarrollar y exponer un servicio REST (POST) para subir y cambiar el formato de un archivo. El usuario debe proveer el archivo que desea convertir, el formato al cual desea cambiarlo y el token de autenticación para realizar dicha operación. El archivo debe ser almacenado en la plataforma, se debe guardar en base datos la marca de tiempo en el que fue subido el archivo y el estado del proceso de conversión (uploaded o processed). Cuando el archivo se termina de procesar se debe notificar al usuario vía correo electrónico.

Para cargar el archivo se puede hacer por medio del formulario web en la aplicación, una vez inicia sesión en la página principal aparece un botón de “Comprimir Archivo” el cual desplegará el formulario para seleccionar el archivo y seleccionar el tipo de formato a convertir, así:

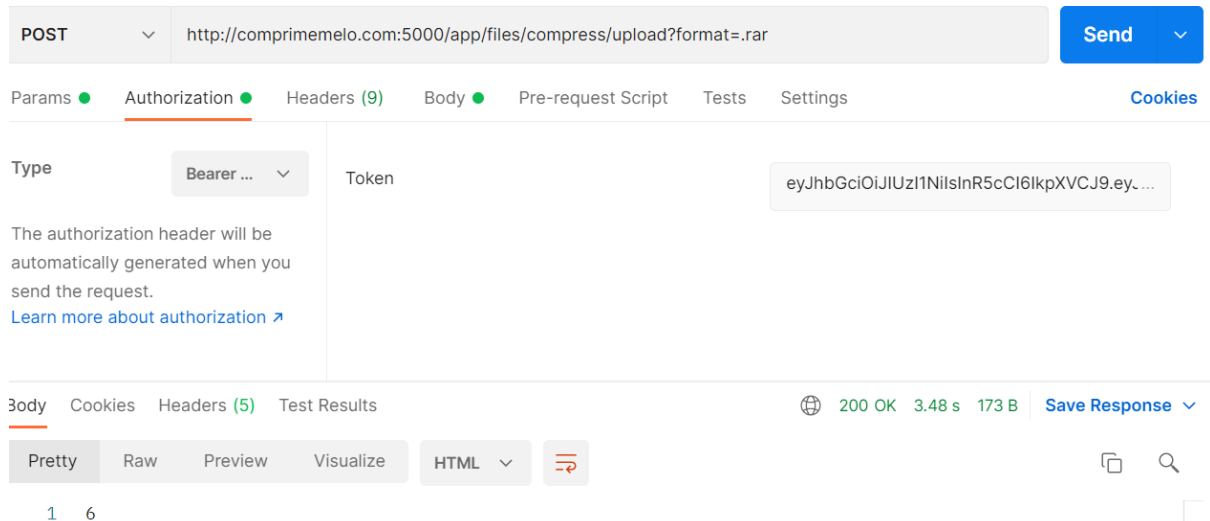


The screenshot shows a web browser at the URL 127.0.0.1:5000/eventos/comprimir. The page has a navigation bar with links for Home, Categorías, Comprimir, and f.pintoc. Below the navigation bar are two buttons: "Crear Evento" and "Comprimir Archivo". The main content area is titled "Compresión de archivos" and contains a form with a file selection button labeled "Seleccionar archivo", a dropdown menu currently set to "zip", and a "Comprimir" button.

Para usar la api rest, debe usar el endpoint Con método POST

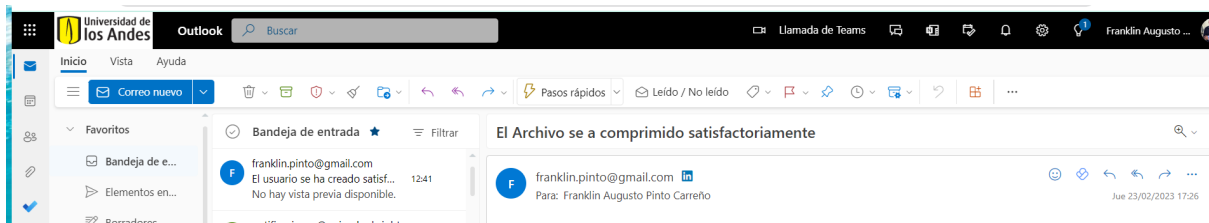
<http://comprimemelo.com:5000/app/files/compress/upload?format=.rar>

indicar el tipo de formato de conversión a través del parámetro “format” enviado como query param.



The screenshot shows a REST client interface. The method is set to POST and the URL is <http://comprimemelo.com:5000/app/files/compress/upload?format=.rar>. The "Authorization" tab is selected, showing a Bearer token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ... The response status is 200 OK, with a response time of 3.48 s and a body size of 173 B. The response is displayed in HTML format.

Cuando el archivo termina de comprimirse se envía correo al usuario autenticado a través de la aplicación o autenticado a través de la api rest con el web token.



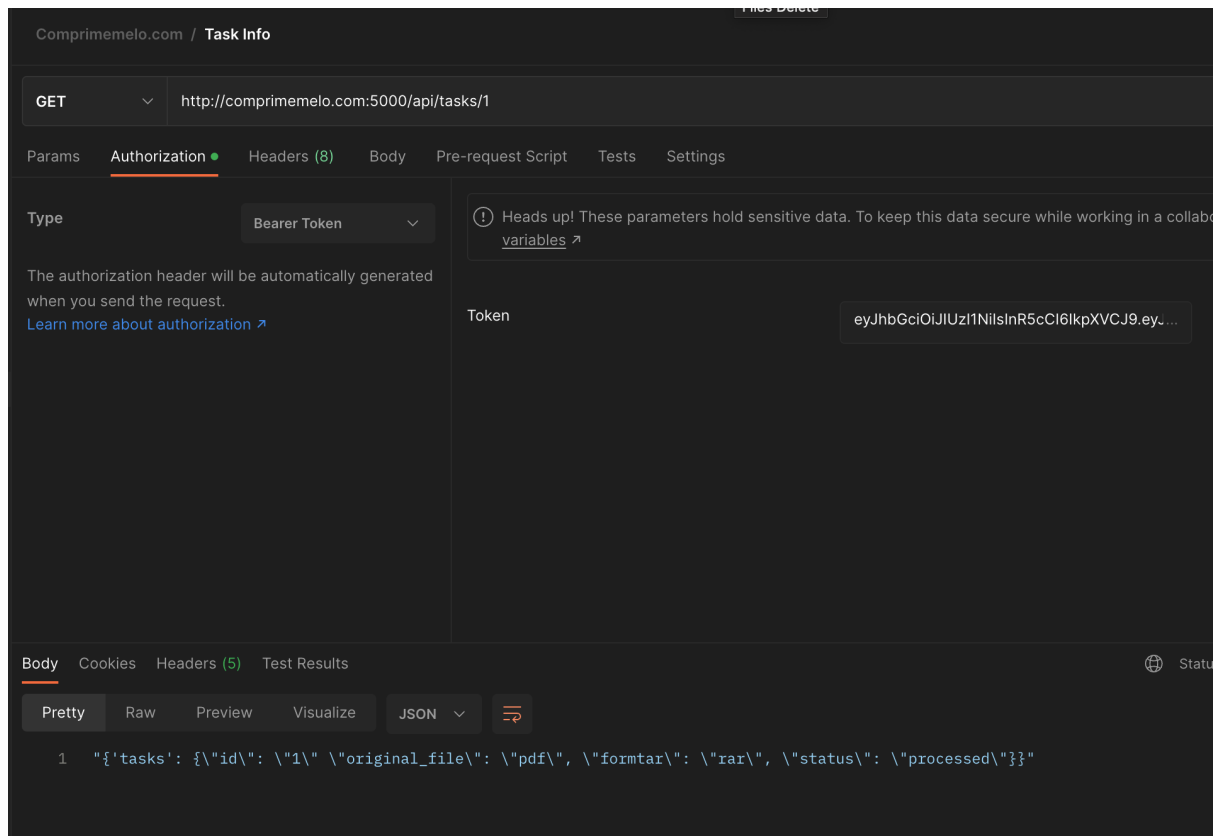
Para descargar el archivo comprimido, con el id generado en la respuesta del endpoint anterior, usar el siguiente endpoint con método POST
<http://comprimemelo.com:5000/app/files/compress/download/6>

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** `http://comprimemelo.com:5000/app/files/compress/download/6`
- Authorization:** Bearer token `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...`
- Response Status:** 200 OK, 732 ms, 2.93 MB
- Response Body (Raw):**

```
1 PKXXXXXXXXXXGXXV.XXX(XX(XX#XXX4-ComputacionDistribuida-Hadoop.pdf%PDF-1.5
2 %XXXXX
3 7511 0 obj
4 <</Linearized 1/L 3076648/O 7513/E 136644/N 178/T 3074778/H [ 497 1843]>>
5 endobj
6
7 7511 0 obj
```

5. Desarrollar y exponer un servicio REST (GET) para obtener la información de una tarea de conversión específica. El usuario debe proveer el token de autenticación para realizar dicha operación.

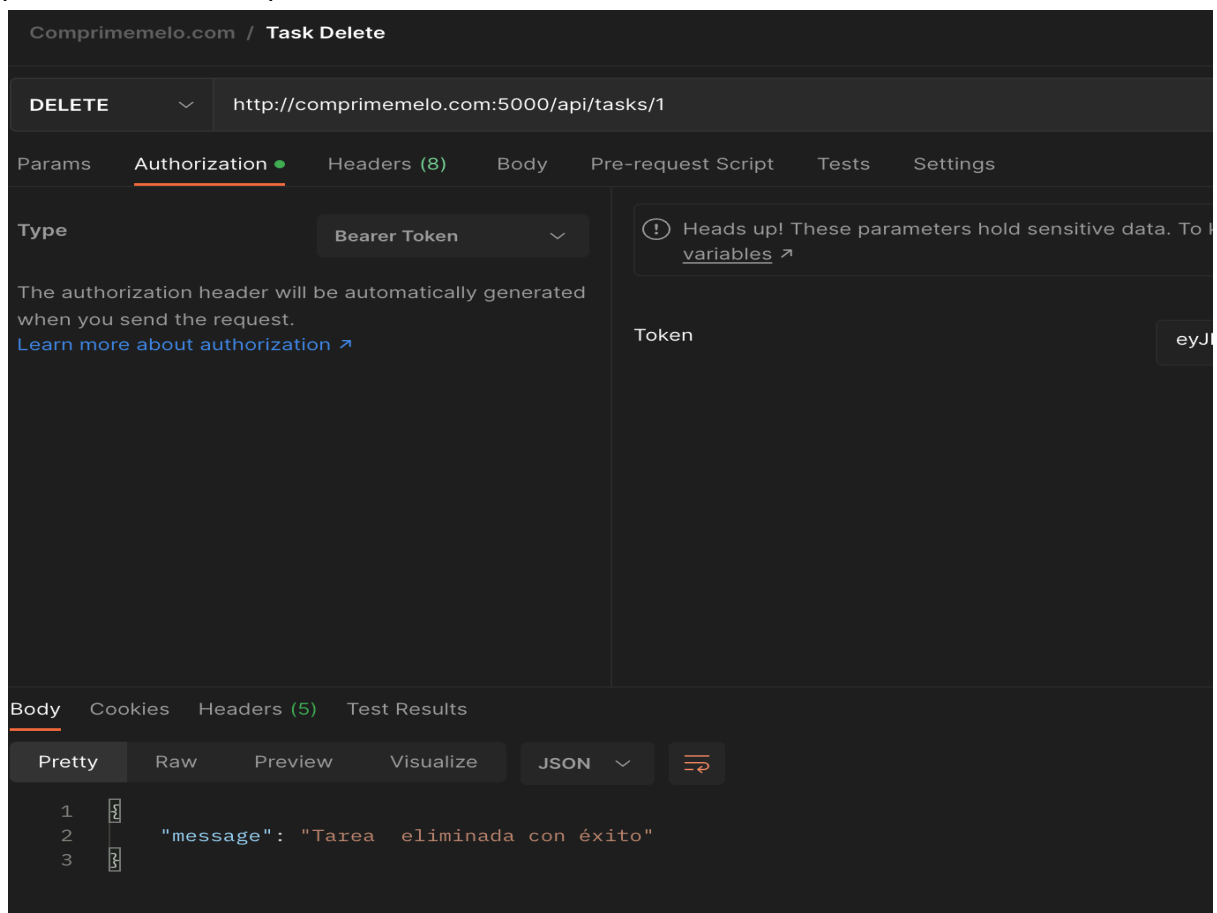


6. Desarrollar y exponer un servicio *REST* (*GET*) que permite recuperar/descargar el archivo original o el archivo procesado de un usuario.

Usar el siguiente endpoint <http://comprimemelo.com:5000/api/files/4-ComputacionDistribuida-Hadoop.pdf> con metodo *GET* para descargar el archivo, se debe indicar el nombre exacto del archivo cargado:



7. Desarrollar y exponer un servicio REST (DELETE) para borrar el archivo original y el archivo convertido de un usuario, si y sólo si el estado del proceso de conversión es Disponible. El usuario debe proveer el identificador de la tarea y el token de autenticación para realizar dicha operación.



5. Bibliografia

<https://docs.celeryq.dev/en/stable/>

<https://flask.palletsprojects.com/en/2.2.x/>

<https://flask-jwt-extended.readthedocs.io/en/stable/>