ENTREGA 1 GRUPO 8

INTEGRANTES:

- Danny Pineda Echeverri - d.pinedae@uniandes.edu.co
- Vihlai Maldonado Cuevas - v.maldonado1@uniandes.edu.co
- Felipe Alejandro Paez Guerrero - f.paezg@uniandes.edu.co

LINEAMIENTOS DE DISEÑO

Software:

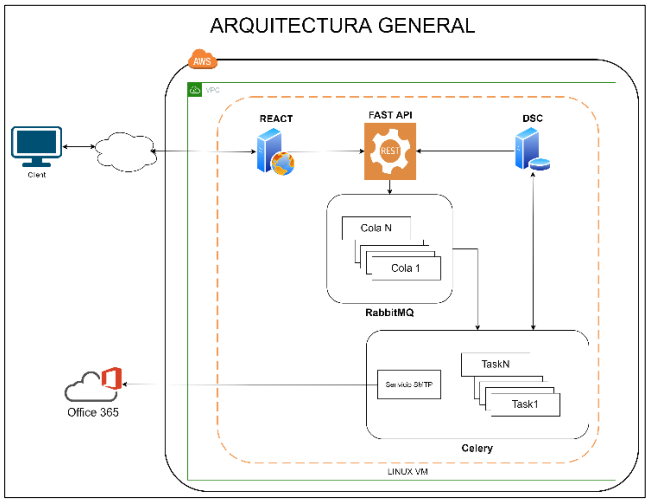
- BackEnd: Python Fast API
- FrontEnd: React 18.2.0
- Base de datos SQLAlchemy
- Sistema Operativo: Máquina Ubuntu 20.04
- Batch: Celery
- SMTP: smtp.office365.com.

Hardware:

- Máquina Virtual Linux en AWS.
- Número de CPUs: 1 CPU/Core
- RAM: 1 GB
- Almacenamiento: 10 GB

DIAGRAMA DE ARQUITECTURA

A continuación, se muestra la arquitectura general de la aplicación, los módulos creados para su funcionamiento y la interacción entre ellos y los usuarios externos que la consumen.



MODELO DE DATOS

La DB se definió con el nombre DSC y crea 2 tablas ***User*** y ***Task***.

User**:** En esta tabla se encuentra la información de identificación del cliente al momento de realizar el registro en la página. A continuación, se definen las columnas de la tabla:

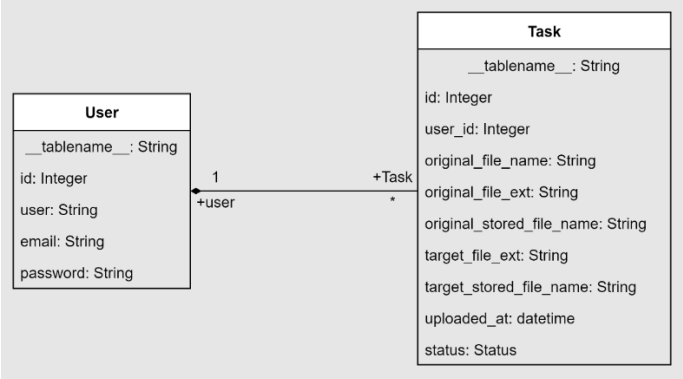
- Id: campo de tipo entero, es el primary key de la tabla e identifica cada cliente con un único número.
- user: campo de tipo String, contiene el nombre del usuario el cual debe ser único en la tabla.
- email: campo de tipo String, contiene el correo del usuario.
- password: campo de tipo String, contiene el password creado por el usuario del usuario.
- tasks: campo de tipo lista, representa la relación entre usuarios y tareas.

Task:

- id: campo de tipo entero, es el primary key de la tabla e identifica la tarea creada por el usuario.
- user_id: campo de tipo entero, contiene el id del usuario que creo la tarea. Llave foránea que relaciona la tabla usuario con la tabla task.
- original_file_name: campo de tipo *String*, contiene el nombre original del archivo.
- original_file_ext: campo de tipo *String*, contiene la extensión original con la que se cargó el archivo.
- original_stored_file_name: campo de tipo *String*, contiene el nombre original con la que se guardó el archivo.
- target_file_ext: campo de tipo *String*, contiene la extensión a la que se desea convertir el archivo.
- target_stored_file_name: campo de tipo *String*, contiene el nombre con la que se guardó el archivo convertido.
- uploaded_at: campo de tipo *timestamp*, contiene la información de la fecha y hora a la que se cargó el archivo.
- status: campo de tipo *Enum*, contiene información del estado de la tarea. Los estados posibles son *"uploaded"* y *"processed"*.

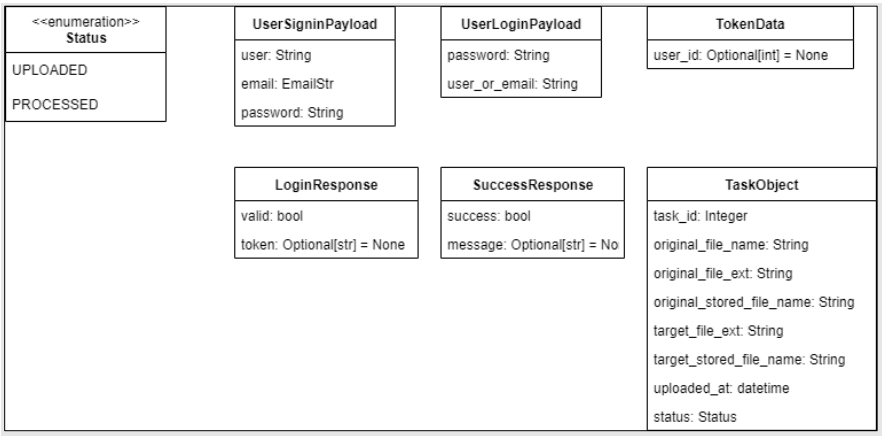
TABLAS DB

A continuación, se muestran las tablas creadas en la DB con sus respectivas relaciones:



CLASES API MODELS

A continuación, se describen las clases definidas para apoyar el proceso de conversión, identificación y estado de la tarea.



- Status

UPLOADED: campo de tipo String, utilizada en la clase como diccionario para identificar un estado, en este caso es estado cargado.

PROCESSED: campo de tipo String, utilizada en la clase como diccionario para identificar un estado, en este caso es estado procesado.

- UserSigninPayload

user: campo de tipo String, contiene el nombre del usuario que viene dentro del payload al momento de registrarse.

email: campo de tipo String, contiene el email del usuario que viene dentro del payload al momento de registrarse.

password: campo de tipo String, contiene el password del usuario que viene dentro del payload al momento de registrarse.

- UserLoginPayload

user_or_email: campo de tipo String, contiene el nombre del usuario registrado que viene dentro del payload al momento de loguearse.

password: campo de tipo String, contiene el password del usuario que viene dentro del payload al momento de loguearse.

- TokenData

user_id: int (Optional), contiene el identificador del usuario.

- LoginResponse

valid: campo de tipo booleano, contiene el estado de la respuesta de la autenticación.

token: campo de tipo String, contiene el token resultado de la autenticación.

- SuccessResponse

success: campo de tipo booleano, contiene estado de la creación de la tarea

message: campo de tipo String, contiene el mensaje de respuesta de creación de la tarea.

- TaskObject

task_id: campo de tipo entero, identifica la tarea creada.

original_file_name: campo de tipo String, contiene el nombre original del archivo.

original_file_ext: campo de tipo String, contiene la extensión original con la que se cargó el archivo.

original_stored_file_name: campo de tipo String, contiene el nombre original con la que se guardó el archivo.

target_file_ext: campo de tipo String, contiene la extensión a la que se desea convertir el archivo.

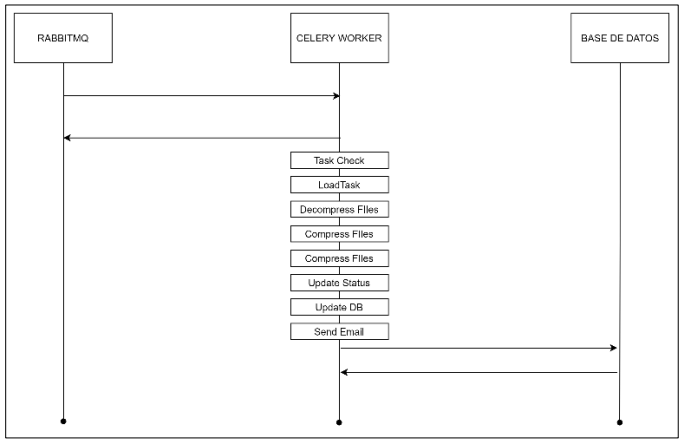
target_stored_file_name: campo de tipo String, contiene el nombre con la que se guardó el archivo convertido.

uploaded_at: campo de tipo datetime, contiene la información de la fecha y hora a la que se cargó el archivo.

status: campo de tipo Enum, contiene información del estado de la tarea.

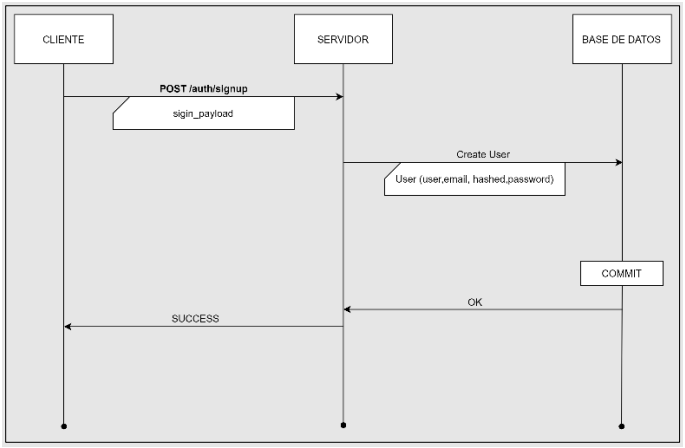
BATCH

Se uso Celery para consulta de archivos en estado uploaded, convertir archivos y guardarlos, actualizar estado de las tareas y enviar correos

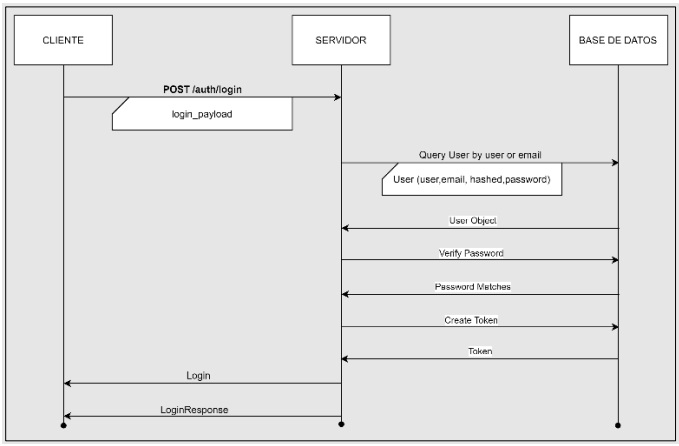


DIAGRAMAS DE SECUENCIA

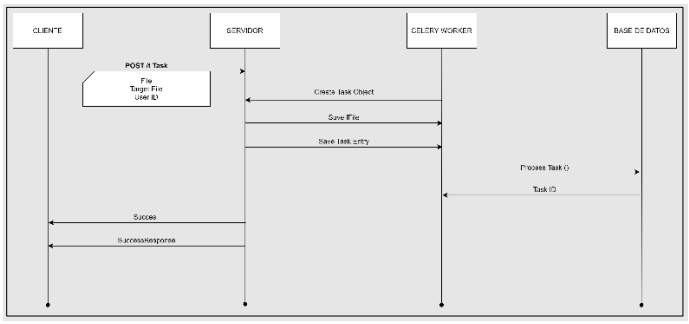
SIGN UP



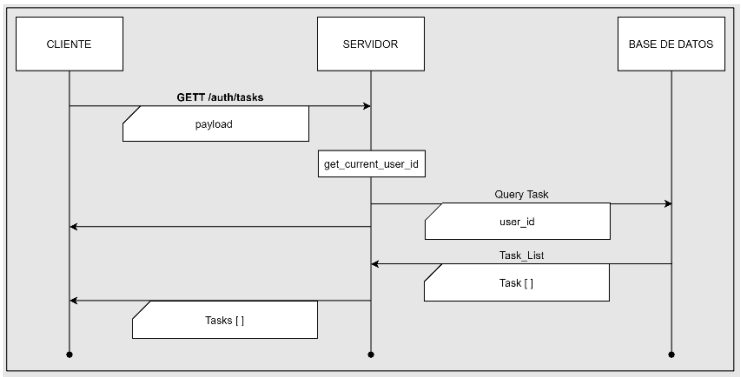
LOGIN



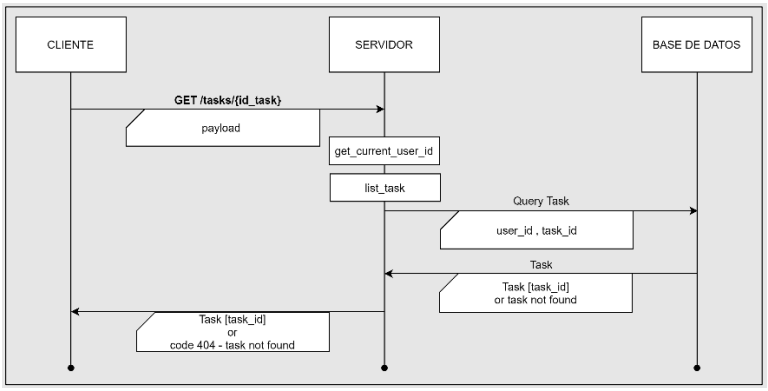
TASKS – CREAR TAREA



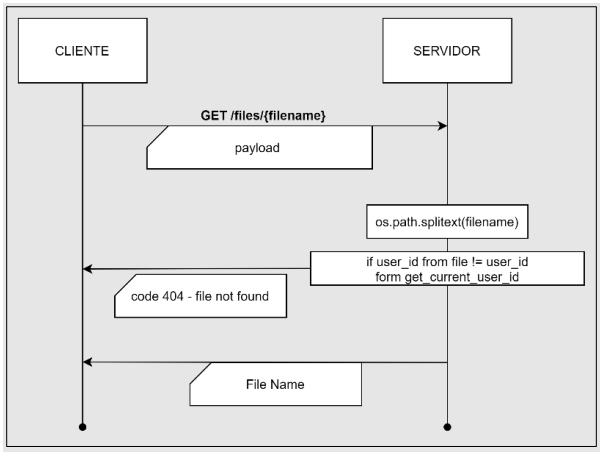
TASKS – RECUPERAR LISTA DE TAREAS



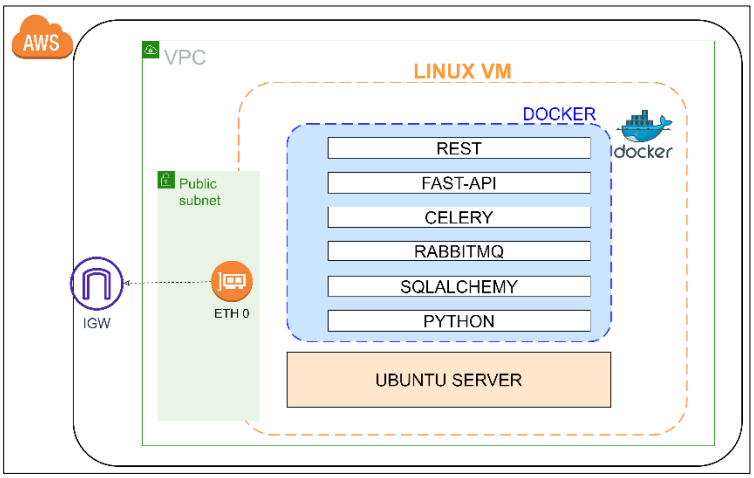
TASK/IDTASK



FILES



DIAGRAMAS DE DESPLIEGUE



Inicialmente se utiliza una máquina virtual con Ubuntu Server como SO y el cual se despliega en una Subnet pública dentro de una VPC en AWS. Para dar acceso a la aplicación se tiene un IGW de AWS el cual es la ruta por defecto de la VPC. Se utilizó Docker Compose para despliegue de todos los componentes necesarios para la ejecución de la aplicación.

CONSIDERACIONES

Formato de entrada permitidos: No hay restricciones de formato de entrada.

Formatos de compresión permitidos: ZIP, TAR.GZ, TAR.BZ2

Cada reinicio de la maquina en AWS Academy cambia la IP publica para acceder desde internet.

Requerimientos de seguridad de la contraseña al momento de solicitar la creación del usuario:

- La contraseña debe contener más de 8 caracteres.
- La contraseña debe contener caracteres especiales.
- La contraseña debe contener un número.
- La contraseña debe contener una letra mayúscula.

CONCLUSIONES

- Se construyó una aplicación que permite la compresión de archivos a través de un sistema de encolamiento entre Celery y RabbitMQ. Esta aplicación permite una mayor escalabilidad y mejor tolerancia a fallos ya que las tareas son procesadas de manera asíncrona y distribuida, permitiendo a varios usuarios acceder y utilizar la aplicación de manera concurrente.
- La base de datos de Docker utilizada para esta aplicación web utiliza los recursos de la máquina donde está desplegada la aplicación. Por tal motivo se propone migrar esta base de datos a alguna de las ofrecidas por el proveedor cloud para mejorar la escalabilidad. Asimismo, se propone utilizar un servicio de almacenamiento de archivos tal como S3 para el almacenamiento de los archivos que sube el archivo y los que son comprimidos.