Sistema Conversión Cloud - Entorno Tradicional

Objetivo

- Documentar la estrategia de arquitectura de proyecto del curso.
- El proyecto se centra en la escalabilidad como atributo de calidad de las aplicaciones. A través del proyecto se podrá identificar e implementar las mejores prácticas para diseñar y desarrollar aplicaciones en la nube. Mediante una aplicación desarrollada para un entorno de infraestructura tradicional y de forma incremental se integrarán los elementos arquitectónicos de la nube, midiendo rendimiento y las cargas de trabajo para soportar millones de usuarios.

Descripción

La documentación de la estrategia de arquitectura es fundamental para un proyecto de computación en la nube, dado que tiene como beneficios mejorar la productividad, reducir el riesgo técnico, evitar la deuda técnica, mejorar el conocimiento sobre el negocio, mejorar la integración del desarrollo con la operación, reducir el tiempo de desarrollo y mejorar la comunicación entre los miembros del equipo.

La presente guía tiene como propósito dar las directrices necesarias para la construcción del documento de arquitectura durante el desarrollo del proyecto del curso. Para esto se les proporcionará el documento de arquitectura de referencia de la aplicación, el cual tendrán que evolucionar teniendo en cuenta los nuevos requerimientos y estructura sugerida a continuación.

Estructura del Documento

Información del equipo de desarrollo

Proyecto	**cloud conversion tool**
Grupo	Grupo 22
Integrantes	**Nombre**
	Ginett Rosales
	Maria Angelica Gaitan Sanchez
	Daniel Quiceno
	Carlos Pinto

Sistema Conversión Cloud - Entorno Tradicional

Contexto del problema

Contexto del problema

El propósito de esta herramienta es ofrecer a los usuarios que requieren con más frecuencia un conversor de archivos, con el fin de cambiar el formato de archivos multimedia en línea de un formato a otro, seleccionando el formato de audio: MP3 - ACC - OGG - WAV – WMA. Sin embargo, cada vez menos los usuarios desean instalar aplicaciones en sus equipos y recurren con más frecuencia a la web.

El modelo general de funcionamiento de la aplicación se basa en crear una cuenta en el portal web y acceder al administrador de archivos y al acceder a ella y elegir entre las opciones disponibles, en pocos pasos el usuario obtenga el recurso deseado. Por ejemplo, si el usuario requiere convertir un video, simplemente debe click en la opción conversor de multimedia, seleccionar Convertir a y nos aparece una página en la que subir el archivo original.

Requerimientos

Endpoint 1: [POST] /api/auth/signup

Permite crear una cuenta de usuario, con los campos usuario, correo electrónico y contraseña. El usuario y el correo electrónico deben ser únicos en la plataforma, la contraseña debe seguir unos lineamientos mínimos de seguridad, además debe ser solicitada dos veces para que el usuario confirme que ingresa la contraseña correctamente

Endpoint 2: [POST] /api/auth/login

Permite recuperar el token de autorización para consumir los recursos del API suministrando un nombre de usuario y una contraseña correcta de una cuenta registrada.

Endpoint 3 [GET] `/api/tasks`

Permite recuperar todas las tareas de conversión de un usuario autorizado en la aplicación.

Endpoint 4 [POST] `/api/tasks`

Permite crear una nueva tarea de conversión de formatos. El usuario requiere autorización.

Endpoint 5: [GET] `/api/tasks/<int:id_task>`

Permite recuperar la información de una tarea en la aplicación. El usuario requiere autorización

Endpoint 6: [PUT] `/api/tasks/<int:id_task>`

Permite actualizar la información de una tarea en la aplicación, le facilita al usuario actualizar el formato de conversión de un archivo ya disponible en la aplicación. El usuario requiere autorización.

Endpoint 7: [DELETE] `/api/tasks/<int:id_task>`

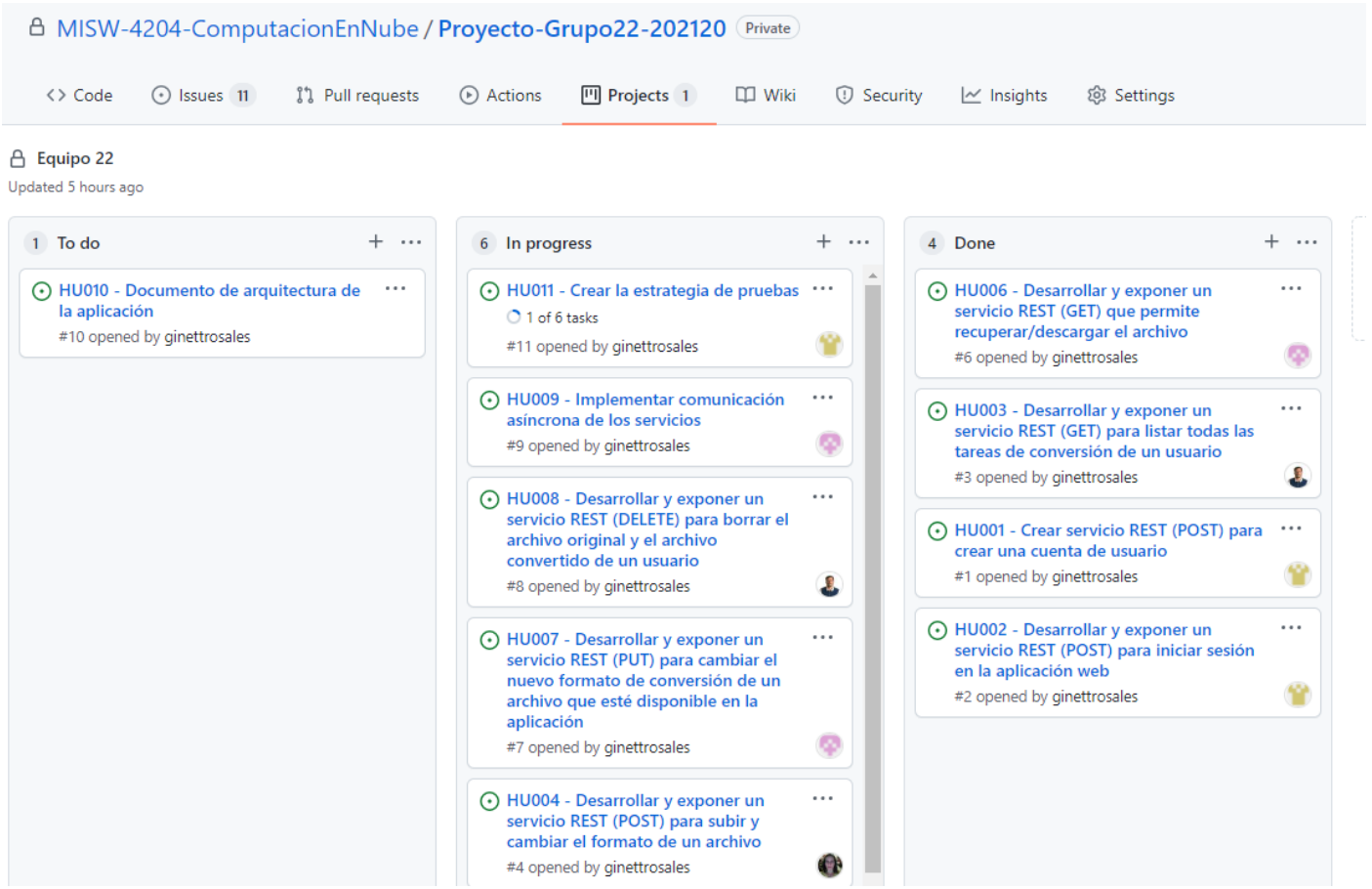
Permite eliminar una tarea en la aplicación. El usuario requiere autorización

Endpoint 8: [POST] `/api/files/<filename>`

Permite recuperar el archivo original o procesado

Planificación **https://github.com/MISW-4204-ComputacionEnNube/Proyecto-Grupo22-202120/projects/1**

Se realizó la planificación y distribución de trabajo mediante un tablero, y asignación de tareas a cada miembro del equipo.



Sistema Conversión Cloud - Entorno Tradicional

Infraestructura requerida para el despliegue

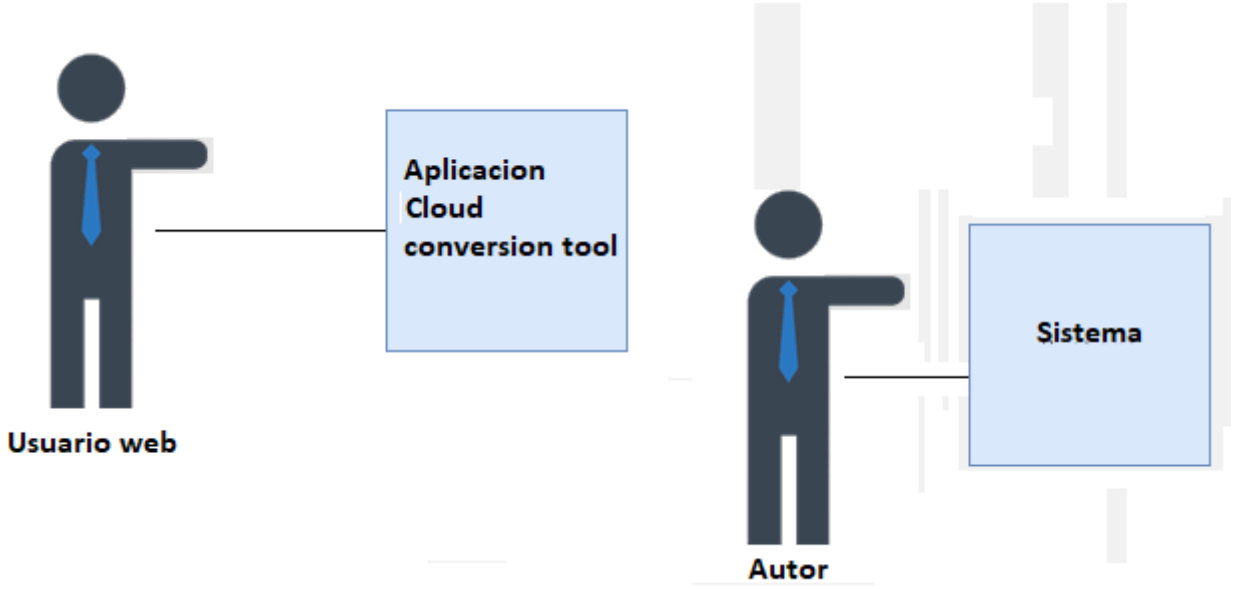
En este proyecto se planea utilizar la siguiente infraestructura y servicios:

- Una máquina virtual como servidor único, aprovisionada en la infraestructura de cómputo de la Universidad de los Andes (ISCloud – Nube Privada).
- La máquina virtual cuenta con las siguientes especificaciones técnicas 1 vCPU, 1 GiB RAM, 20 GiB en almacenamiento.
- Sistema Operativo Ubuntu (GNU/Linux) 20.04 LTS. Cada grupo deberá configurar su ambiente con el servidor de aplicaciones, librerías, base de datos y en general todos los componentes necesarios para el despliegue correcto de la aplicación.
- El despliegue de todos los componentes requeridos por la aplicación

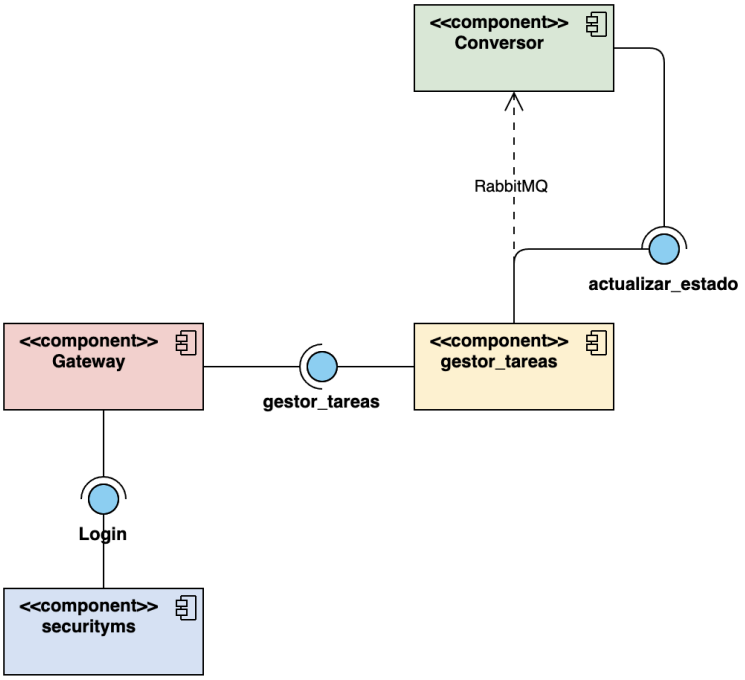
Estrategia de arquitectura

La estrategia de arquitectura para el proyecto contempla las siguientes vistas y modelos:

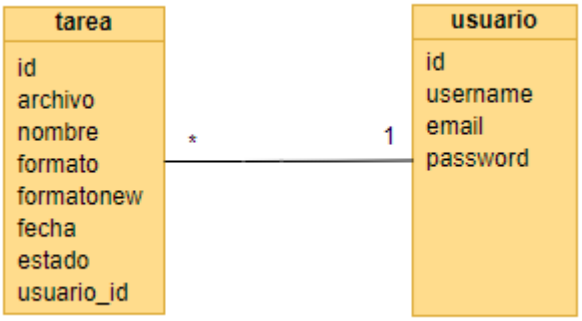
- **Vista de contexto**
  - Modelo de contexto
  - Identifica los tipos de usuarios del sistema.
  - Identifica los sistemas de información con los que debe interactuar el sistema a diseñar.
  - Frontera del sistema: deja claro qué se va a diseñar como parte de la arquitectura y qué no hace parte del diseño.



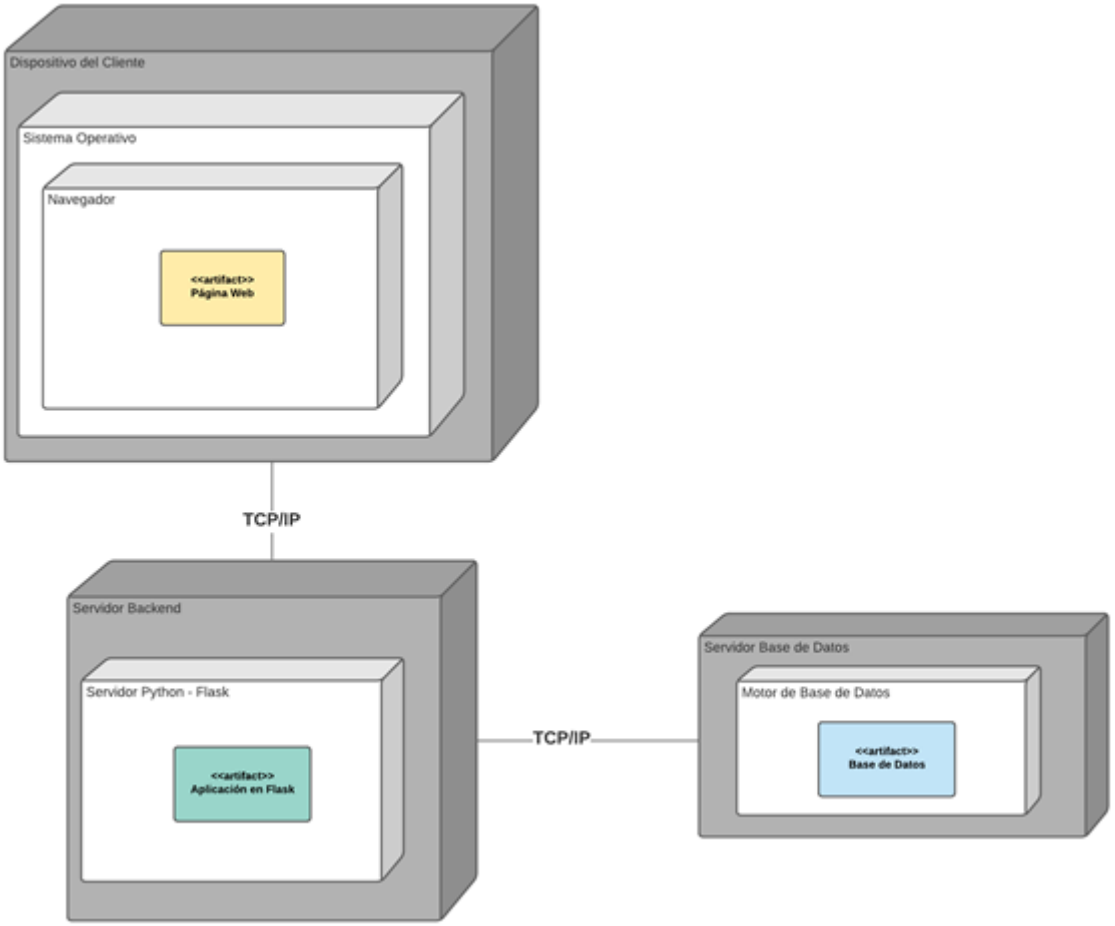
- **Vista funcional**
  - Modelo componente-conector

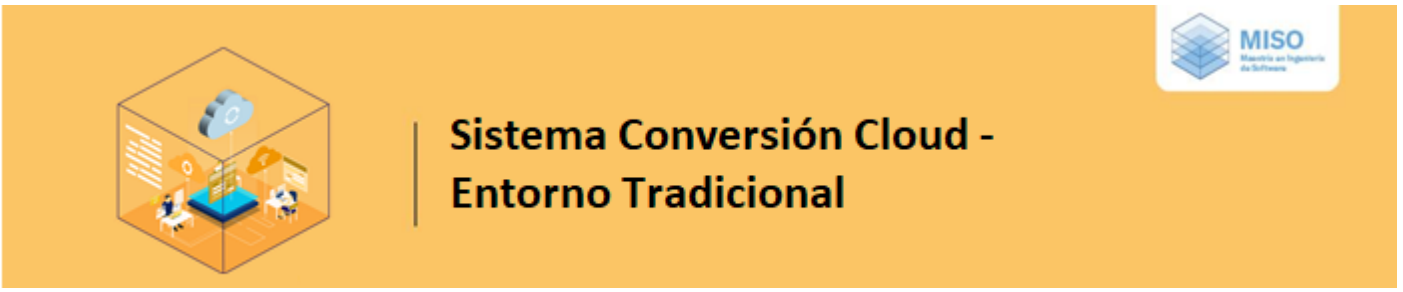


- Vista de clases



- Vista de despliegue





Tecnologías para el desarrollo

Se requerirá para el desarrollo del proyecto:

- **Python 3.8 o superior.**
- **Flask Framework 1.1.x:** micro framework web.
- **Flask SQLAlchemy:** una extensión de Flask que agrega soporte para el ORM SQLAlchemy, mapeador relacional de objetos que simplifica la interacción con una base de datos SQL.
- **Flask RESTful:** una extensión de Flask para desarrollar API REST con un patrón de diseño orientado a objetos.
- **Flask Marshmallow:** una extensión de Flask que se integra con Marshmallow, una biblioteca de Python para la serialización de objetos.
- **Celery:** Es una biblioteca de Python para gestionar colas de tareas distribuidas, es decir, nos permite trabajar con diferentes tareas y distribuirlas en procesos.
- **Redis o RabbitMQ:** servicios de mensajería que actúan como intermediarios; los brokers enrutan, agregan y permiten crear servicios de publicación/subscripción, comúnmente llamados servicios pubsub.
- **Flask JWT Extended:** agrega soporte para el uso de JSON Web Tokens (JWT) a Flask para proteger las vistas, creación de tokens a partir de objetos complejos u objeto complejo a partir de tokens recibidos, actualizar tokens, entre otras funciones.
- **Werkzeug:** biblioteca completa de aplicaciones web WSGI.
- **PostgreSQL:** Motor open source de base de datos SQL.
- **Gunicorn:** servidor HTTP WSGI para Python y ambientes Unix.
- **Nginx:** servidor web HTTP de código abierto. Ofrece servicios como proxy inverso, balanceador de carga HTTP y proxy de correo electrónico para IMAP, POP3 y SMTP. Se utiliza en conjunto con Gunicorn para desplegar en producción aplicaciones web en Flask.
- **Git:** para el control de versiones
- **AWS:** Servicios de nube pública.

Verificación funcional

De los requerimientos de la aplicación, junto a la documentación en POSTMAN:

- requerimientos de la aplicación.
  https://github.com/MISW-4204-ComputacionEnNube/Proyecto-Grupo22-202120/blob/main/README.md

- documentación en POSTMAN
  https://documenter.getpostman.com/view/15434532/UV5agbwB