Universidad de los Andes
Facultad de Ingeniería

Desarrollo de Soluciones Cloud – ISIS 4426,

Proyecto 4.

DOCUMENTACIÓN TÉCNICA

Grupo 5

María Catalina Ibáñez, Jairo Céspedes, David Saavedra, José Manuel Moreno

Contenido

Proyecto 4 1
Descripción. 3
Requerimientos. 3
Archivos: 4
Diagrama de Arquitectura 5
Base de datos 5
Diagrama de componentes 7
Información de despliegue 9
Conclusiones 10

Ilustración 1. Diagrama de arquitectura 5
ilustración 2. Diagrama de relaciones base de datos 7
ilustración 3. Diagrama de componentes 9
ilustración 4. Información de despliegue servicios en Cloud Run 9
ilustración 5 información de despliegue Cloud Functions 10

Descripción.

Enlace del repositorio: https://github.com/Cloud-G05/project1.git
Enlace de la video sustentación: Sustentación proyecto 4 - Cloud (youtube.com)

Este documento detalla la arquitectura técnica implementada en el proyecto 4. Manteniendo la esencia de nuestra tercera entrega, hemos evolucionado la aplicación web para ofrecer un servicio gratuito de conversión de archivos de múltiples formatos (.odt, .docx, .pptx, .xlsx) a PDF. Dicha aplicación se destaca por su arquitectura asíncrona, la cual permite la ejecución de tareas en batch sin que los usuarios necesiten esperar en línea. En lugar de ello, se les notifica mediante un cambio de estatus cuando la conversión ha sido completada.

La arquitectura de la solución se fundamenta en servicios de la nube, concretamente a través de Google Cloud Services. Esta incluye los componentes de backend implementados en un servicio de Cloud Run el cual está vinculado directamente con el repositorio de Github. Lo anterior también se hizo para el frontend (en un servicio de Cloud Run distinto). Por otro lado, se cuenta con un bucket en Cloud Storage para el almacenamiento de archivos, Cloud Pub/Sub para la orquestación de mensajes y Cloud SQL para la gestión de la base de datos.

Además, el worker está dividido en dos partes. Por un lado, se hizo una función en Cloud Functions la cual se activa cuando se publica un mensaje en un tema específico de Cloud Pub/Sub. Esta función al ser activada recibe el mensaje y hace una petición a un endpoint de una api que se encuentra desplegada en un servicio de Cloud Run. Esta petición le pide al servicio convertir un archivo específico a pdf, por lo que la tarea de conversión se ejecuta en el servicio de Cloud Run. Finalmente, el archivo convertido se monta a Cloud Storage y el estatus de la tarea es actualizado en Cloud SQL.

Un avance significativo en esta fase es la transición completa a servicios nativos de Google Cloud, respondiendo así a una arquitectura nativa de servicios altamente escalables y bajo demanda.

Requerimientos.

Para esta entrega, se ha ampliado el stack tecnológico, incluyendo servicios adicionales de GCP y una arquitectura serverless, detallada a continuación:

Google Cloud Run (Front): Los componentes del frontend ahora se alojan nativamente en un servicio de Google Cloud Run.

Google Cloud Run (Back): De igual manera, los componentes del backend ahora se alojan nativamente en un servicio de Google Cloud Run.

Google Cloud SQL: La gestión de la base de datos se realiza a través del servicio Google Cloud SQL, utilizando PostgreSQL para optimizar el almacenamiento y acceso a los datos.

Google Cloud Pub/Sub: Se ha integrado este servicio para la orquestación de mensajes entre el backend y el worker.

Google Cloud Storage: Este componente almacena los archivos originales y convertidos que son consumidos por los demás servicios.

Google Cloud Functions (Worker): La función del worker ahora se encuentra en un servicio exclusivo de Google Cloud Functions. Posteriormente comunica las peticiones mediante HTTPS al worker alojado en Google Cloud Run.

Google Cloud Run (Worker): Recibe la petición HTTPS desde la función del worker en Cloud Functions para posteriormente hacer la conversión de un archivo, montarlo a Cloud Storage y cambiar el estatus de la tarea en Cloud SQL.

Esta arquitectura propuesta busca no solo cumplir con los requerimientos técnicos del proyecto sino también ofrecer una plataforma robusta, escalable y de alta disponibilidad para el procesamiento asincrónico de conversiones de archivos, facilitando así una mejor experiencia de usuario y un desarrollo sostenible del proyecto en el entorno de la nube.

Archivos:

Este conjunto de diagramas detalla la arquitectura de la aplicación, destacando tanto la estructura de los servicios de GCP así como la interacción entre los componentes del software.

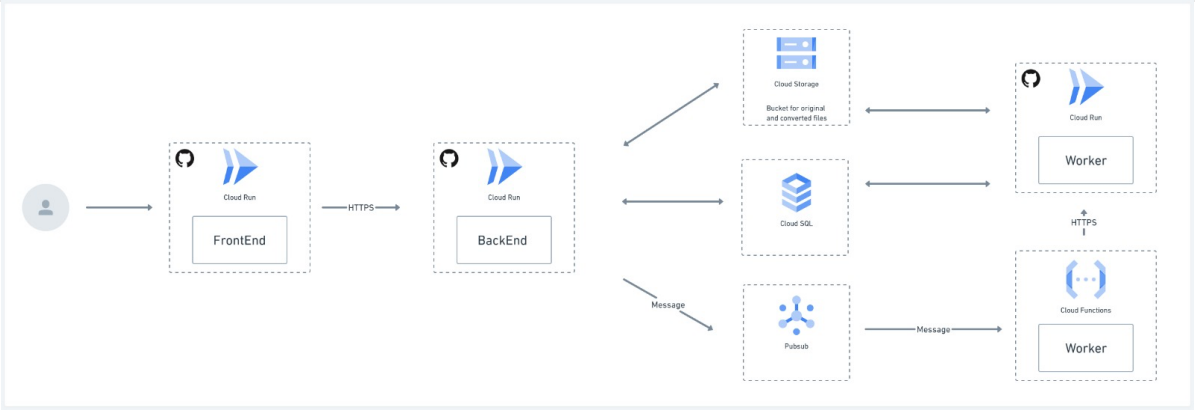

Ilustración 1. Diagrama de Arquitectura

Diagrama de Arquitectura

En contraste con entregas previas, los componentes son desplegados en su totalidad mediante el uso de servicios nativos de la nube, específicamente de Google Cloud Services.

Base de datos

La base de datos mantiene la misma estructura construida para la entrega 1, la misma se detalla a continuación:

Entidades Principales:

1. **User (Usuario):**
    - **username: str** - El nombre de usuario, el cual es único para cada usuario.
    - **email: str** - La dirección de correo electrónico del usuario, utilizado como identificador único para cada usuario.
    - **password: str** - La contraseña del usuario para autenticación.

Cada usuario puede tener asociadas múltiples tareas de conversión de archivos, lo que refleja una relación uno a muchos con la entidad Task.

2. **Task (Tarea):**
    - **id: str** - Un identificador único para cada tarea de conversión.
    - **name: str** - El nombre o título asignado a la tarea de conversión.

- **original_file_ext: str** - La extensión del archivo original antes de la conversión.
- **file_conversion_ext: str** - La extensión deseada del archivo después de la conversión.
- **available: bool** - Un valor booleano que indica si el archivo convertido está disponible para descarga.
- **status: enum** - Un enumerado que representa el estado de la tarea (por ejemplo, 'UPLOADED' para tareas cargadas y 'PROCESSED' para tareas cuya conversión ha finalizado).
- **time_stamp: datetime** - La fecha y hora en que se creó la tarea.
- **input_file_path: str** - La ruta de acceso al archivo original almacenado.
- **output_file_path: str** - La ruta de acceso al archivo convertido almacenado.

La entidad Task está vinculada a la entidad User, indicando que cada tarea está asociada a un usuario específico.

Relaciones:

- **Relación Usuario-Tarea:** Cada usuario puede tener varias tareas asociadas, pero cada tarea está vinculada a un solo usuario. Esto establece una relación uno a muchos entre User y Task, representada en el diagrama.

Enumeraciones:

- **status: enum** - Define los posibles estados de una tarea con los valores 'UPLOADED' (indicando que el archivo ha sido cargado, pero no procesado aún) y 'PROCESSED' (indicando que el archivo ha sido procesado y está listo para ser descargado).

Diagrama de relaciones

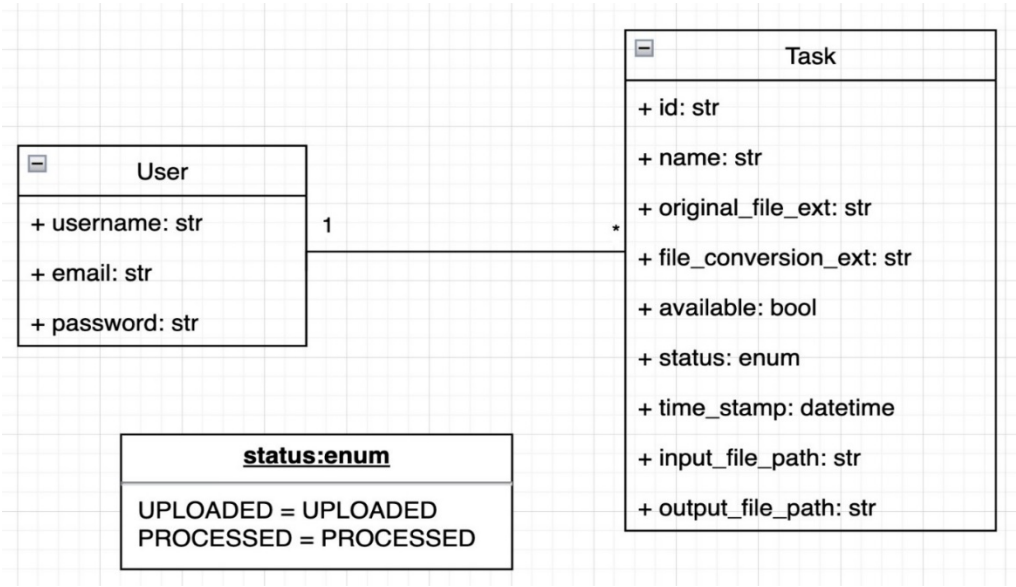

Ilustración 2. Diagrama de relaciones base de datos

No obstante, vale la pena reiterar que dicha estructura de datos ahora se encuentra alojada en el servicio de Google Cloud SQL con PostgreSQL tal como se indicó con anterioridad.

Diagrama de componentes

De igual manera, los componentes utilizados mantienen la misma estructura. El diagrama de componentes proporciona una visión clara de la arquitectura de la solución implementada para el proyecto. Este esquema refleja la interacción entre los distintos componentes de la aplicación, asegurando una operación eficiente y alineada con los estándares actuales de la industria para soluciones web escalables.

Componentes Principales:

1. **Usuario (User)**:
    - Autenticación mediante **JSON Web Tokens (JWT)**: Asegura una comunicación segura y verifica la identidad del usuario al interactuar con la plataforma.
2. **Frontend (React.js)**:
    - Proporciona una interfaz de usuario interactiva y dinámica, desarrollada con el framework moderno de JavaScript, React.js.
    - Envía solicitudes HTTP al backend, autenticadas mediante JWT y maneja las respuestas para reflejar los cambios en la interfaz de usuario.
3. **Backend (FastAPI)**:

- Gestiona las solicitudes HTTP provenientes del frontend, procesando la lógica de negocio y las operaciones de la aplicación.
- Interactúa con la base de datos PostgreSQL para recuperar o almacenar información, con Cloud Pub/Sub para publicar tareas por hacer en forma de mensaje y con Cloud Storage para acceder y almacenar archivos.
- Envía y recibe datos en formato JSON, asegurando un intercambio de datos eficiente y estandarizado.

4. **Worker (Python):**
    - Cuando se publica un mensaje en el tema de Cloud Pub/Sub, se activa la función de Cloud Functions.
    - Envía solicitues HTTP al worker desplegado en Cloud Run solicitando la conversión de un archivo.
5. **Worker (FastAPI):**
    - Realiza la conversión de archivos de un formato dado (.odt, .docx, .pptx, .xlsx) a pdf.
    - Se comunica con la base de datos para actualizar el estado de las tareas asegurando que los usuarios puedan rastrear el progreso de sus conversiones.
    - Interactúa con Cloud Storage para acceder y almacenar archivos en un bucket.
6. **Base de Datos (PostgreSQL)**:
    - Almacena y gestiona los datos de la aplicación, incluyendo información de usuarios y detalles de las tareas de conversión.
    - Recibe consultas del backend y actualiza el estado de las tareas a medida que el worker convierte archivos (realiza tareas).
7. **Almacenamiento de archivos (File Storage):**
    - Almacenamiento de archivos originales cargados por el usuario en un bucket de Cloud Storage.
    - Almacenamiento de archivos convertidos en un bucket de Cloud Storage.
8. **Sistema de mensajería (Pub/Sub):**
    - Almacena los mensajes que publica el backend.
    - Activa la función de Cloud Functions del worker cuando el backend publica un mensaje a un tema específico.

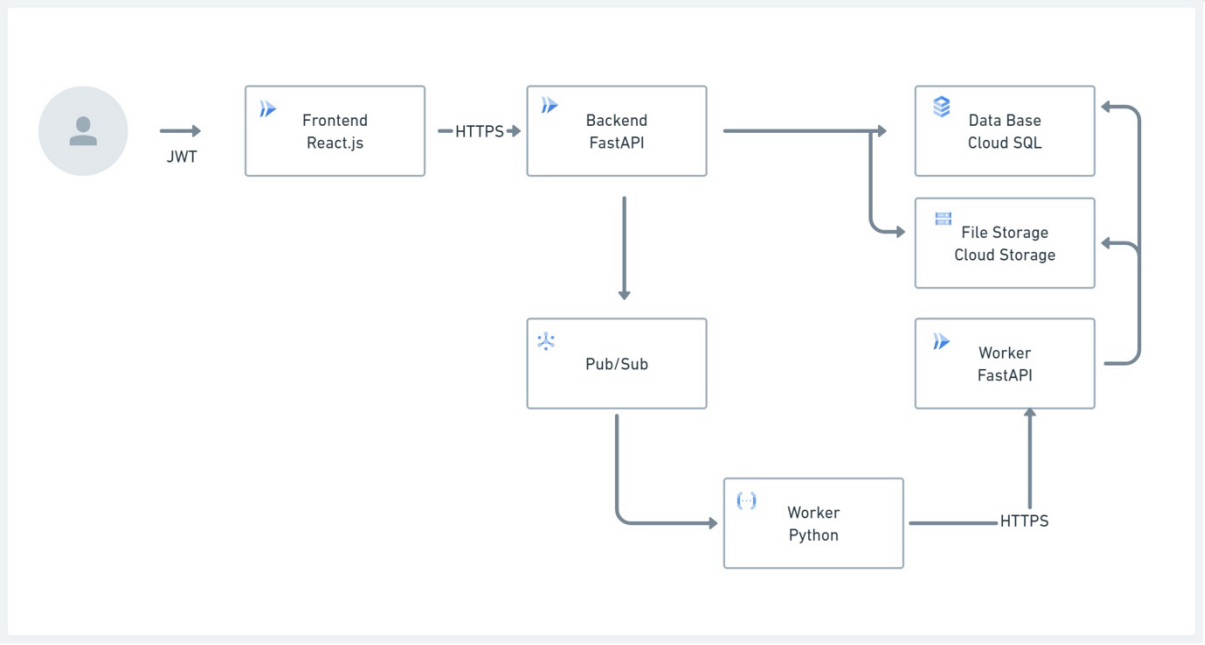

Ilustración 3. Diagrama de componentes

Información de despliegue

La información de los servicios desplegados en Cloud Run es fácilmente identificable en la ilustración 4 que contienen a su vez la información más relevante sobre los componentes desplegados en dicho servicio.

Ilustración 4. Información de despliegue servicios en Cloud Run

Adicionalmente, se realizó el despliegue de la función del worker mediante el servicio de Google Cloud Functions, identificable en la ilustración 5.

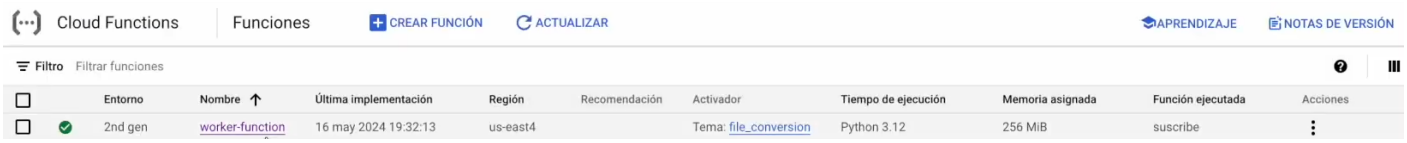

Ilustración 5 Información de despliegue Cloud Functions

Conclusiones

El presente proyecto representa la culminación de diversos esfuerzos realizados durante el curso de desarrollo de soluciones en la nube, demostrando un alto grado de maduración técnica y de negocio. Al proporcionar una infraestructura completamente nativa en Google Cloud Platform, se destacan varias ventajas significativas.

Una de las principales ventajas es el amplio control sobre la respuesta del sistema ante picos de demanda, gracias a la capacidad de ajustar el auto escalado de manera sencilla y rápida. Además, el uso de diversos servicios de Google Cloud permite un monitoreo preciso del desempeño de la plataforma en tareas de procesamiento y almacenamiento de información. Esto proporciona una ventaja competitiva considerable al priorizar acciones para el control de costos y la mejora del rendimiento del producto.

Como resultado, es posible enfocar los esfuerzos en el desarrollo y mejora de los distintos servicios desplegados en la arquitectura en la nube aliviando la carga de la gestión de la infraestructura y su respuesta. Esto permite centrar la atención en el mejoramiento del código y su rendimiento en tareas como la conversión de archivos.

Por último, es importante señalar que los diversos pasos previos para llegar a este punto han sido un insumo invaluable para la comprensión de los distintos requisitos técnicos y financieros al escalar soluciones tecnológicas en la nube. Estos aprendizajes han sentado una base sólida para futuros desarrollos y optimizaciones en el ámbito de soluciones cloud.