**FACULTAD DE INGENIERÍA**

**Departamento de Ingeniería en Sistemas**

**INFRAESTRUCTURA DE E-COMMERCE EN AZURE**

**Ing. Uayeb Caballero Rodríguez**

**Sistemas Expertos**

**IS912 - 0700**

**Estudiante:**

Estiven Josue Mejía Rodríguez - 20201002454

**Ciudad Universitaria**

**Tegucigalpa, Francisco Morazán 29/06/2025**

Índice

[1 Descripción del Sistema E-commerce 3](#_Toc18398)

[2 Servicios identificados a usar 3](#_Toc14398)

[3 Diseño de la solución 4](#_Toc13830)

[3.1 Diagrama de la Arquitectura 4](#_Toc32035)

[3.2 Justificación de los componentes arquitectónicos 4](#_Toc7906)

[3.3 Justificación más detallada del uso de los servicios (opcional) 5](#_Toc27412)

[4 Evidencias de Despliegue 6](#_Toc28194)

[4.1 Despliegue de los servicios de la infraestructura de aplicaciones 6](#_Toc31286)

[4.2 Despliegue de los servicios de la infraestructura de almacenamiento 8](#_Toc29098)

[4.3 Despliegue de los servicios de la infraestructura de analítica 10](#_Toc26096)

[5 Evidencias de Configuraciones y Conexión con los Servicios de Almacenamiento 11](#_Toc12333)

[5.1 Servicio de Bases de datos 11](#_Toc30591)

[5.1.1 Base de datos OLTP 12](#_Toc7645)

[5.1.2 Datawarehouse 13](#_Toc10866)

[5.2 Storage Account 14](#_Toc26516)

[5.3 Storage Account para Queue y Azure Functions 15](#_Toc8635)

[6 Código en Tres Respositorios Independientes 16](#_Toc4111)

[6.1 Enlaces a los Repositorios 16](#_Toc23149)

[6.2 Repositorio Infraestructura de Aplicaciones 16](#_Toc14388)

[6.3 Repositorio Infraestructura de Almacenamiento 17](#_Toc12653)

[6.4 Repositorio Infraestructura de Analítica 17](#_Toc29319)

[7 Reflexiones Finales 17](#_Toc15256)

[7.1 Separación de la Infraestructura en Repositorios 17](#_Toc17202)

[7.2 Sistema implementado para el procesamiento asíncrono de tareas en segundo plano 17](#_Toc11312)

[7.3 Uso de un Datawarehouse y herramienta de orquestación ETL 18](#_Toc14662)

[8 Experiencia Personal Sobre el Proyecto 18](#_Toc9095)

[9 Enlace al portafolio 18](#_Toc13095)

# Desc**ripción del Sistema E-commerce**

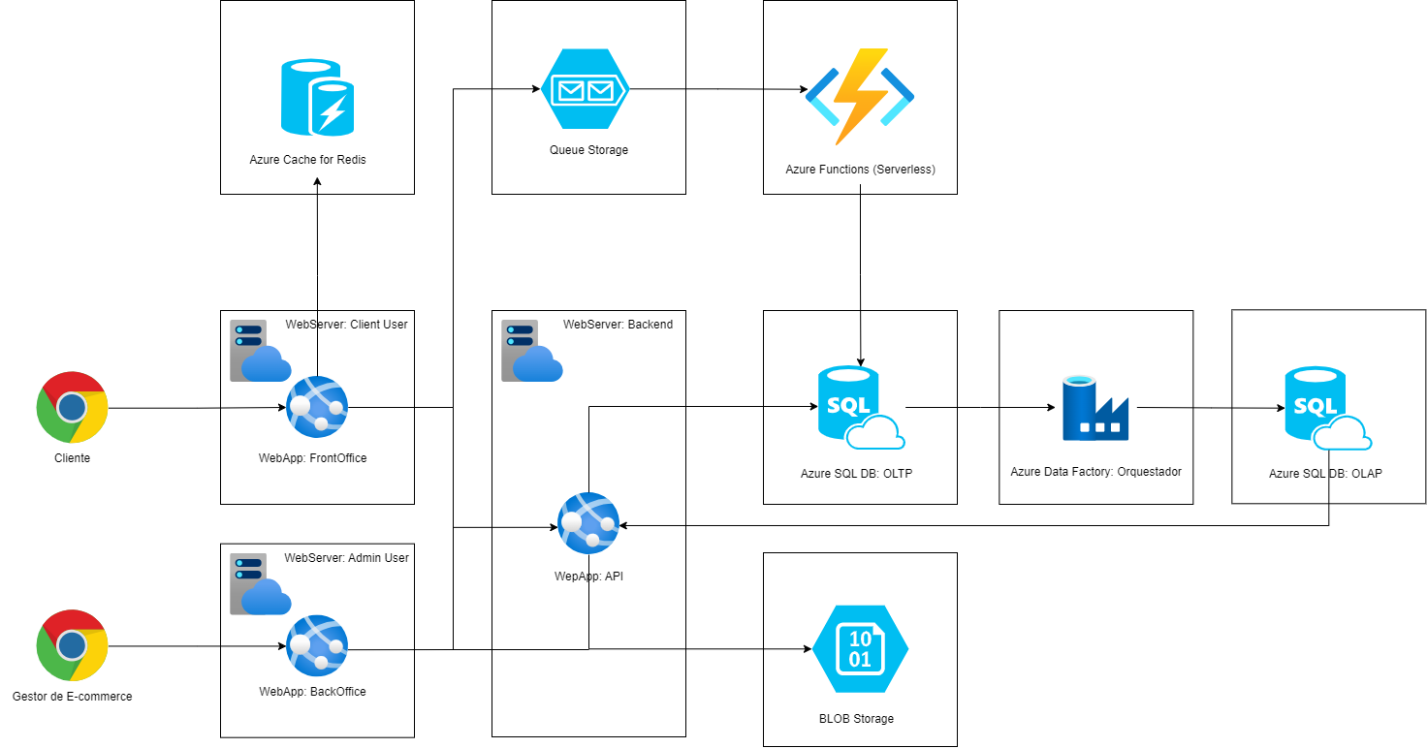
* Infraestructura escalable y robusta en la nube.
* Gestión de operaciones de venta en línea.
* Soportar alto volumen de transacciones.
* Proporcionar experiencia de usuario fluida a través de un portal principal.
* Permitir la gestión interna de productos y pedidos desde un BackOffice.
* Ofrecer capacidades analíticas avanzadas sobre los datos de venta para la toma de decisiones estratégicas.
* Plataforma capaz de manejar tareas intensivas o que no requieren una respuesta inmediata al usuario de forma eficiente en segundo plano.
* Asegurar que la experiencia de usuario no se vea afectada por procesos de larga duración, como confirmación de pedidos, actualización de inventarios complejos o generación de notificaciones masivas.

# Servicios identificados a usar

* Azure App Service - Webapp FrontOffice
* Azure App Service - WebApp BackOffice
* Azure App Service - WebApp API
* Queue storage - Cola de tareas sin respuesta inmediata
* Azure Functions - Serverless para ejecución de tareas en segundo plano
* Azure SQL Database - DB OLTP
* Storage Account - Almacenamiento de multimedia
* Azure Cache for Redis - Almacenamiento de datos volátiles
* Azure SQL Database - DW
* Azure Data Factory - Orquestador del ETL

# Diseño de la solución

## Diagrama de la Arquitectura

****

## Justificación de los componentes arquitectónicos

* **Dos WebServer para UI**. Separar responsabilidades entre un FrontOffice, BackOffice, y mejorar la segurida.
* **DB SQL OLTP.**  Alojar la información transaccional del negocio.
* **Storage Account.** Almacenar archivos (imágenes, pdf, etc) del negocio.
* **Webserver para API.** Maneja el acceso a los almacenamientos de datos del negocio. Una API compartida por ambas UI, ya que la UI del negocio no exige un costo computacional tan grande como para realizar una API única.
* **Queue Storage y Azure functions.** Manejar tareas en segundo plano y que no requieren respuesta inmediata, para evitar una alta carga de trabajo en el servidor que aloja la API.
* **Azure Cache for Redis.** Para alojar data frecuentemente visitada, carrito de compras, sesión del cliente, para lograr una UX fluida.
* **Azure Data Factory y DW.** Un orquestador para manejar el proceso ETL entre la DB OLTP del negocio y el DW que guardará los datos listos para realizar análisis estratégico.

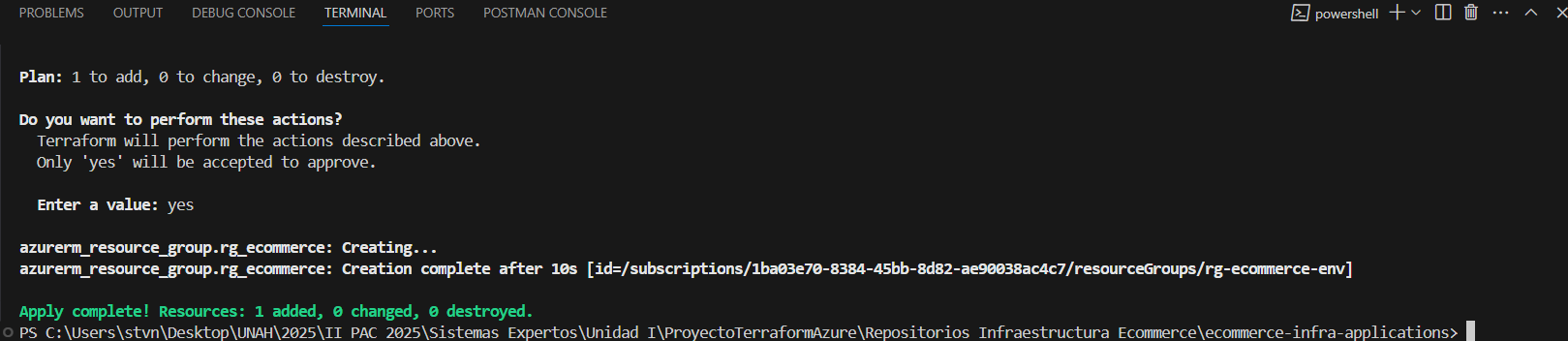
## Justificación más detallada del uso de los servicios (opcional)

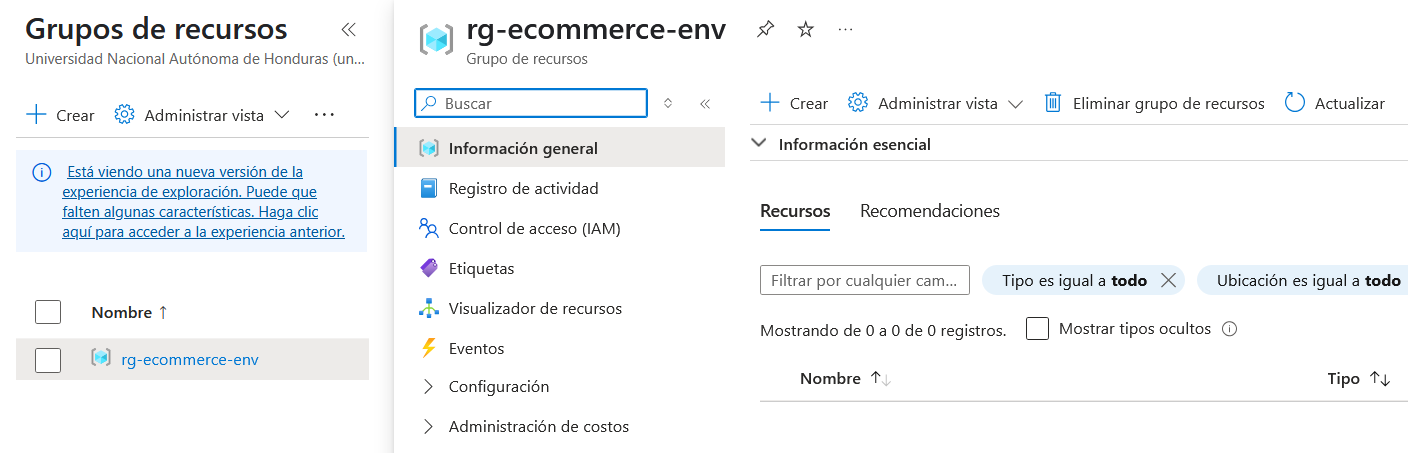
* Cada tipo de usuario (clientes, editores) podrá acceder desde una UI en un servidor dedicado (FrontOffice y BackOffice).
* Las UI toman información de la base de datos SQL por medio de una única API (se ha decidido dejar una sola API, porque la gestión del negocio requiere de un costo computacional mucho menor que el realizado por los clientes).
* La UI de cliente prioriza la eficiencia y experiencia de usuario fluida, entonces la UI de cliente podrá enviar mensajes en Queue storage para pedidos, transacciones, etc., que luego serán ejecutadas en segundo plano por un Serverless.
* UI de cliente usará el almacenamiento en caché (Azure Cache for Redis) para alojar información como el carrito de compras, datos de sesión e información frecuentemente visitada, asegurando una experiencia de usuario sin retardos.
* Para el análisis de negocio se usa el orquestador Data Factory, para hacer el proceso ETL entre la base de datos OLTP y el Datawarehouse, para que luego la información del Datawarehouse pueda ser solicitada por la API o una herramienta externa (Power BI), y que la gestión del negocio pueda realizar el análisis estratégico.

# Evidencias de Despliegue

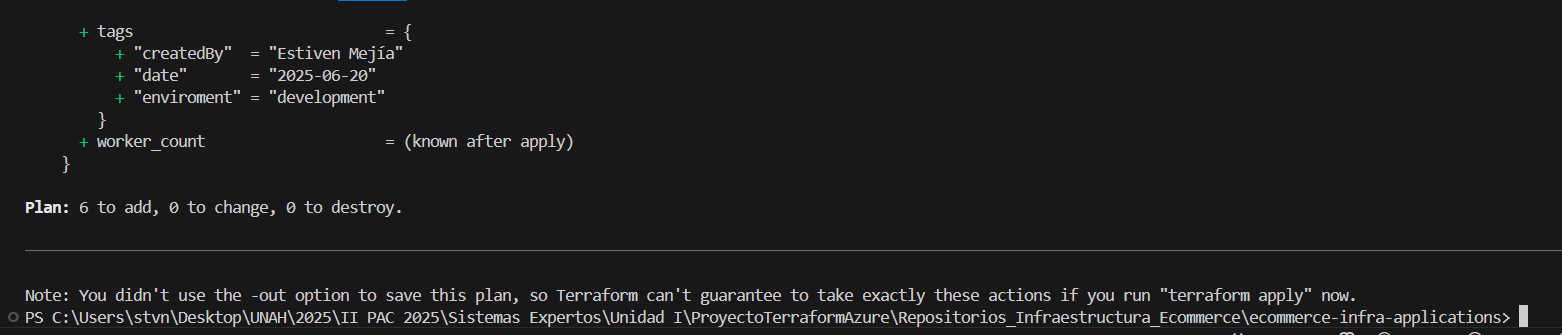
## Despliegue de los servicios de la infraestructura de aplicaciones

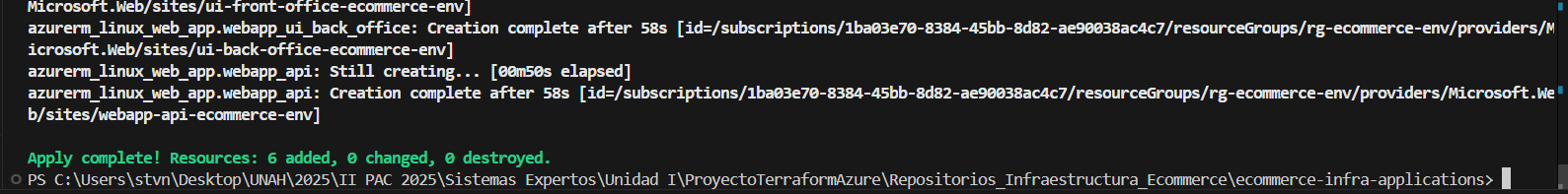
Creación del grupo de recursos:

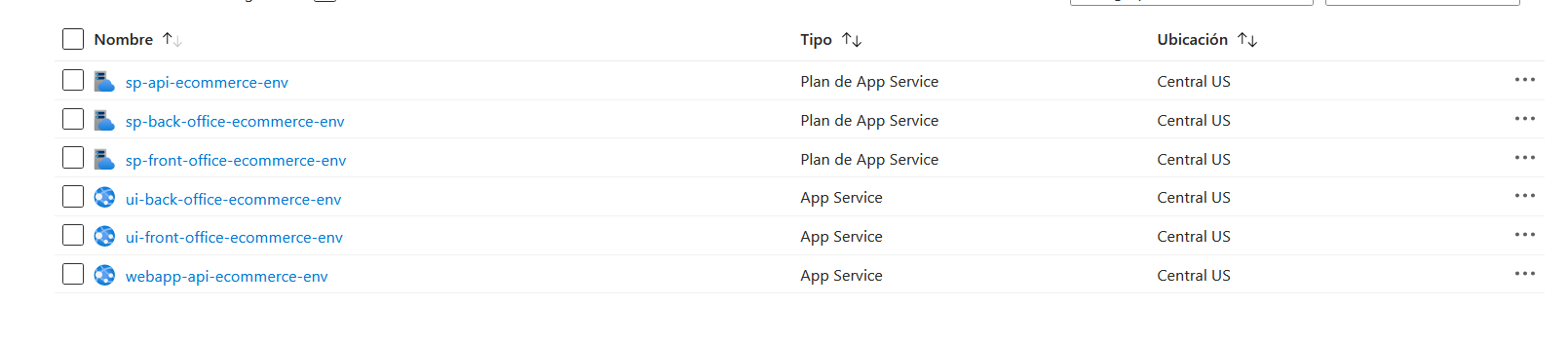




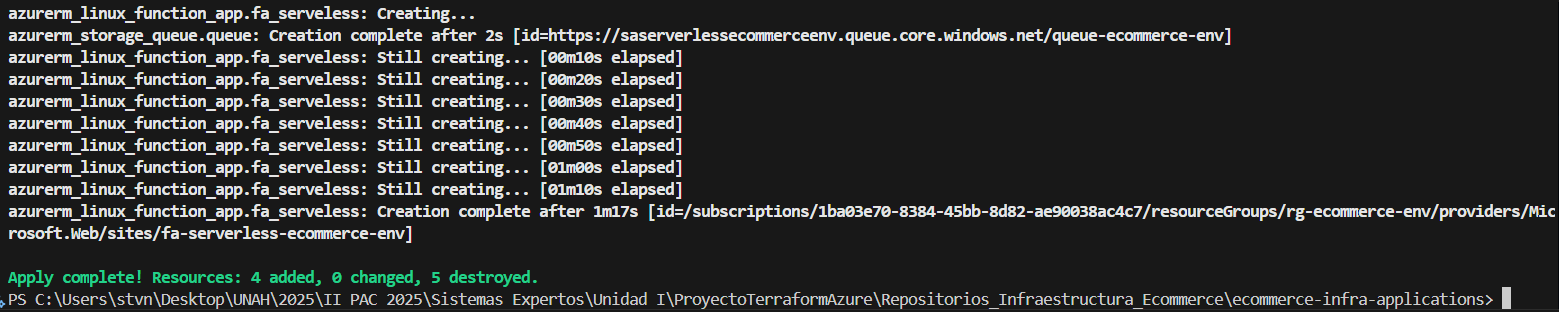
Creación de los tres Service Plan y sus respectivas Web Apps:

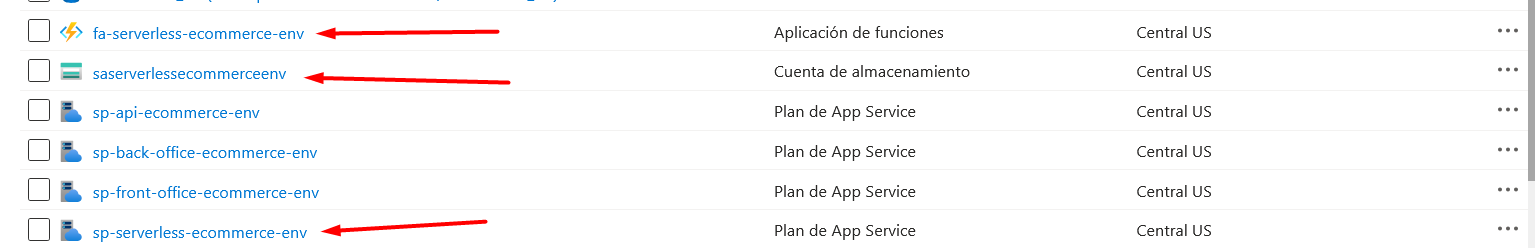






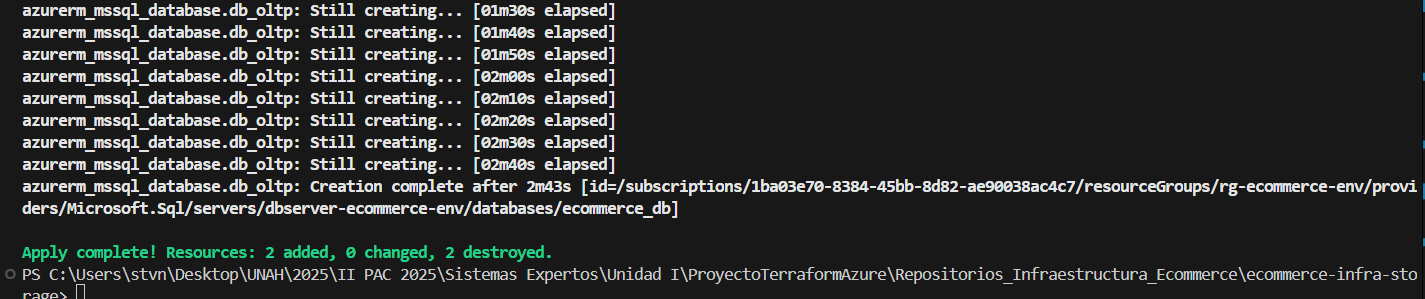
Creación de los servicios Queue Storage y Azure Functions (Servicios Serverless):

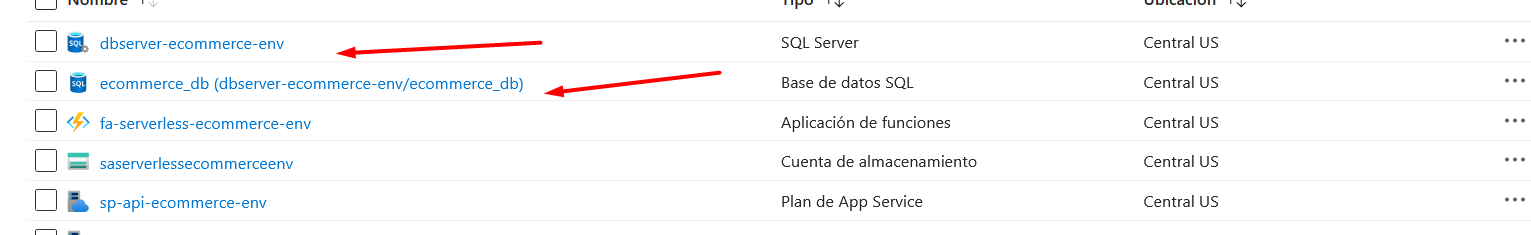




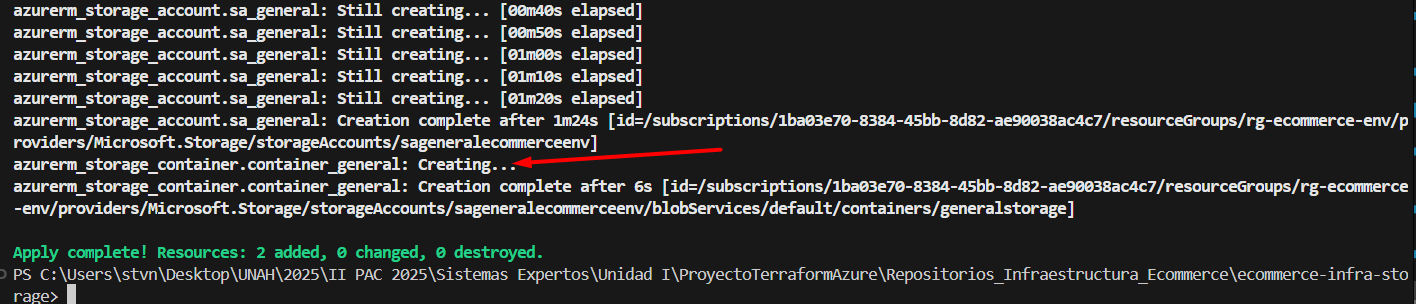
## Despliegue de los servicios de la infraestructura de almacenamiento

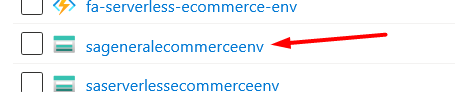
Creación de la DB OLTP



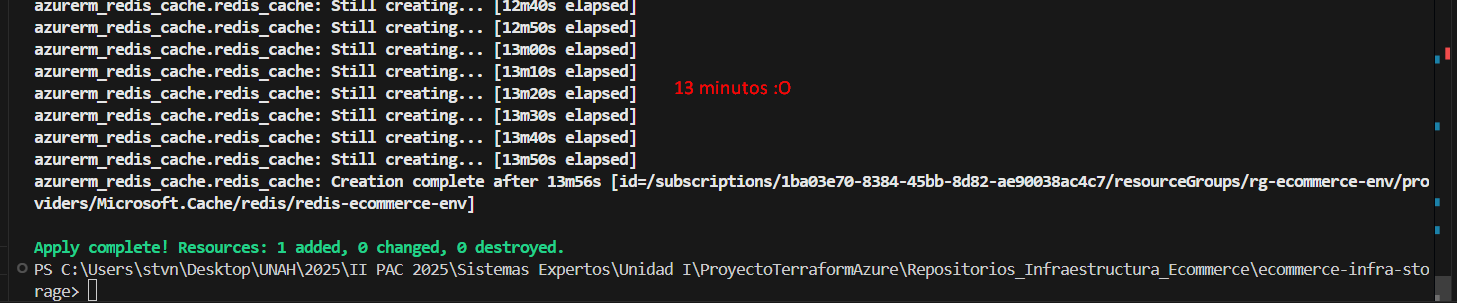


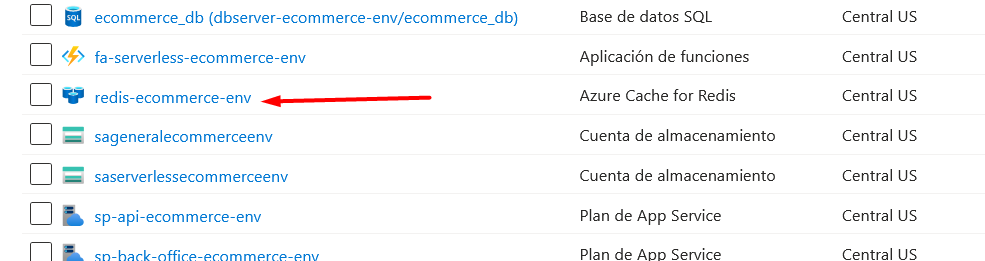
Creación del Storage Account





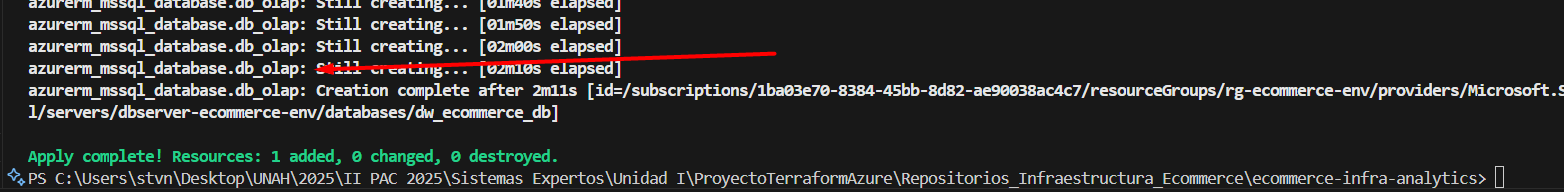
Creación de Azure Redis for Cache:

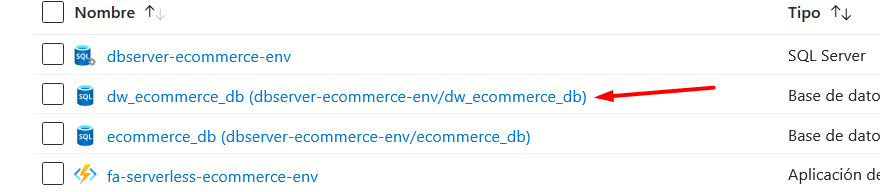




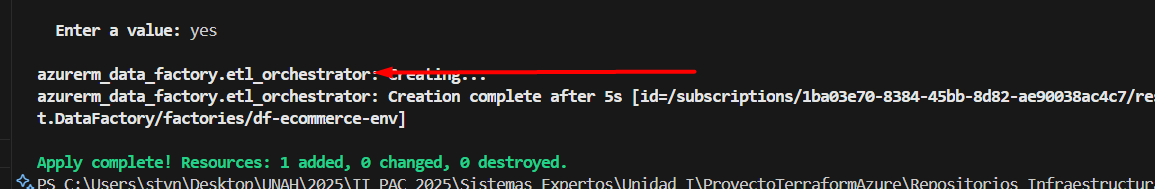
## Despliegue de los servicios de la infraestructura de analítica

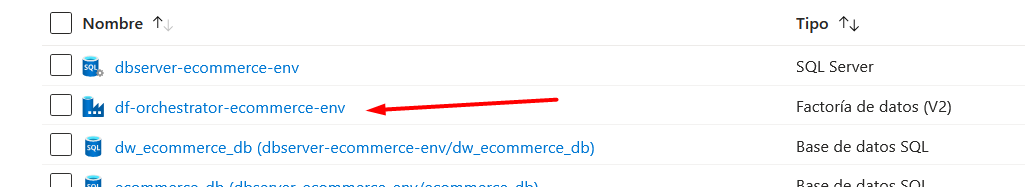
Creación de la DB para Datawarehouse





Creación de Azure Data Factory (Orquestador ETL) [Imagen 16] [Imagen 17]

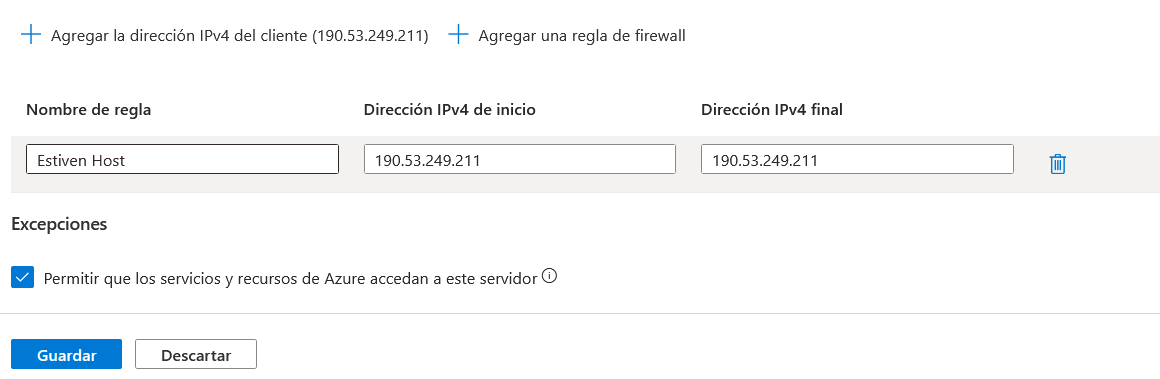
****

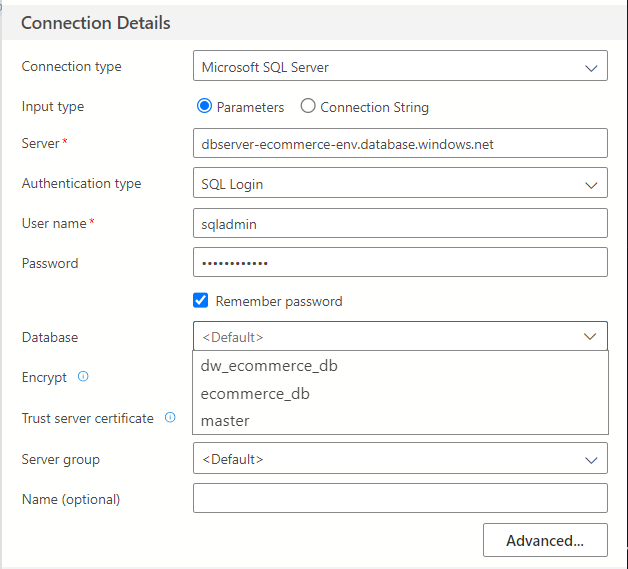
****

# Evidencias de Configuraciones y Conexión con los Servicios de Almacenamiento

## Servicio de Bases de datos

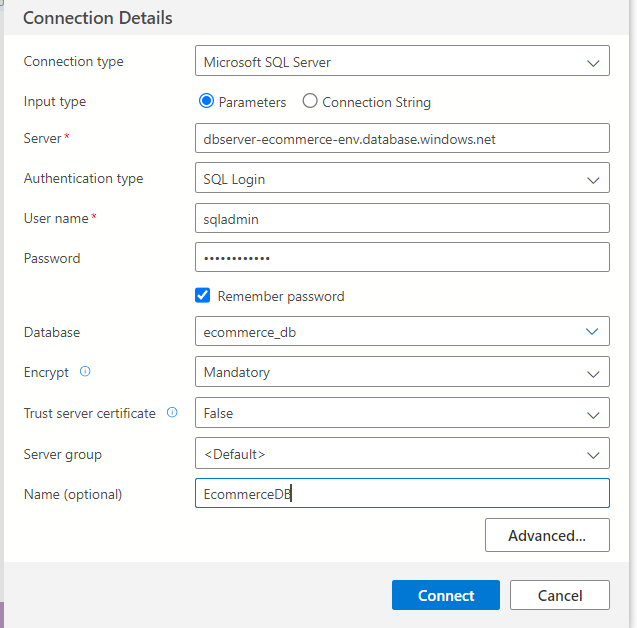
**Configuración de reglas de Firewall del Servicio de DB**

****

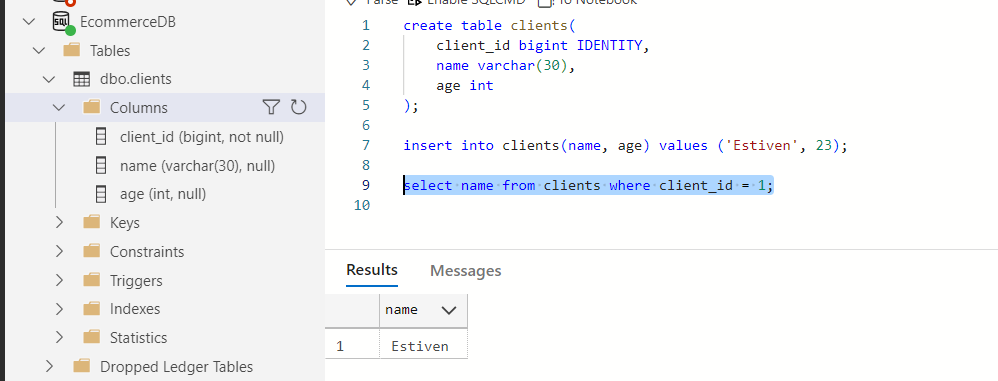
****

### Base de datos OLTP

**Conexión con ecommerce\_db desde Azure Data Studio**

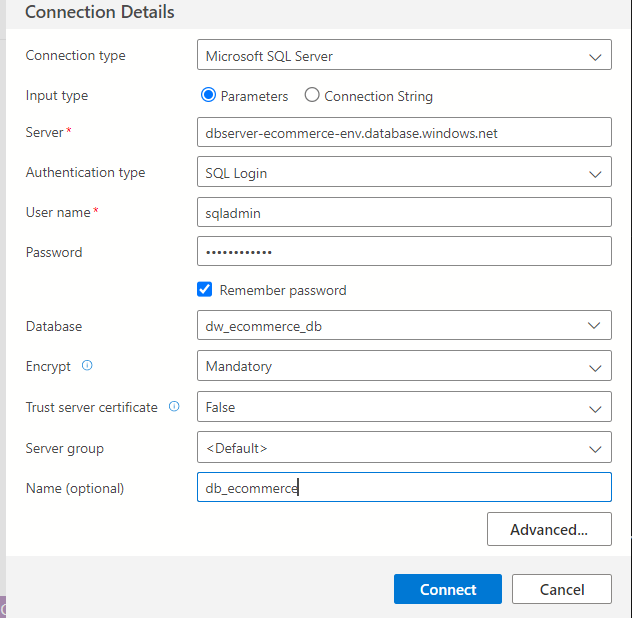
****

**Creación de una tabla en ecommerce\_db**

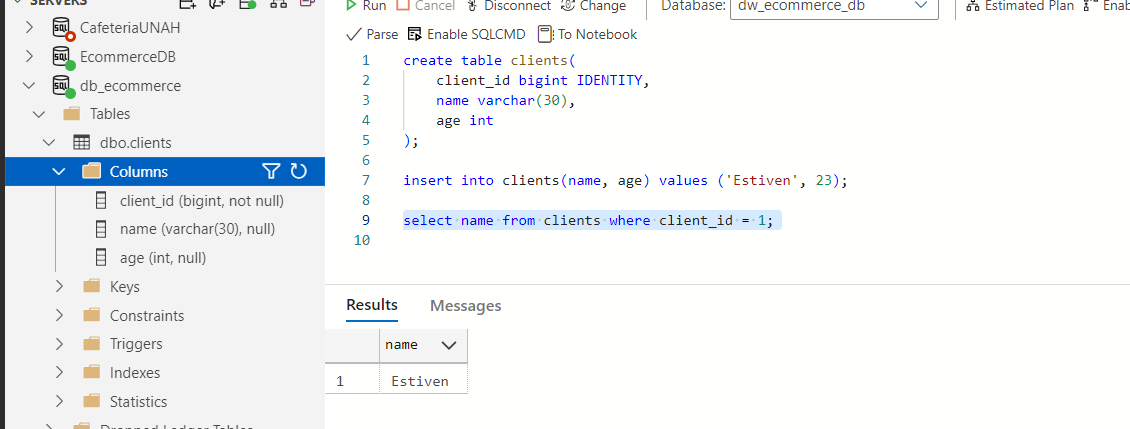
****

### Datawarehouse

**Conexión con dw\_ecommerce\_db desde Azure Data Studio**

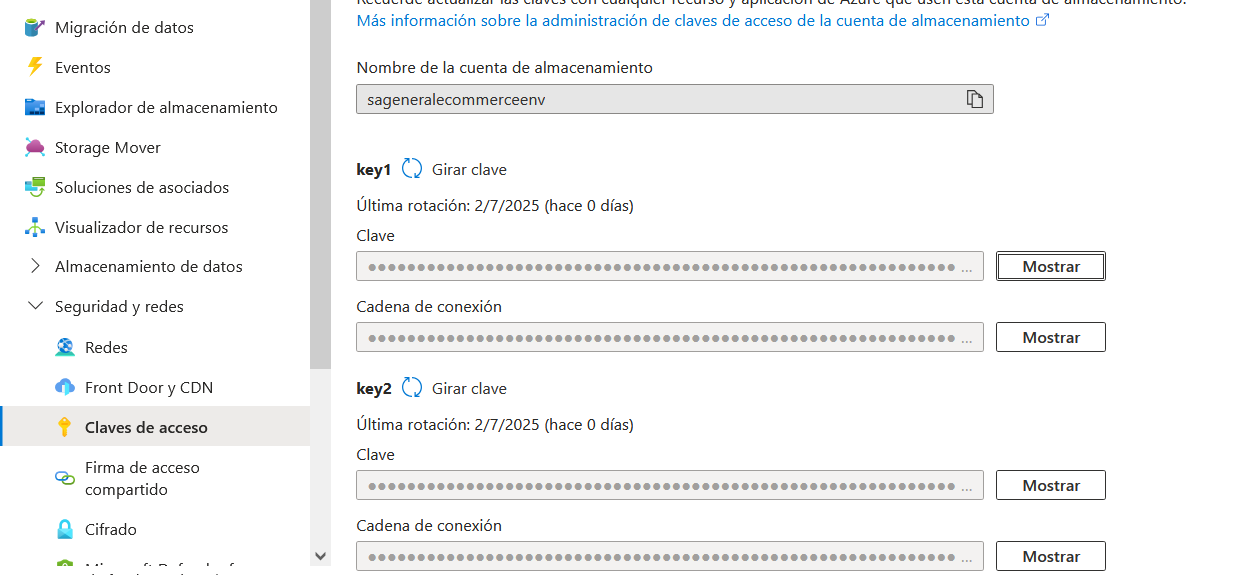
****

**Creación de tabla y campos para dw\_ecommerce\_db**

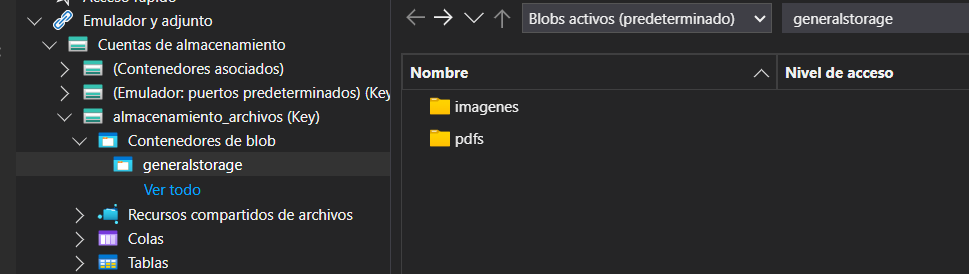
****

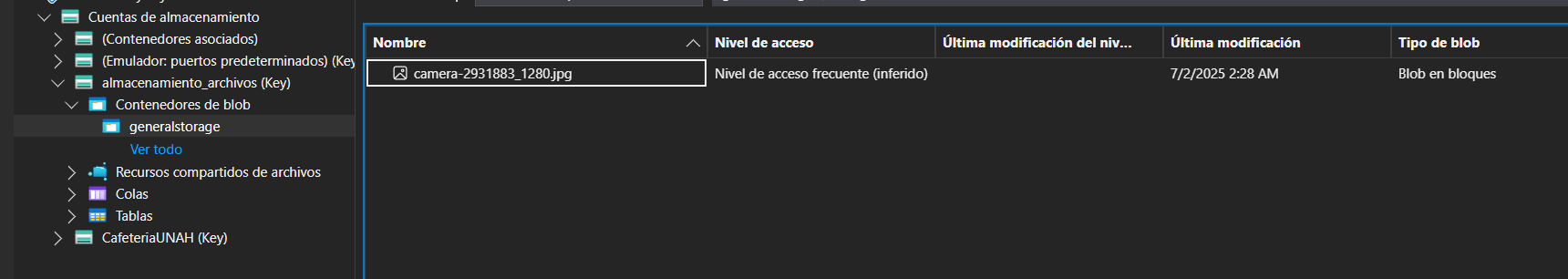
## Storage Account

**Elección de Key para conexión a sageneralecommerceenv**

****

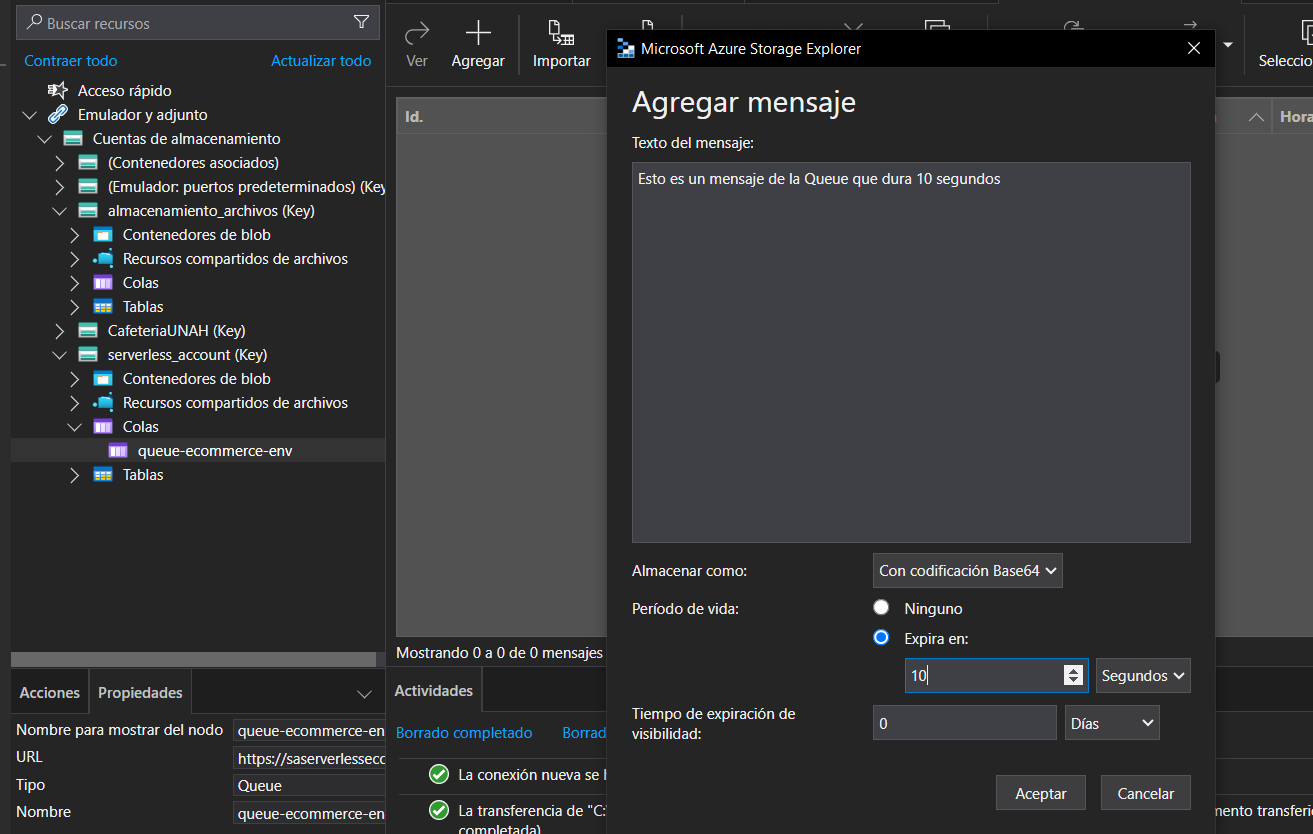
**Conexión e interacción desde Microsoft Azure Storage Explorer**

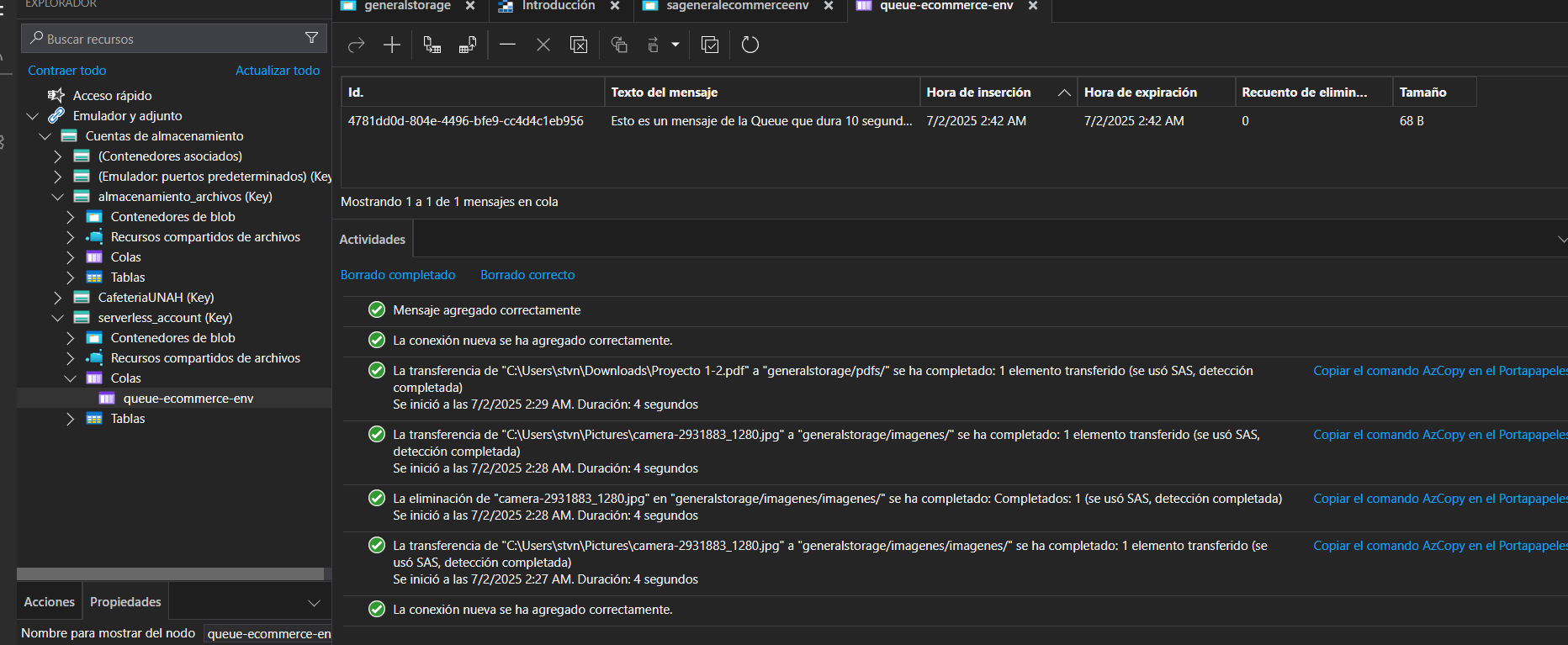
****

****

## Storage Account para Queue y Azure Functions

**Conexión e interacción desde Microsoft Azure Storage Explorer**

****

****

# Código en Tres Respositorios Independientes

## Enlaces a los Repositorios

* [https://github.com/Estvn/ecommerce-infra-applications](https://github.com/Estvn/ecommerce-infra-applications" \t "https://chat.deepseek.com/a/chat/s/_blank)
* [https://github.com/Estvn/ecommerce-infra-storage](https://github.com/Estvn/ecommerce-infra-storage" \t "https://chat.deepseek.com/a/chat/s/_blank)
* [https://github.com/Estvn/ecommerce-infra-analytics](https://github.com/Estvn/ecommerce-infra-analytics" \t "https://chat.deepseek.com/a/chat/s/_blank)

Se ha decidido separar los recursos en módulos del negocio, en tres repositorios diferentes:

* Módulo de aplicaciones
* Módulo de almacenamiento
* Módulo de analítica

## Repositorio Infraestructura de Aplicaciones

* Azure App Service - Webapp FrontOffice
* Azure App Service - WebApp BackOffice
* Azure App Service - WebApp API
* Queue storage - Cola de tareas sin respuesta inmediata
* Azure Functions - Serverless para ejecución de tareas en segundo plano

## Repositorio Infraestructura de Almacenamiento

* Azure SQL Database - DB OLTP
* Storage Account - Almacenamiento de multimedia
* Azure Cache for Redis - Almacenamiento de datos volátiles

## Repositorio Infraestructura de Analítica

* Azure SQL Database - DW
* Azure Elastic Jobs - Orquestador del ETL

# Reflexiones Finales

## Separación de la Infraestructura en Repositorios

* Inicialmente se consideró la idea de hacer la separación estrictamente por los servicios ofrecidos.
* Finalmente los servicios se agruparon en tres categorías de actividades realizadas (UI, almacenamiento y analítica).
* Con este tipo de separación entre los repositorios, habían recursos que se deseaba ser compartidos, pero estaba en repositorios distintos.

## Sistema implementado para el procesamiento asíncrono de tareas en segundo plano

* La aplicación está compuesta por una sola API que es consumida por dos UI.
* Para reducir aún más el consumo de la API por parte del FrontOffice se hace uso de un servicio de Redis Cache.
* También se hace uso de servicio Queue Storage, para guardar en cola las tareas de los usuarios que no exigen una respuesta inmediata.

## Uso de un Datawarehouse y herramienta de orquestación ETL

* Una base de datos OLTP donde se guarda toda la información transaccional del negocio.
* Azure Data Factory (orquestador) y el Datawarehouse para guardar la información procesada para análisis del negocio.
* Los procedimientos almacenados que forman parte del ETL están alojados en el Datawarehouse.

# Experiencia Personal Sobre el Proyecto

El desarrollo de una infraestructura en la nube y la creación de los servicios en Terraform para un proyecto de este tipo, es sencillo cuando ya se conocen los servicios que se necesitan crear, pero el análisis de los servicios requeridos es un proceso complejo.

Como opinión personal, la clave para saber que servicios se pueden usar es identificar el costo de los servicios que esté dentro de las posibilidades de los clientes, y también conocer como funciona una tecnología e identificar si es la correcta para cubrir una necesidad en la aplicación que se va a implementar.

# Enlace al portafolio

* **<https://estiven-mejia-portfolio.pages.dev/projects/>**