



UNAH

UNIVERSIDAD NACIONAL
AUTÓNOMA DE HONDURAS

INGENIERÍA EN SISTEMAS

**Nombre Clase: IS912-
Sistemas Expertos**



UNAH

UNIVERSIDAD NACIONAL
AUTÓNOMA DE HONDURAS

IS912 – Sistemas Expertos II Periodo 2025

Proyecto II: Creación de un Pipeline de Datos & API con Caché Inteligente

Nº cuenta

20181003916

Nombre

David Alejandro García Ordoñez

Catedrático: Uayeb Caballero Rodriguez

18/07/25



Informe del Proyecto: Creación de un Pipeline de Datos y API con Caché Inteligente

Contenido

Enlaces:	3
Introducción.....	3
Fase 1: Infraestructura y Migración de Datos en Azure con Terraform	3
Fases 2 a 5: Desarrollo de API, Autenticación, Monitoreo y Caché Inteligente	8
Fase 6: Despliegue con Docker y Azure.....	16
Conclusión	18





UNAH

UNIVERSIDAD NACIONAL
AUTÓNOMA DE HONDURAS

INGENIERÍA EN SISTEMAS

**Nombre Clase: IS912-
Sistemas Expertos**

Enlaces:

- ❖ Portafolio: <https://davidgarcia-portafolio.pages.dev/>
- ❖ Github: <https://github.com/DaroGard/movies.api>
- ❖ Movies API: <https://api-pr2-env.azurewebsites.net/docs>

Introducción

Este proyecto aborda la construcción de un pipeline de datos y una API con integración a la nube, monitoreo, caché inteligente y despliegue automatizado.

A continuación, se detallan las fases y componentes principales de la solución implementada.

Fase 1: Infraestructura y Migración de Datos en Azure con Terraform

Se utilizó Terraform para automatizar la provisión de recursos en Azure.

- **main.tf:** Configura el provider de Azure y crea el grupo de recursos.
- **storage.tf:** Provisiona una cuenta de almacenamiento y un contenedor privado para almacenar archivos CSV.
- **db.tf:** Crea un servidor SQL Server y una base de datos para alojar los datos migrados.
- **df.tf:** Implementa Azure Data Factory para orquestar la migración ETL del dataset almacenado en Blob Storage hacia la base de datos SQL.
- **appinsight.tf:** Configura Application Insights para monitorear la API.
- **webapps.tf:** Despliega planes de servicio y aplicaciones web Linux con soporte para imágenes Docker.
- **acr.tf:** Implementa Azure Container Registry para almacenar imágenes Docker privadas.



UNAH

UNIVERSIDAD NACIONAL
AUTÓNOMA DE HONDURAS

INGENIERÍA EN SISTEMAS

Nombre Clase: IS912-
Sistemas Expertos

Evidencias:

Terraform plan

```
terraform plan -var-file="local.tfvars"

+ network_rules (known after apply)

+ queue_properties (known after apply)

+ routing (known after apply)

+ share_properties (known after apply)

+ static_website (known after apply)
}

# azurerm_storage_container.csv will be created
+ resource "azurerm_storage_container" "csv" {
+   container_access_type = "private"
+   default_encryption_scope = (known after apply)
+   encryption_scope_override_enabled = true
+   has_immutability_policy = (known after apply)
+   has_legal_hold = (known after apply)
+   id = (known after apply)
+   metadata = (known after apply)
+   name = "csv-content"
+   resource_manager_id = (known after apply)
+   storage_account_id = (known after apply)
}

Plan: 7 to add, 0 to change, 0 to destroy.
```

Terraform apply

```
terraform apply -var-file="local.tfvars"

azurerm_storage_account.storage: Creation complete after 1m9s [id=/subscriptions/3d7a4176-0edb-47e6-8b4d-756e5a78a211/resourceGroups/rg-pr2-env/providers/Microsoft.Storage/storageAccounts/storagepr2env]
azurerm_mssql_server.sqlserver: Still creating... [01m10s elapsed]
azurerm_storage_container.csv: Creating...
azurerm_storage_container.csv: Creation complete after 2s [id=/subscriptions/3d7a4176-0edb-47e6-8b4d-756e5a78a211/resourceGroups/rg-pr2-env/providers/Microsoft.Storage/storageAccounts/storagepr2env/blobServices/default/containers/csv-content]
azurerm_mssql_server.sqlserver: Still creating... [01m20s elapsed]
azurerm_mssql_server.sqlserver: Still creating... [01m30s elapsed]
azurerm_mssql_server.sqlserver: Still creating... [01m40s elapsed]
azurerm_mssql_server.sqlserver: Still creating... [01m50s elapsed]
azurerm_mssql_server.sqlserver: Creation complete after 1m55s [id=/subscriptions/3d7a4176-0edb-47e6-8b4d-756e5a78a211/resourceGroups/rg-pr2-env/providers/Microsoft.Sql/servers/sqlserver-pr2-env]
azurerm_mssql_database.moviesdb: Creating...
azurerm_mssql_database.moviesdb: Still creating... [00m10s elapsed]
azurerm_mssql_database.moviesdb: Still creating... [00m20s elapsed]
azurerm_mssql_database.moviesdb: Still creating... [00m30s elapsed]
azurerm_mssql_database.moviesdb: Still creating... [00m40s elapsed]
azurerm_mssql_database.moviesdb: Still creating... [00m50s elapsed]
azurerm_mssql_database.moviesdb: Still creating... [01m00s elapsed]
azurerm_mssql_database.moviesdb: Still creating... [01m10s elapsed]
azurerm_mssql_database.moviesdb: Still creating... [01m20s elapsed]
azurerm_mssql_database.moviesdb: Still creating... [01m30s elapsed]
azurerm_mssql_database.moviesdb: Still creating... [01m40s elapsed]
azurerm_mssql_database.moviesdb: Still creating... [01m50s elapsed]
azurerm_mssql_database.moviesdb: Still creating... [02m00s elapsed]
azurerm_mssql_database.moviesdb: Still creating... [02m10s elapsed]
azurerm_mssql_database.moviesdb: Creation complete after 2m11s [id=/subscriptions/3d7a4176-0edb-47e6-8b4d-756e5a78a211/resourceGroups/rg-pr2-env/providers/Microsoft.Sql/servers/sqlserver-pr2-env/databases/dbpr2]

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.
```

Azure Resource Group

rg-pr2-env

Subscription: Azure for Students

Subscription ID:

Tags: Created by: Terraform, David Garcia | date: 07-2025 | environment: dev

Name	Type	Location
adf-pr2-env	Data factory (V2)	Central US
appi-pr2-env	Application Insights	Central US
dbpr2 (sqlserver-pr2-env/dbpr2)	SQL database	Central US
sqlserver-pr2-env	SQL server	Central US
storagepr2env	Storage account	Central US



Almacenamiento

Container: csv-content

Search: [Search] + Add Directory Upload Change access level Refresh Delete Copy Paste Rename Acquire lease Break lease Edit columns

Overview

Diagnose and solve problems

Access Control (IAM)

Settings

Authentication method: Access key (Switch to Microsoft Entra user account)

Add filter

Search blobs by prefix (case-sensitive)

Only show active blobs

Showing all 1 items

Name	Last modified	Access tier	Blob type	Size	Lease state
movies.csv	18/7/2025, 10:37:23	Hot (inferred)	Block blob	2.73 MiB	Available

Base de datos

sqlserver-pr2-env.dat...

- Databases
 - System Databases
 - dbpr2
 - Tables
 - cinema.movies
 - cinema.users
 - Dropped Ledger Tables
 - Views
 - Synonyms
 - Programmability
 - Stored Procedures
 - cinema.users_insert
 - Functions
 - Database Triggers
 - Assemblies
 - Types
 - Sequences
 - External Resources
 - Storage
 - Security
 - Security



Linked Service

Linked services

Linked service defines the connection information to a data store or compute. [Learn more](#)

+ New

Filter by name Annotations: Any

Showing 1 - 2 of 2 items

Name	Type	Related	Annotations
Abs_LinkedService	Azure Blob Storage	0	
ASD_LinkedService	Azure SQL Database	0	

Datasets

Factory Resources

Filter resources by name

Pipelines

0

Change Data Capture (preview)

0

Datasets

1

ds_movies_abs

0

Data flows

0

Power Query

0

ds_movies_abs

DelimitedText

ds_movies_abs

Preview data

Linked service: Abs_LinkedService

Object: movies.csv

	movieid	title	genres
1	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
2	2	Jumanji (1995)	Adventure Children Fantasy
3	3	Grumpier Old Men (1995)	Comedy Romance
4	4	Waiting to Exhale (1995)	Comedy Drama Romance
5	5	Father of the Bride Part II (1995)	Comedy
6	6	Heat (1995)	Action Crime Thriller
7	7	Sabrina (1995)	Comedy Romance
8	8	Tom and Huck (1995)	Adventure Children
9	9	Sudden Death (1995)	Action
10	10	GoldenEye (1995)	Action Adventure Thriller

view data Detect format

Connection Schema Parameters

Linked service *

File path *

Compression type

Column delimiter

Row delimiter

Encoding

Quote character

Escape character

First row as header

Null value

Properties

General Related

Name *

ds_movies_abs

Description

Annotations

+ New

Azure SQL Database

ds_movies_ASD

Connection Schema Parameters

Linked service *

ASD_LinkedService

Test connection Edit + New Learn more

Table

cinema.movies

Refresh Preview data

Enter manually



Pipeline

Activities Validate Debug Add trigger

Search activities

Move and transform

- Copy data
- Data flow

Synapse

Azure Data Explorer

Azure Function

Batch Service

Databricks

Data Lake Analytics

General

HDInsight

Iteration & conditionals

Machine Learning

Power Query

Copy data

csv_abs_to_asd

Parameters Variables Settings **Output**

Pipeline run ID 9baee922-cb46-44f3-af31-3f8a13ba6f7c Pipeline status Succeeded [View debug run consumption](#)

All status

Showing 1 - 1 of 1 items

Activity name	Activity st...	Activit...	Run start	Duration	Integration runtime	Us
csv_abs_to_asd	Succeeded	Copy data	7/18/2025, 10:50:09 AM	27s	AutoResolveIntegrationRuntime (Central US)	

Datos

SERVERS

- Personal
 - sqlserver-gc-enu.database.windows.net...
 - Databases
 - System Databases
 - Adventureworks2017
 - Tables
 - Cinemamovies
 - Columns
 - Keys
 - Constraints
 - Triggers
 - Indexes
 - Statistics
 - Cinemasusers
 - Dropped Ledger Tables
 - Views
 - Synonyms
 - Programmability
 - Stored Procedures
 - cinemasusers_insert
 - Functions
 - Database Triggers
 - Assemblies
 - Types
 - Sequences
 - External Resources
 - Storage
 - Security
- AZURE

Database: dbp0

1 SELECT * FROM CINEMA_MOVIES;

Results Messages

movieId	title	genres
1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
2	Jumanji (1995)	Adventure Children Fantasy
3	Grumpier Old Men (1995)	Comedy Romance
4	Waiting to Exhale (1995)	Comedy Drama Romance
5	Father of the Bride Part II (1995)	Comedy
6	Heat (1995)	Action Crime Thriller
7	Sabrina (1995)	Comedy Romance
8	Ten and Hank (1995)	Adventure Children
9	Sudden Death (1995)	Action
10	GoldEye (1995)	Action Adventure Thriller
11	American President, The (1995)	Comedy Drama Romance
12	Dracula: Dead and Loving It (1995)	Comedy Horror
13	Batle (1995)	Adventure Animation Children
14	Nixon (1995)	Drama
15	Cutthroat Island (1995)	Action Adventure Romance
16	Casino (1995)	Crime Drama
17	Sense and Sensibility (1995)	Drama Romance
18	Four Rooms (1995)	Comedy
19	Ace Ventura: When Nature Calls (1995)	Comedy
20	Honey Train (1995)	Action Comedy Crime Drama Thriller
21	Get Shorty (1995)	Comedy Crime Thriller
22	Copcat (1995)	Crime Drama Horror Mystery Thriller
23	Assassins (1995)	Action Crime Thriller



Fases 2 a 5: Desarrollo de API, Autenticación, Monitoreo y Caché Inteligente

Autenticación con Firebase (controllers/firebase.py):

- ✓ Registro y login de usuarios gestionados con Firebase Authentication.
- ✓ Usuarios almacenados también en la base de datos SQL con campos `is_admin` e `is_active` para permisos y estado.
- ✓ Emisión de tokens JWT personalizados para autorización interna.

Catálogo de películas (controllers/moviescatalog.py):

- ✓ Endpoint GET `/catalog` con soporte para filtros dinámicos (`category`), usando caché Redis para acelerar respuestas.
- ✓ Endpoint POST `/catalog` para agregar películas, protegido con validación de administrador, que invalida caché relacionada automáticamente tras inserción.

Modelos (carpeta models)

- ✓ Definición de modelos con validaciones estrictas para usuarios y películas, incluyendo validaciones de formato y complejidad de contraseñas.

Utilidades (carpeta utils)

- **database.py:** Manejo asíncrono de conexión a SQL Server con `pyodbc`, ejecución de queries con manejo de errores y conversión a JSON.
- **redis_cache.py:** Gestión robusta de conexión a Redis, obtención, almacenamiento y eliminación de caché con manejo de errores y logging.
- **security.py:** Creación y validación de tokens JWT, decoradores para autorización de usuarios normales y administradores.
- **telemetry.py:** Configuración y habilitación de Application Insights para telemetría completa de la API usando OpenTelemetry.

Esquema de base de datos (DDL.sql)

Definición de esquema y tablas para películas y usuarios, con procedimiento almacenado para inserción segura de usuarios.

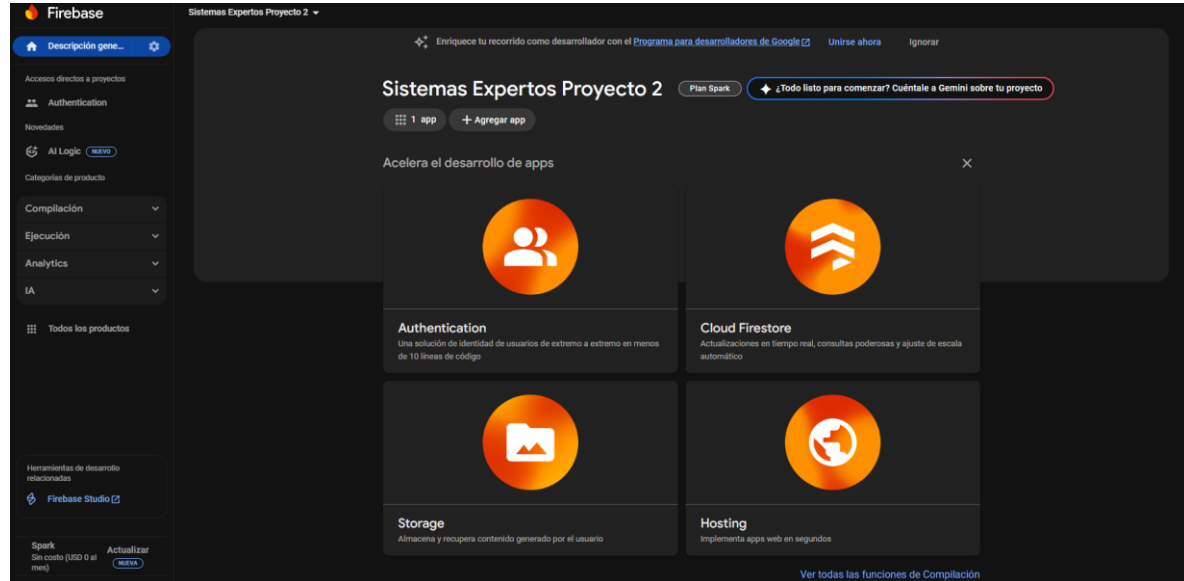
API principal (main.py)

Inicialización de FastAPI con integración a telemetry y definición de endpoints.
Uso de dependencias para seguridad y autorización con JWT.

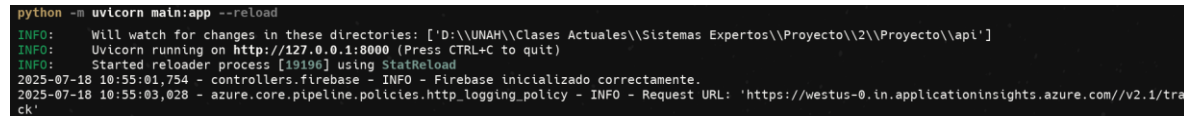


Evidencias:

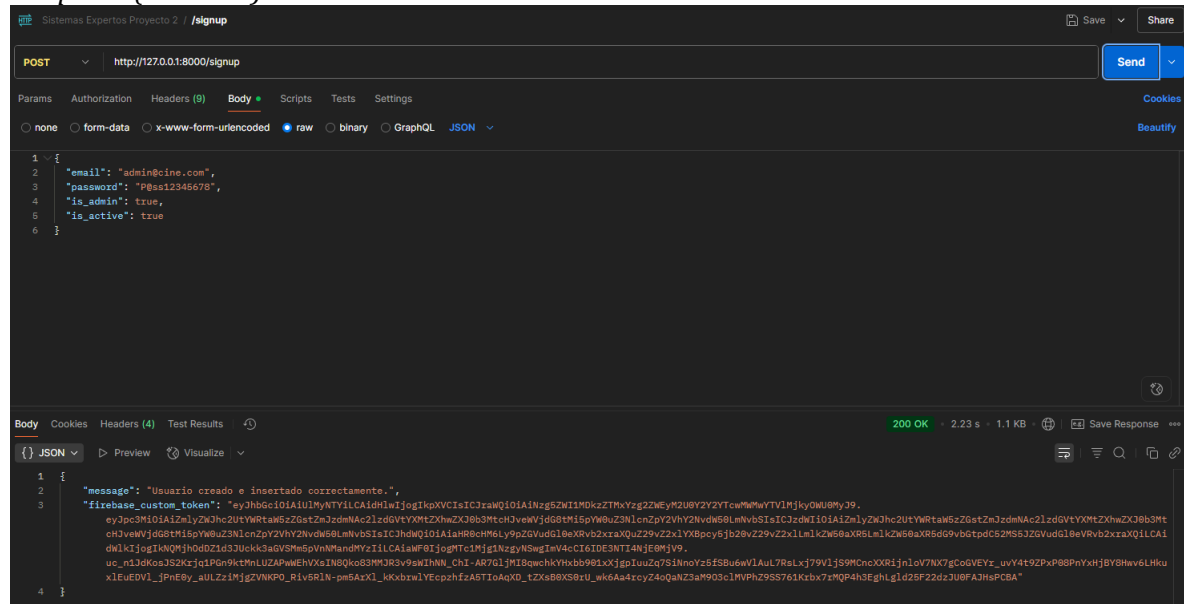
Firebase

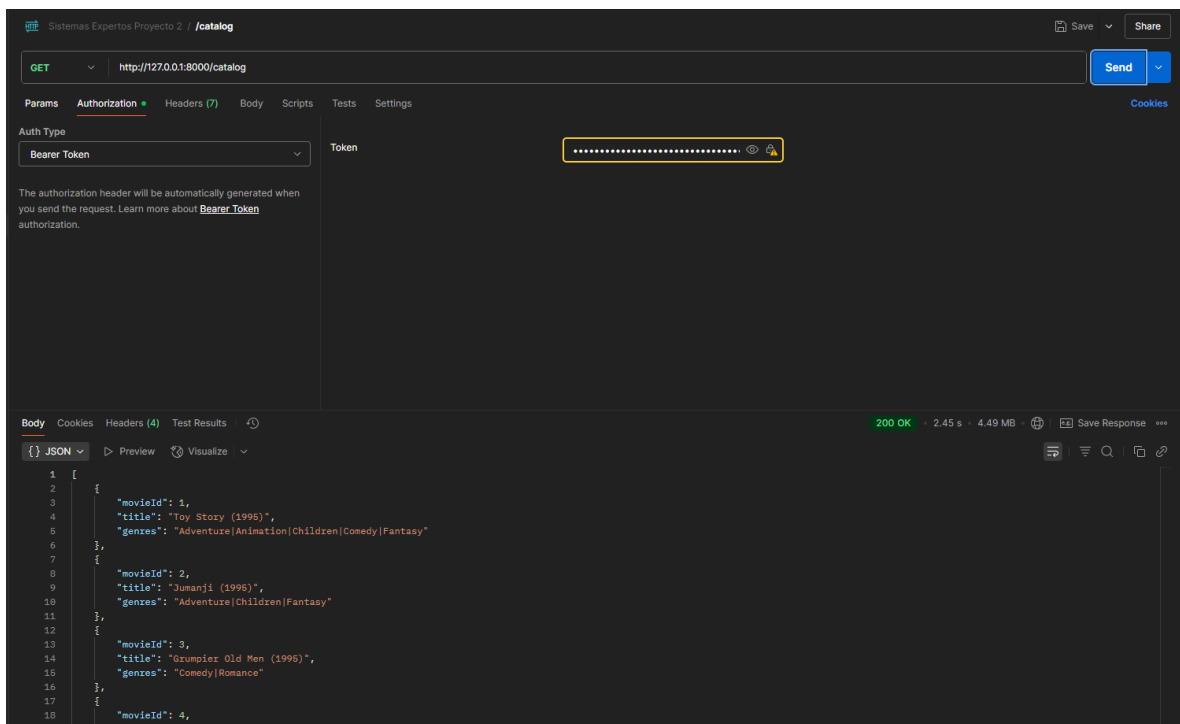
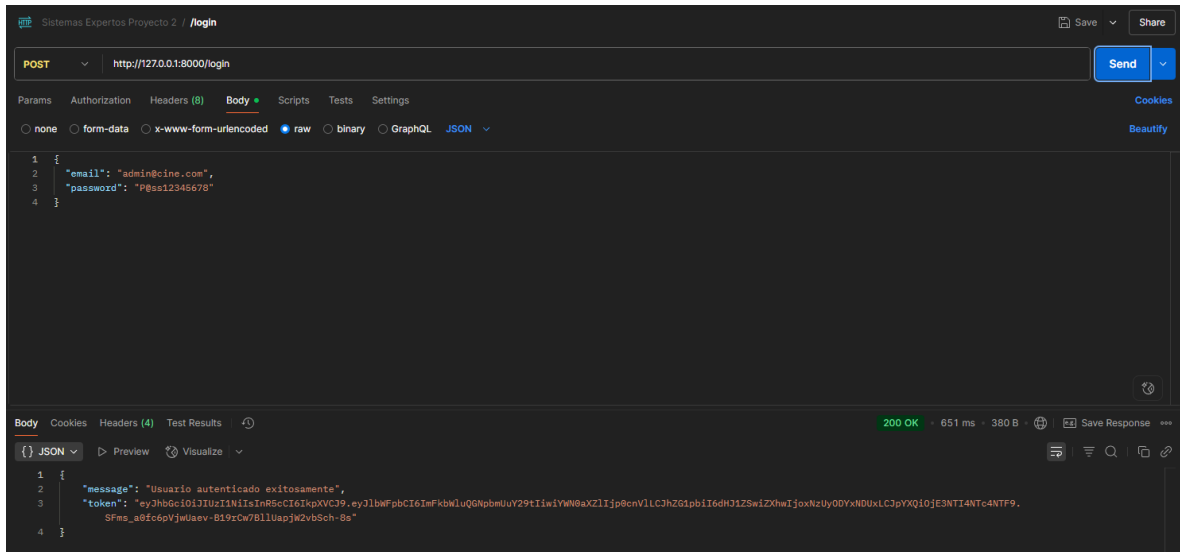


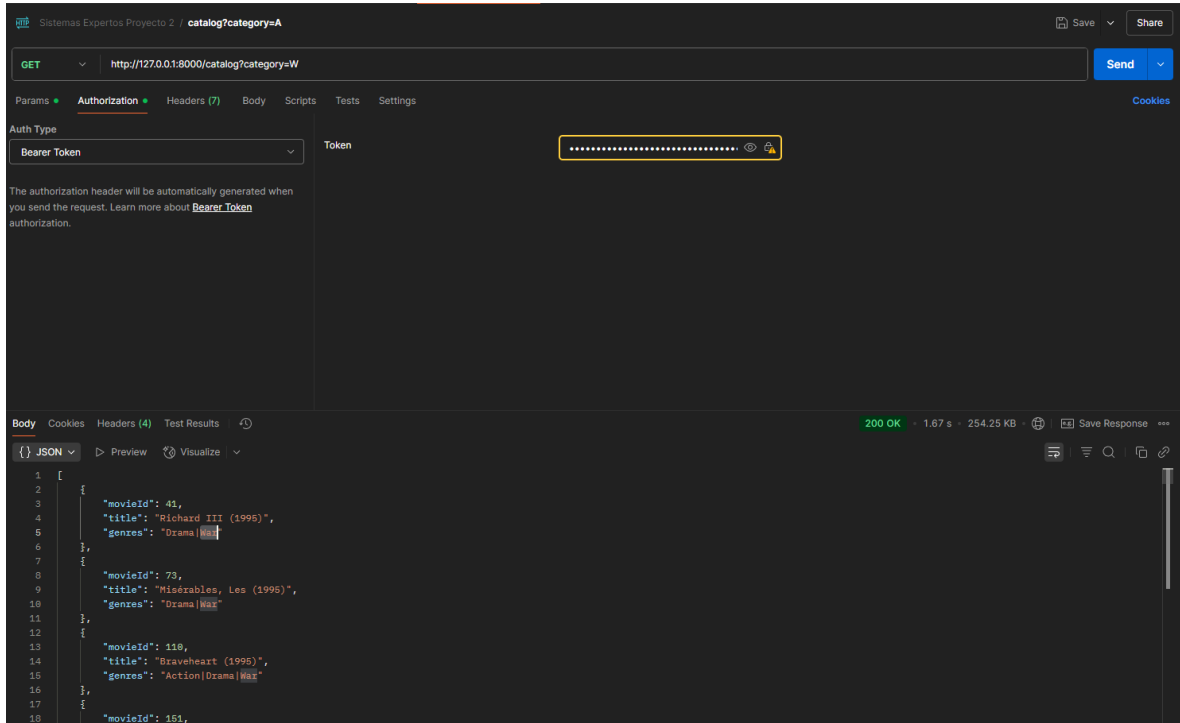
Uvicorn



Endpoints(Postman)

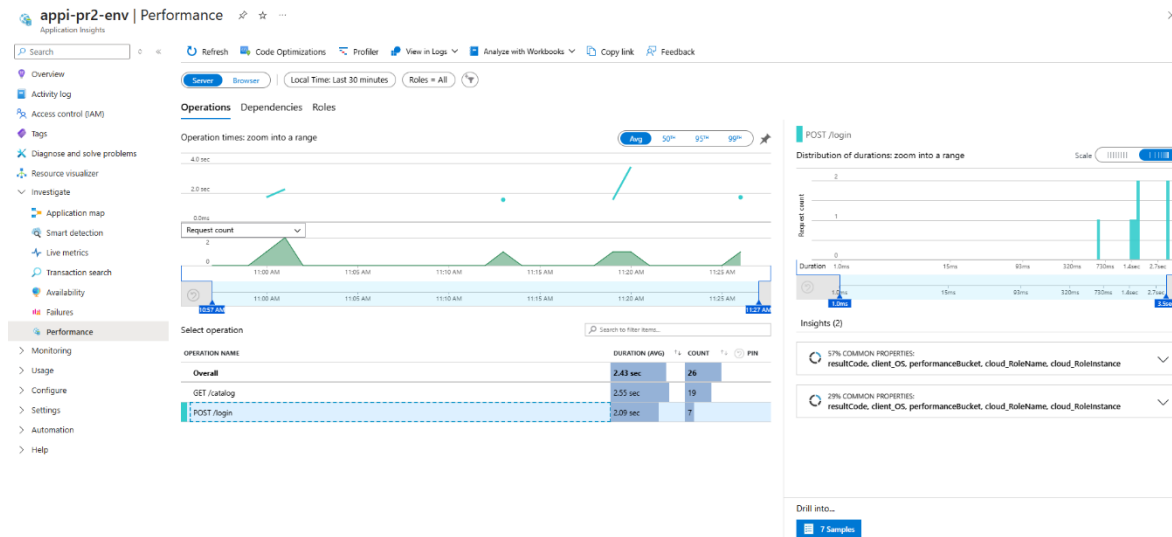
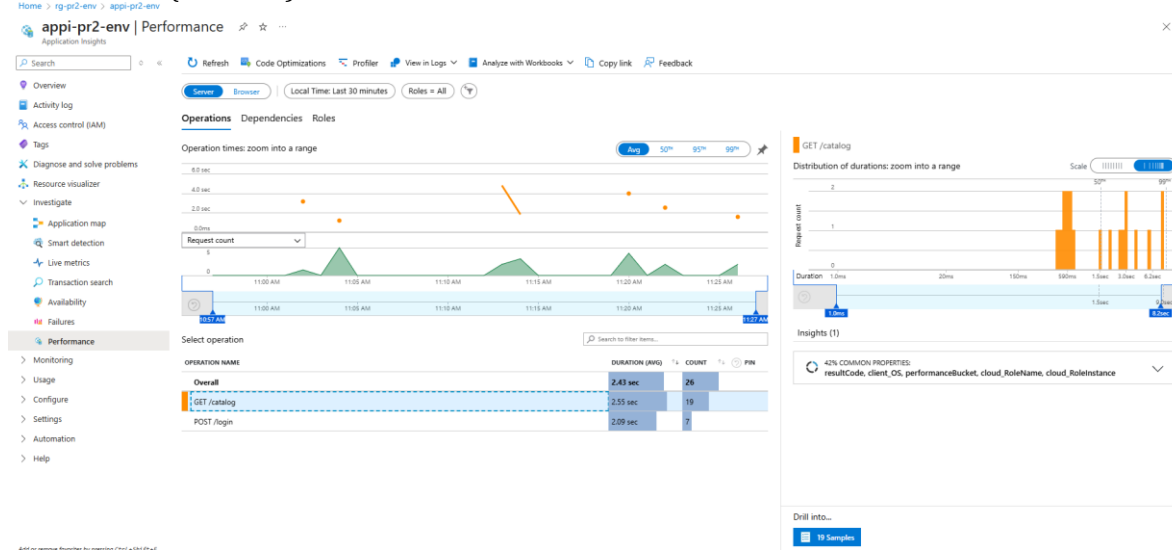








Rendimiento (sin Redis)



Redis

```
2025-07-18 13:07:35,990 - utils.redis_cache - INFO - ☒ Conectado exitosamente a Redis usando Connection String
2025-07-18 13:07:36,077 - controllers.moviescatalog - INFO - Consultando todo el catálogo de películas.
2025-07-18 13:07:36,077 - utils.database - INFO - Intentando conectar a la base de datos...
2025-07-18 13:07:36,658 - azure.core.pipeline.policies.http_logging_policy - INFO - Request URL: 'https://centralus-2.in.applicationinsights.azure.com/track'
```

```
2025-07-18 13:07:41,646 - utils.redis_cache - INFO - ☒ Datos almacenados en caché con clave 'movies:catalog:all' por 1800 segundos
2025-07-18 13:07:41,650 - controllers.moviescatalog - INFO - Películas almacenadas en caché con clave: movies:catalog:all
2025-07-18 13:07:41,681 - azure.core.pipeline.policies.http_logging_policy - INFO - Request URL: 'https://centralus-2.in.applicationinsights.azure.com/v2.1/track'
```



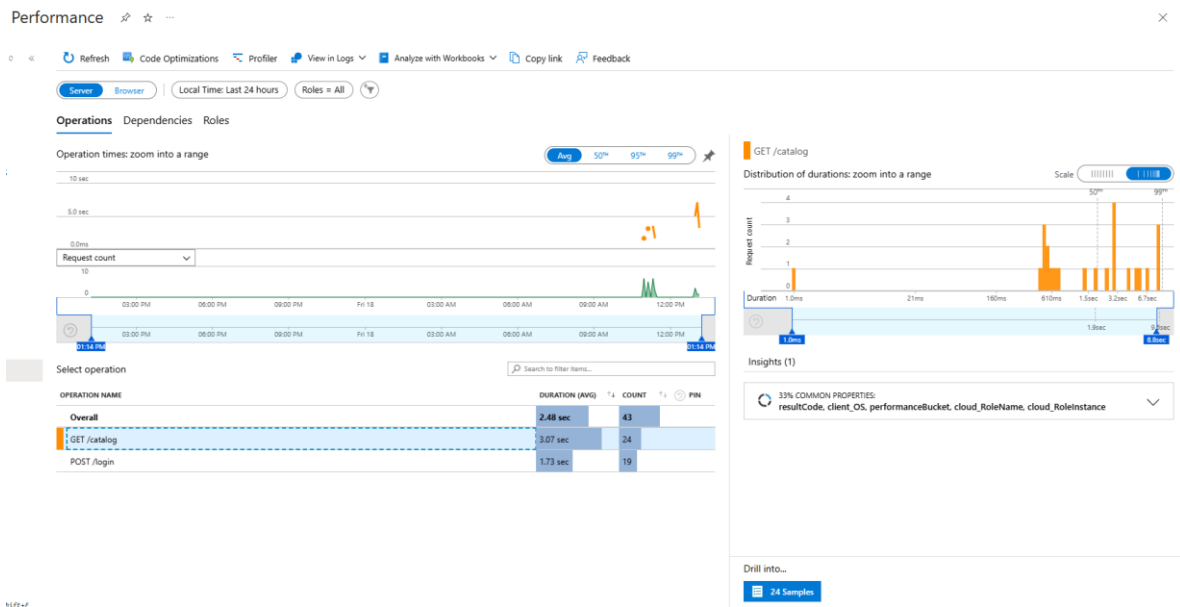
Endpoint con redis

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

- Method:** GET
- URL:** http://127.0.0.1:8000/catalog
- Auth Type:** Bearer Token
- Token:** [Redacted]
- Response:** 200 OK, 7.08 s, 4.49 MB
- Body (JSON):**

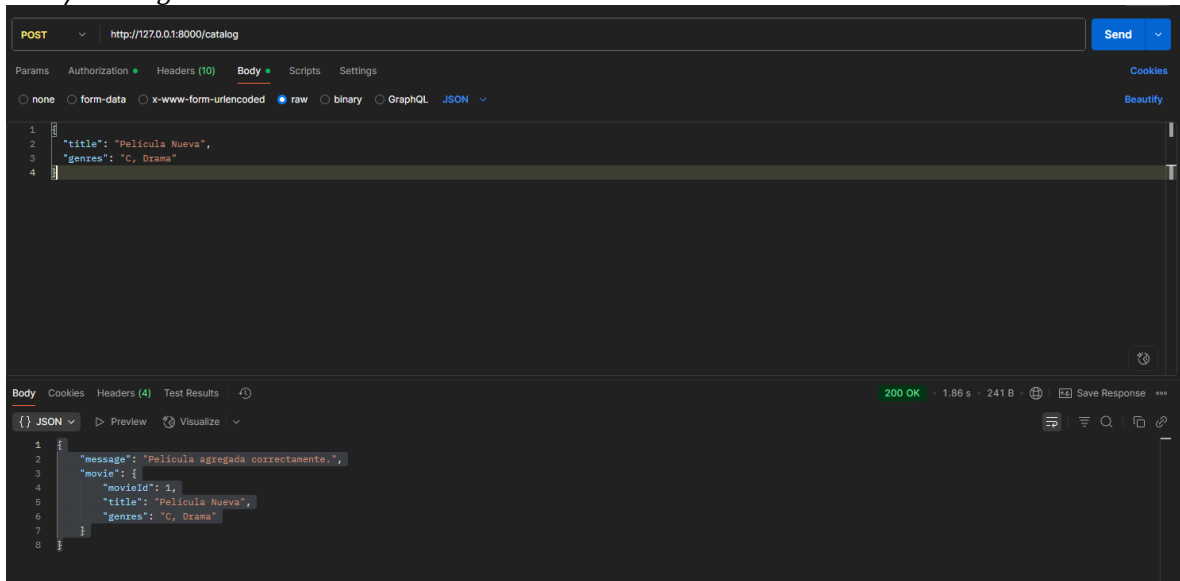
```
[{"movieid": 1, "title": "Toy Story (1995)", "genres": "Adventure|Animation|Children|Comedy|Fantasy"}, {"movieid": 2, "title": "Jumanji (1995)", "genres": "Adventure|Children|Fantasy"}, {"movieid": 3, "title": "Grumpier Old Men (1995)", "genres": "Comedy|Romance"}, {"movieid": 4, ...}]
```

Rendimiento con redis



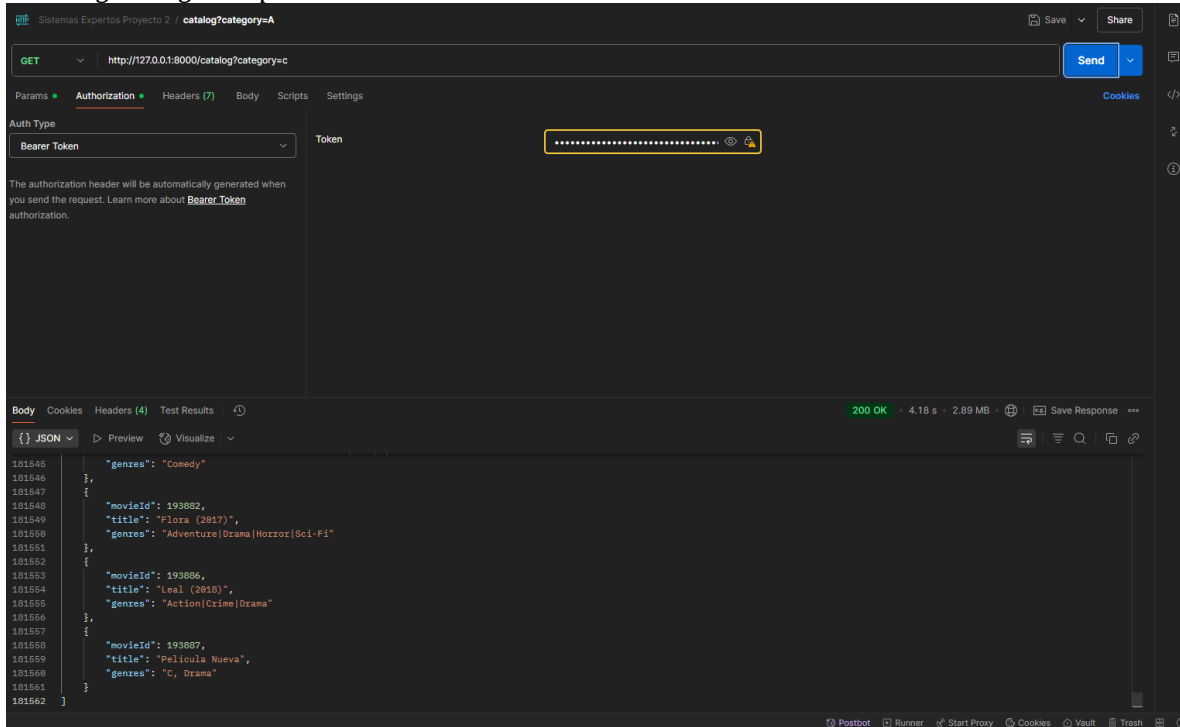


Post /Catalog con redis



```
2025-07-18 13:31:10,574 - utils.redis_cache - INFO - Datos almacenados en caché con clave 'movies:catalog:c' por 1800 segundos
INFO: 127.0.0.1:53364 - "GET /catalog?category=C HTTP/1.1" 200 OK
```

Primer get luego del post:





Segundo get luego del post

GET `http://127.0.0.1:8000/catalog?category=c` **Send**

Params • **Authorization** • Headers (7) • Body • Scripts • Settings

Auth Type: **Bearer Token** Token: `*****`

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

Body Cookies Headers (4) Test Results **200 OK** 1.54 s 2.89 MB Save Response

JSON Preview Visualize

```
181545     "genres": "Comedy"
181546   },
181547   {
181548     "movieId": 193882,
181549     "title": "Flora (2017)",
181550     "genres": "Adventure|Drama|Horror|Sci-Fi"
181551   },
181552   {
181553     "movieId": 193886,
181554     "title": "Isleil (2016)",
181555     "genres": "Action|Crime|Drama"
181556   },
181557   {
181558     "movieId": 193887,
181559     "title": "Película Nueva",
181560     "genres": "C, Drama"
181561   }
181562 ]
```



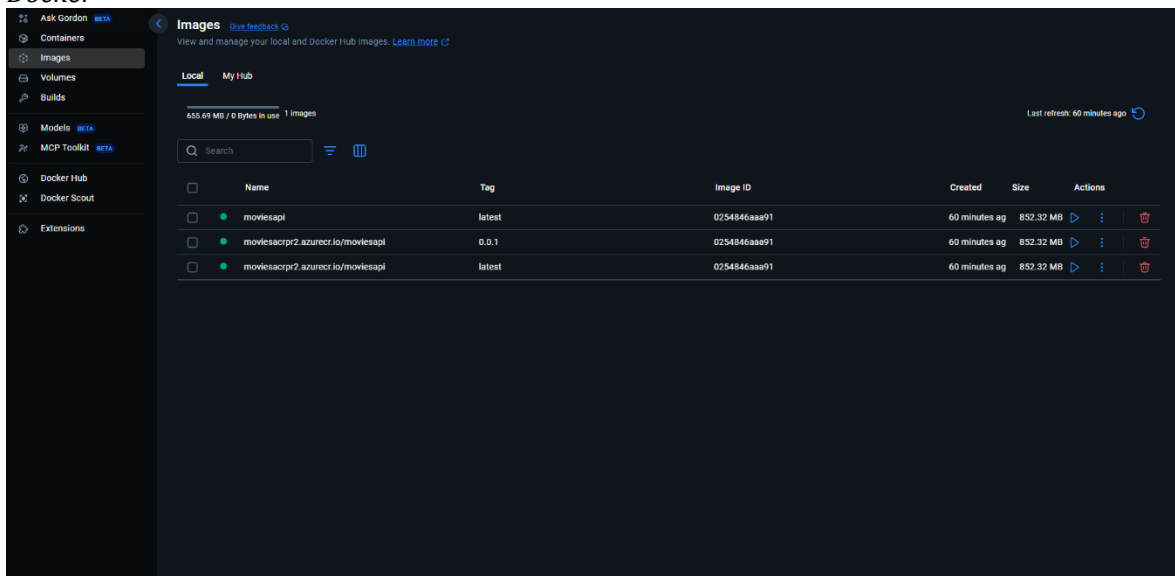
Fase 6: Despliegue con Docker y Azure

Se creó una imagen Docker para la API, almacenada en Azure Container Registry (ACR).

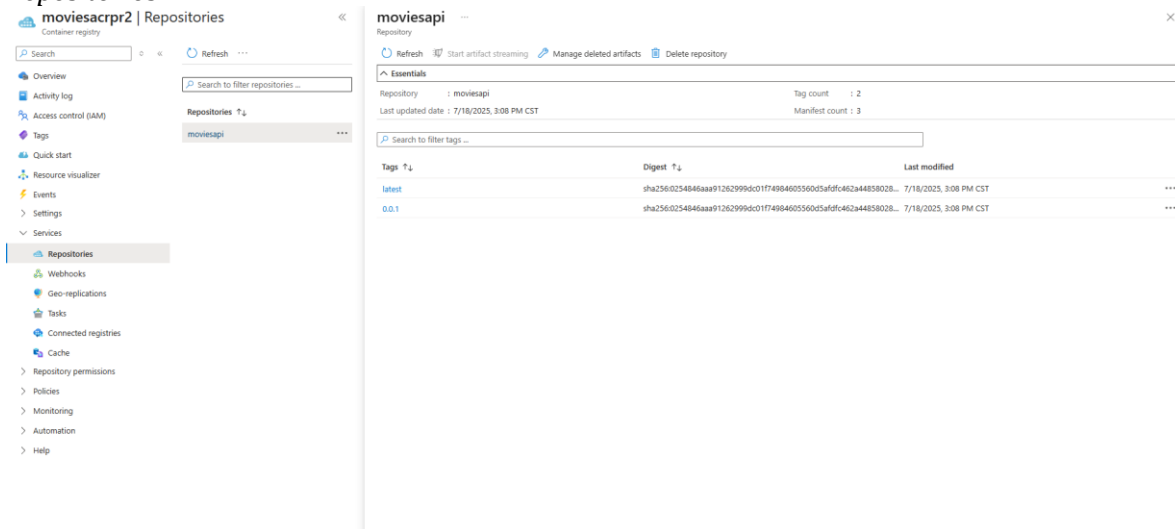
La API y UI se desplegaron en Azure Web Apps Linux con planes de servicio configurados, integrando las imágenes Docker desde ACR.

Evidencias

Docker



Repositorios





UNAH

UNIVERSIDAD NACIONAL
AUTÓNOMA DE HONDURAS

INGENIERÍA EN SISTEMAS

**Nombre Clase: IS912-
Sistemas Expertos**

Lanzamiento

Movies API 1.0.0 OAS 3.1
[/openapi.json](#)

default ^

- POST** /signup Signup
- POST** /login Login
- GET** /catalog Catalog
- POST** /catalog Create Movie
- GET** /health Health Check
- GET** / Root

Schemas ^

- HTTPValidationError** > Expand all object
- MovieCatalog** > Expand all object
- UserLogin** > Expand all object
- UserRegister** > Expand all object
- ValidationError** > Expand all object



UNAH

UNIVERSIDAD NACIONAL
AUTÓNOMA DE HONDURAS

INGENIERÍA EN SISTEMAS

**Nombre Clase: IS912-
Sistemas Expertos**

Conclusión

Este proyecto fue una experiencia clave para integrar y aplicar conocimientos sobre desarrollo backend y servicios en la nube, a través de herramientas como Azure Data Factory, FastAPI, Firebase, Redis, Terraform y Docker, se construyó una solución completa; desde la migración de datos hasta el despliegue en un entorno productivo.

Más allá de lo técnico, el trabajo permitió comprender cómo cada tecnología contribuye al rendimiento, seguridad y escalabilidad del sistema, como también, reforzó la importancia de la automatización y el monitoreo en tiempo real para mantener servicios confiables.

En definitiva, este proyecto no solo fortaleció mis habilidades prácticas, sino que me ofreció una visión clara de lo que implica diseñar y operar soluciones modernas en la vida profesional.