**MEMORIA DEL PROYECTO  
INFRAESTRUCTURA DE NUBE – CLOUDEDU SERVICES**

**Introducción**

En este proyecto se ha desarrollado una arquitectura cloud‑native para la aplicación CloudEdu Services, aplicando conceptos fundamentales de infraestructura en la nube como contenedorización, orquestación con Kubernetes, automatización, escalabilidad y monitorización. El objetivo principal ha sido desplegar una aplicación funcional y preparada para entornos reales, capaz de adaptarse dinámicamente a la carga del sistema mediante el uso de métricas reales.

El proyecto presentado supone una evolución del trabajo inicial, incorporando como mejora principal el autoescalado horizontal basado en métricas reales y un sistema de monitorización que permite observar el comportamiento del sistema en tiempo real, acercándose a un escenario de producción realista.

**Documentación técnica**

**Arquitectura y componentes**

La arquitectura del sistema se basa en una aplicación web desarrollada en Python y empaquetada en una imagen Docker personalizada. Esta imagen se despliega sobre un clúster Kubernetes, que actúa como plataforma de organizaciony gestión de los contenedores.

La aplicación se ejecuta mediante un Deployment que mantiene varias réplicas activas para garantizar disponibilidad y tolerancia a fallos. El acceso se realiza a través de un Service que distribuye el tráfico entre los distintos pods de forma transparente. Para asegurar la persistencia de la información, se utiliza un volumen persistente asociado a los pods, evitando la pérdida de datos ante reinicios o recreaciones.

La escalabilidad del sistema se gestiona mediante un Horizontal Pod Autoscaler que ajusta automáticamente el número de réplicas en función del consumo real de CPU y memoria. Para ello, el clúster dispone de un sistema de monitorización que proporciona métricas reales, permitiendo que el escalado se realice de forma automática y fiable.

**Flujo de funcionamiento**

El funcionamiento del sistema comienza con el acceso del usuario a la aplicación a través del servicio expuesto por Kubernetes. Dicho servicio redirige las peticiones hacia los pods disponibles, balanceando la carga entre ellos.

Cada pod ejecuta una instancia de la aplicación y accede al almacenamiento persistente configurado. Kubernetes supervisa continuamente el estado de los contenedores, mientras que el sistema de monitorización recopila métricas de uso de recursos que son utilizadas por el autoescalado para aumentar o reducir el número de réplicas según la carga existente.

1. - Usuario accede a la aplicación
2. -LoadBalancer distribuye tráfico a los pods
3. -Pod procesa request(Flask)
4. - Datos se guardan en almacenamiento persistente
5. -Prometheus recopila métricas
6. HPA decide si escalar (CPU mayor a 70% y memoria mayor a 80%)
7. Si es necesario, se aumentan los pods o se disminuyen
8. Carga distribuida entre los pods

**Despliegue y configuración**

El despliegue del proyecto se realiza preparando previamente el entorno con Docker y Kubernetes. A continuación, se construye la imagen Docker de la aplicación y se aplican los manifiestos de Kubernetes que definen la configuración del sistema, el despliegue de la aplicación, la exposición del servicio y el almacenamiento persistente.

Posteriormente, se instala el sistema de monitorización y el componente encargado de proporcionar métricas reales del clúster, lo que permite habilitar el funcionamiento correcto del Horizontal Pod Autoscaler. Una vez completados estos pasos, el sistema queda operativo y preparado para escalar de forma automática.

**Seguridad y control de acceso**

La seguridad del sistema se ha abordado mediante la aplicación de políticas de control de acceso acordes a la arquitectura. Se han definido cuentas de servicio específicas para los pods y se han asignado permisos limitados únicamente a los recursos necesarios para su funcionamiento.

Asimismo, se han aplicado configuraciones de seguridad a nivel de contenedor, evitando la ejecución como usuario root y siguiendo el principio de mínimo privilegio, lo que contribuye a un entorno más seguro y controlado.

**Problemas encontrados y soluciones**

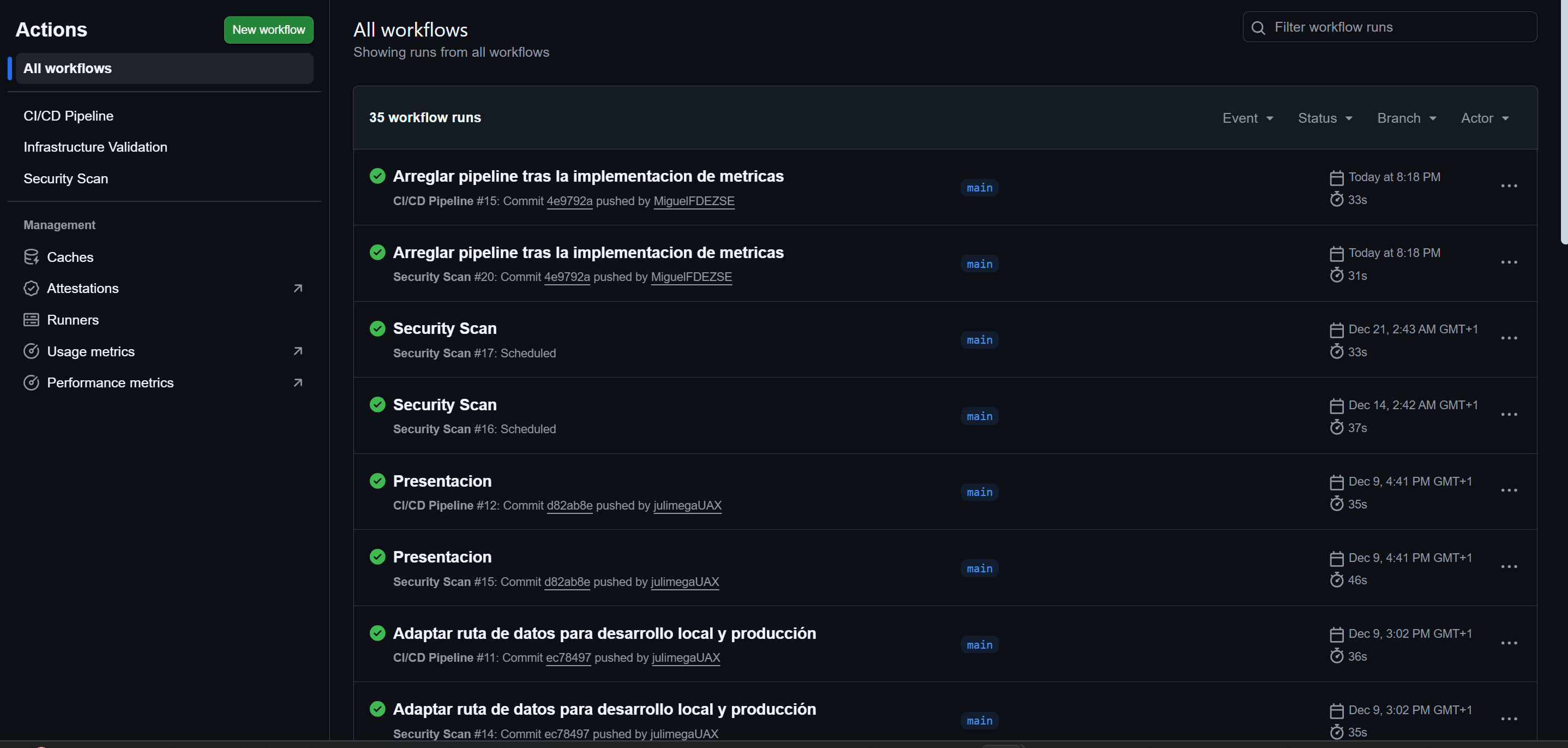
Durante el desarrollo del proyecto se presentaron diversas dificultades técnicas relacionadas con la configuración del clúster, la persistencia de los datos y la exposición del servicio. Estos problemas fueron resueltos ajustando la configuración de Kubernetes y validando el correcto funcionamiento de cada componente.

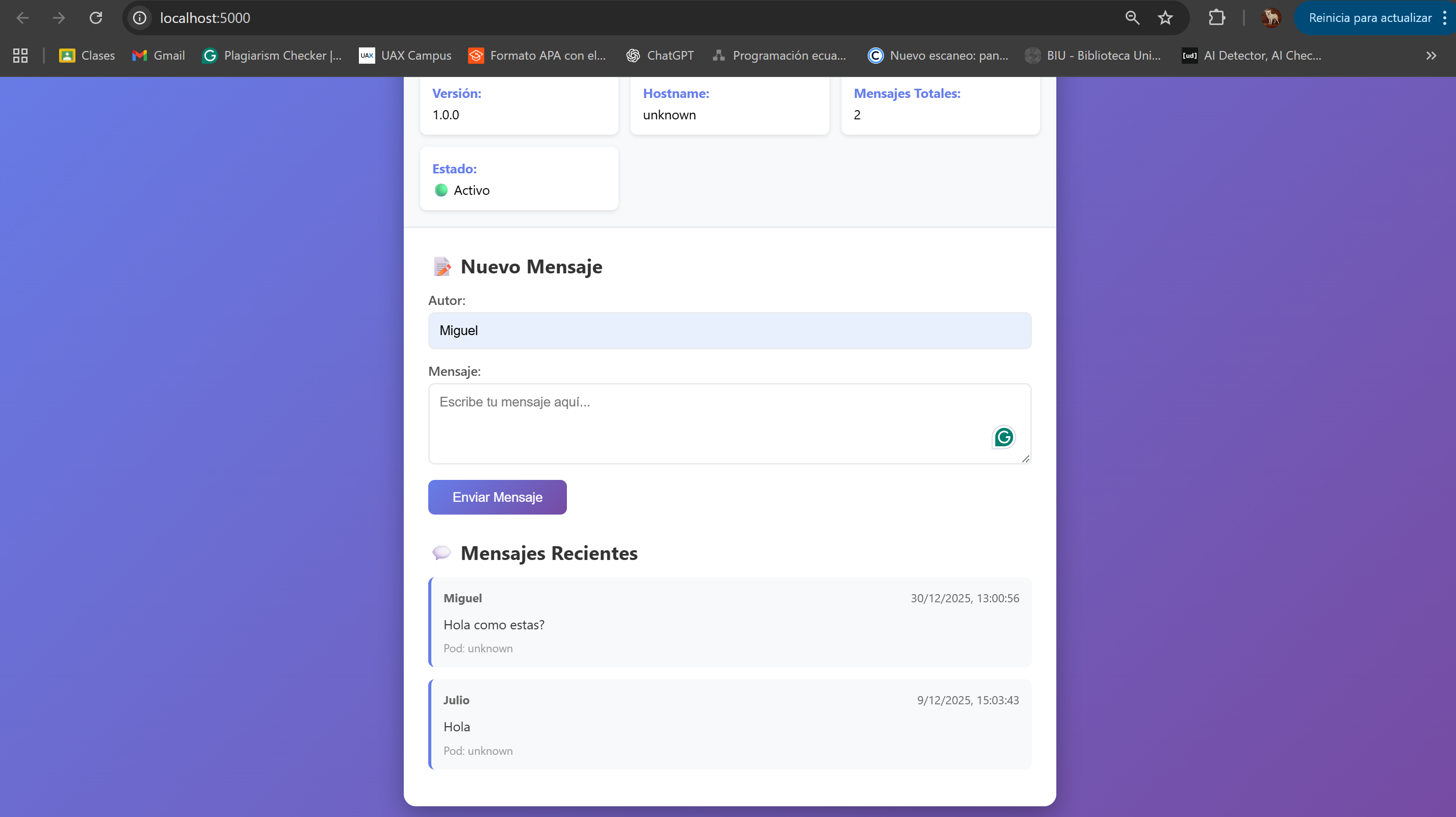
Uno de los aspectos más relevantes fue la correcta activación del autoescalado. Inicialmente, el escalado no funcionaba debido a la ausencia de métricas reales, problema que se resolvió mediante la instalación del sistema de monitorización adecuado, permitiendo a Kubernetes obtener información real del uso de recursos.

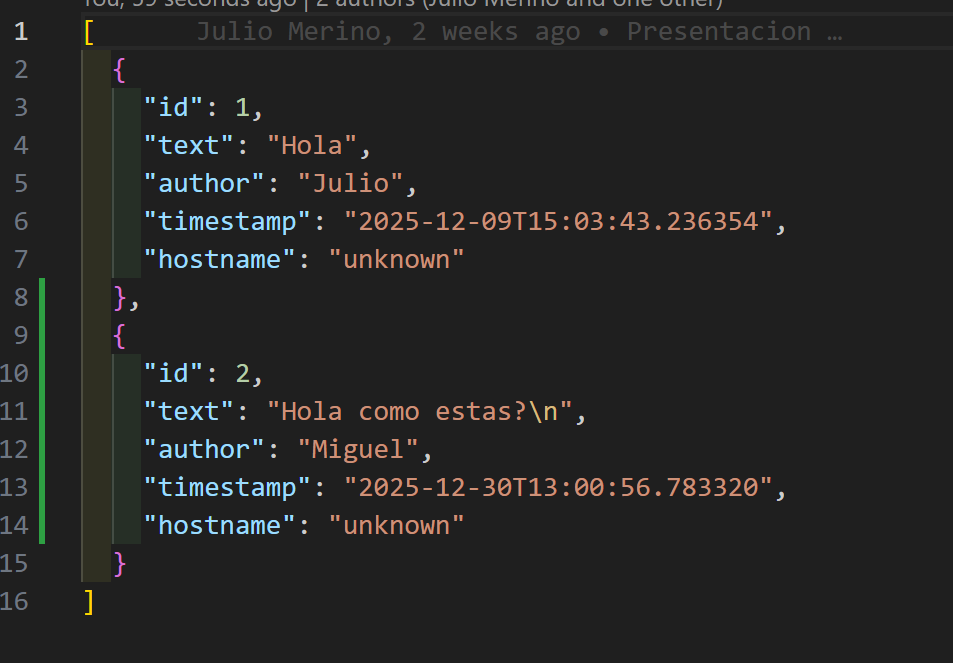
**Evidencia del funcionamiento**

El correcto funcionamiento del sistema se ha verificado observando el estado de los pods y servicios desplegados, así como comprobando el comportamiento del autoescalado bajo diferentes niveles de carga. Las métricas recogidas permiten comprobar cómo el número de réplicas se ajusta automáticamente en función del consumo de recursos.

Asimismo, se ha validado el acceso a la aplicación desde el navegador y la persistencia de los datos tras reinicios de los pods, confirmando que la arquitectura cumple con los requisitos establecidos.







**Reflexión final**

A lo largo del desarrollo del proyecto se han tomado decisiones técnicas orientadas a construir una arquitectura robusta, escalable y cercana a un entorno real de producción. La incorporación del autoescalado basado en métricas reales y de un sistema de monitorización ha supuesto una mejora significativa respecto a la versión inicial.

Como aprendizaje final, el proyecto ha permitido consolidar conocimientos sobre infraestructura en la nube, Kubernetes, escalabilidad automática y monitorización, así como comprender la importancia de una documentación clara y una arquitectura bien diseñada en proyectos de este tipo.