UNIVERSIDAD INTERNACIONAL DEL ECUADOR

MACHALA

-o-

PROGRAMACION ORIENTADA A OBJETOS

Evaluación en Contacto con el Docente

DOCENTE:

ING. MILTON RICARDO PALACIOS

CURSO: 1-ECC-3A

ELABORADO POR:

ANDRÉS MONTESDEOCA CRUZ

Machala, 29 junio 2025

Informe Final del Proyecto

Sistema de Gestión de Streaming en Go

1. Introducción

El presente proyecto consiste en el desarrollo de un sistema de gestión de streaming, que permite registrar usuarios, gestionar contenidos audiovisuales (películas y series), reproducirlos y calificarlos. Implementado en Go, el sistema aplica programación orientada a objetos, manejo de errores, interfaces y concurrencia para atender múltiples solicitudes. Este proyecto integra los conocimientos teóricos y prácticos adquiridos durante el curso, representando un producto funcional que demuestra habilidades en diseño y desarrollo de software.

2. Justificación

El streaming de contenidos digitales es una tendencia consolidada en la industria tecnológica, debido a la creciente demanda de acceso inmediato a medios audiovisuales. Elegir Go para esta implementación es estratégico, ya que ofrece eficiencia en la concurrencia y facilidad para construir servicios web escalables. Este proyecto permite aplicar conceptos fundamentales de programación funcional y orientada a objetos, facilitando la mantenibilidad, ampliación y escalabilidad del sistema.

3. Integración de las Cuatro Unidades del Curso

El desarrollo de este proyecto refleja la integración de los contenidos y habilidades adquiridas a lo largo de las cuatro unidades del curso:

- **Unidad 1: Programación Orientada a Objetos** — Definición de estructuras y métodos, encapsulamiento y modularidad en la gestión de usuarios, contenidos y calificaciones.

- **Unidad 2: Estructuras de Datos e Interfaces** — Uso de slices y punteros para manejar colecciones, implementación de interfaces para abstraer comportamientos y promover el desacoplamiento.

- **Unidad 3: Manejo de Errores y Concurrencia** — Validaciones robustas para garantizar integridad de datos y uso de la concurrencia nativa de Go para atender múltiples solicitudes HTTP simultáneamente.

- **Unidad 4: Servicios Web REST y JSON** — Diseño e implementación de ocho servicios web RESTful que permiten interacción remota con el sistema a través de JSON, facilitando la interoperabilidad y escalabilidad.

Esta integración asegura que el proyecto no solo cumple con los requisitos académicos, sino que representa un sistema realista y funcional para la gestión de streaming.

4. Análisis y Diseño

4.1 Objetivos del Sistema

- Registrar y gestionar usuarios con validación de datos.

- Administrar un catálogo de contenidos (películas y series).

- Proveer funcionalidad para la reproducción simulada de contenidos.

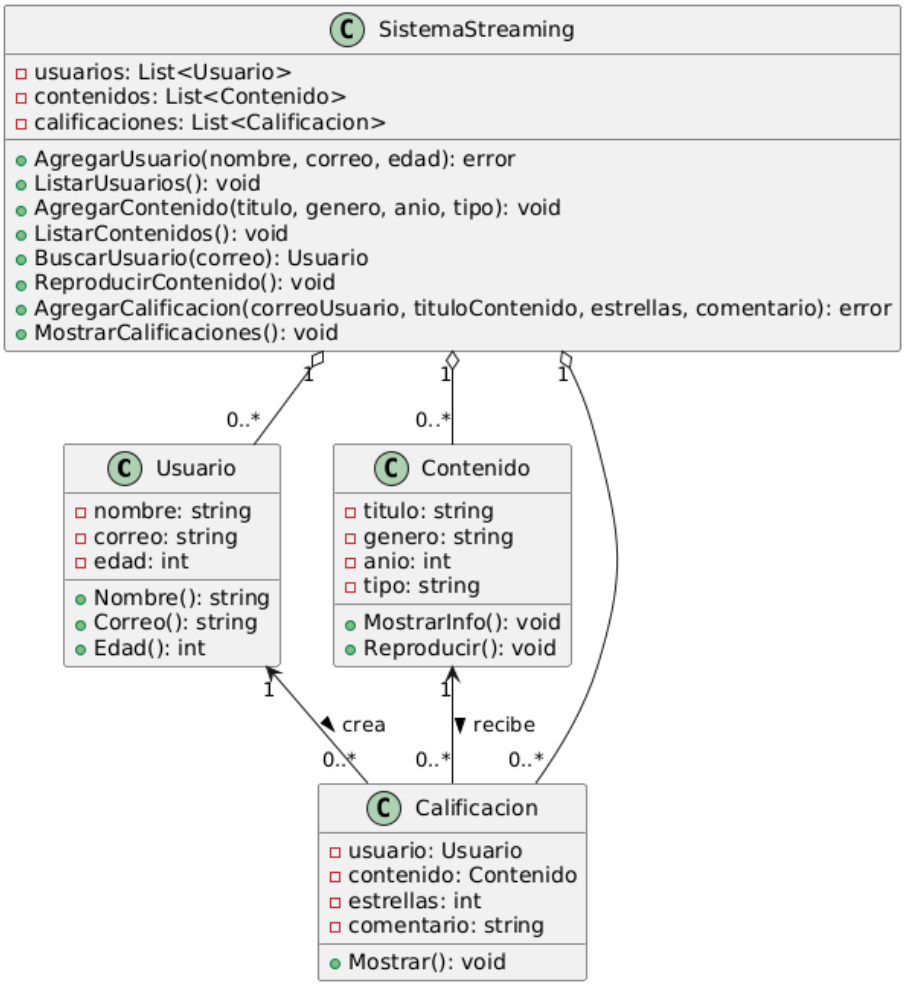
- Permitir calificar contenidos con validación y comentarios.

- Ofrecer una interfaz básica por consola y servicios web REST para acceso remoto.

4.2 Módulos del Sistema

Módulo	Descripción
Gestión de Usuarios	Registro, validación y listado de usuarios
Gestión de Contenidos	Creación, almacenamiento y consulta del catálogo de videos
Reproducción	Simulación de reproducción de contenido
Calificaciones	Agregar y mostrar calificaciones con validación y comentarios
Servicios Web REST	Endpoints para acceso y manipulación de usuarios, contenidos y calificaciones

4.3 Diagrama de Clases



5. Implementación

5.1 Lenguaje y Herramientas

- Lenguaje: Go (Golang)
- Control de versiones: Git y GitHub
- Paquetes: net/http, github.com/gorilla/mux
- IDE: Visual Studio Code
- Módulos de Go para gestión de dependencias

5.2 Servicios Web Implementados

1. GET /videos — Listar videos disponibles.
2. GET /stream?id= — Transmitir video para reproducción.
3. POST /usuarios — Registrar nuevo usuario.
4. GET /usuarios — Listar usuarios registrados.
5. POST /calificaciones — Agregar calificación a contenido.
6. GET /calificaciones — Listar calificaciones.
7. POST /contenidos — Agregar nuevo contenido.
8. GET /contenidos — Listar contenidos.

5.3 Concurrencia

El servidor HTTP aprovecha las goroutines para manejar múltiples solicitudes simultáneas, asegurando respuesta eficiente y sin bloqueos.

6. Pruebas y Resultados

6.1 Pruebas Unitarias

Se implementaron pruebas para validar:

- Prevención de usuarios duplicados.
- Validación correcta de rango de estrellas en calificaciones.
- Búsqueda y listado de usuarios y contenidos.

6.2 Pruebas de Integración

Se verificaron interacciones entre módulos, como:

- Registro de usuarios seguido de calificación de contenido.
- Reproducción simulada de videos desde el catálogo.

6.3 Pruebas Funcionales

- Uso del menú por consola para probar funcionalidades.
- Uso de Postman para probar endpoints REST, validando entradas y salidas JSON.

Los resultados confirmaron la correcta integración y funcionamiento del sistema.

7. Proyección Futura y Aplicaciones

El sistema de gestión de streaming desarrollado en este proyecto representa una base funcional para aplicaciones reales en el mundo digital. A medida que la tecnología avanza, surgen nuevas posibilidades de expansión y sofisticación de este tipo de sistemas.

7.1 Visualización del futuro:

En los próximos años, los sistemas de streaming evolucionarán hacia plataformas altamente personalizadas, impulsadas por inteligencia artificial, aprendizaje automático y análisis de comportamiento del usuario. Se espera que las futuras versiones de este sistema incluyan:

- **Recomendaciones automatizadas** según el historial de visualización del usuario.
- **Interfaces gráficas modernas** (aplicaciones móviles o web con frontend en React, Angular o Flutter).
- **Servicios distribuidos** mediante arquitecturas de microservicios usando contenedores como Docker y orquestadores como Kubernetes.
- **Seguridad y autenticación** integradas mediante OAuth 2.0 o JWT (JSON Web Tokens).
- **Escalabilidad automática** gracias a la integración con plataformas en la nube como AWS o GCP.

7.2 Aplicaciones prácticas:

- **Educación**: Plataformas de e-learning con contenidos en streaming y control de avance.

- **Empresas**: Soluciones internas para capacitación de empleados mediante video on-demand.

- **Entretenimiento**: Aplicaciones para creadores de contenido que permiten gestionar y monetizar sus videos.

- **Salud**: Transmisión de seminarios médicos y capacitaciones a distancia en tiempo real.

7.3 Reflexión final:

La implementación de este sistema fue una oportunidad para aplicar de forma integrada los conocimientos de programación orientada a objetos, servicios web y pruebas de software. Visualizar cómo puede evolucionar a futuro reafirma la importancia de diseñar soluciones escalables, seguras y bien estructuradas desde sus primeras versiones.

8. Conclusiones

Este proyecto permitió consolidar conocimientos en programación orientada a objetos, estructuras de datos, manejo de errores y concurrencia, aplicándolos en un sistema realista y funcional. La incorporación de servicios web REST amplió la accesibilidad y escalabilidad del sistema.

Se identificaron áreas para mejoras futuras, como interfaz gráfica, seguridad avanzada y autenticación de usuarios. En general, el trabajo final demuestra la capacidad para diseñar, implementar y probar sistemas complejos, integrando teoría y práctica en soluciones tecnológicas.

REFERENCIAS

Donovan, A. A., & Kernighan, B. W. (2015). The Go Programming Language. Addison-Wesley Professional.

Google. (s. f.). Go Documentation. Recuperado el 29 de junio de 2025, de https://golang.org/doc/

GitHub. (s. f.). Gorilla Mux - A powerful URL router and dispatcher for golang. Recuperado el 29 de junio de 2025, de https://github.com/gorilla/mux

Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures (Doctoral dissertation, University of California, Irvine).

JSON.org. (s. f.). Introducing JSON. Recuperado el 29 de junio de 2025, de https://www.json.org/json-en.html

Postman. (s. f.). API Platform for Building and Using APIs. Recuperado el 29 de junio de 2025, de https://www.postman.com/