Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Análisis y Diseño de Sistemas 2
Secciones A

Catedráticos:
Ing. José Manuel Ruíz Juarez
Tutores:
Ing. Francisco Yuman

Sistema de gestión para un servicio de cable

INTEGRANTES	
201908338	Mario Jose Rodriguez Vasquez
201700761	Elder Anibal Pum Rojas
201901758	Erick Ivan Mayorga Rodríguez
201908359	Alexis Marco Tulio Lopez Cacoj

Arquitectura propuesta

Arquitectura de Microservicios

Se llegó a la elección de la arquitectura de microservicios para el sistema de gestión de un servicio de cable, esto debido a varias consideraciones que benefician la eficiencia, escalabilidad y mantenimiento del sistema.

A continuación, se listan las razones de la elección de esta arquitectura:

1. **Descomposición por módulos:** La arquitectura de microservicios descompone la aplicación en servicios pequeños, independientes y especializados, permitiendo la gestión focalizada de áreas específicas del negocio, como clientes, facturación y contenido. La capacidad de desarrollar y desplegar cada microservicio de manera independiente simplifica la administración de módulos específicos sin afectar otras partes del sistema.
2. **Mejor escalabilidad:** La escalabilidad granular de los microservicios permite ampliar selectivamente aquellos servicios que experimentan una mayor demanda, como la gestión de clientes durante períodos de inscripción o la facturación al final del mes. Esta aproximación mejora la eficiencia en el uso de recursos y reduce los costos asociados.
3. **Despliegue continuo y rápido:** La arquitectura de microservicios facilita el despliegue continuo y rápido, posibilitando la implementación ágil de actualizaciones y nuevas características en servicios específicos sin afectar globalmente al sistema. Esta flexibilidad es esencial para introducir mejoras de manera eficiente y responder rápidamente a los cambios en los requisitos del negocio.
4. **Mantenimiento fácil de realizar:** La descomposición en microservicios simplifica el mantenimiento a largo plazo, permitiendo a los desarrolladores concentrarse en áreas específicas del sistema. Las actualizaciones o correcciones se aplican de manera más localizada, mejorando la capacidad de mantenimiento y reduciendo el riesgo de errores en otras partes del sistema.
5. **Fácil adaptabilidad en casos futuros:** La independencia de los microservicios proporciona una base sólida para adaptarse a cambios futuros, como la introducción de nuevos servicios o la expansión del negocio a diferentes áreas. Este enfoque estratégico garantiza una fácil adaptación y evolución del sistema a medida que se presentan nuevas necesidades o oportunidades.

Análisis

El enunciado proporciona varias razones para elegir la arquitectura de microservicios para el sistema de gestión de un servicio de cable. Estas razones incluyen:

- **Descomposición por módulos:** La arquitectura de microservicios descompone el sistema en servicios pequeños e independientes. Esto facilita la gestión del sistema y la implementación de cambios.
- **Mejor escalabilidad:** La arquitectura de microservicios permite escalar el sistema de manera granular. Esto significa que solo se pueden escalar los servicios que experimentan una mayor demanda.
- **Despliegue continuo y rápido:** La arquitectura de microservicios facilita el despliegue continuo y rápido de nuevos cambios.
- **Mantenimiento fácil de realizar:** La arquitectura de microservicios facilita el mantenimiento del sistema. Los desarrolladores pueden centrarse en áreas específicas del sistema y las actualizaciones o correcciones se aplican de manera más localizada.
- **Fácil adaptabilidad en casos futuros:** La arquitectura de microservicios proporciona una base sólida para adaptarse a cambios futuros.

Opinión

En base al análisis anterior, se considera que la arquitectura de microservicios es una excelente decisión para el sistema de gestión de cable. Esta arquitectura proporciona una serie de beneficios que son importantes para este tipo de sistemas, como la escalabilidad, la adaptabilidad y la facilidad de mantenimiento.

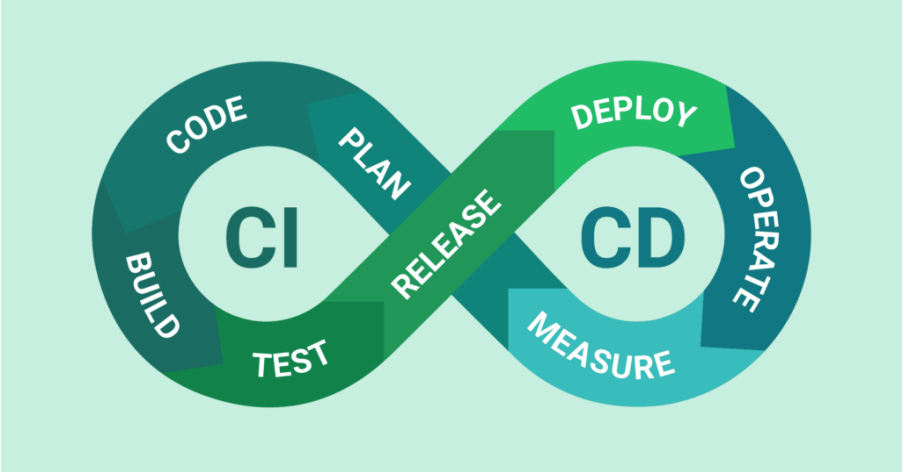
Sin embargo, es importante tener en cuenta que la elección de la arquitectura adecuada depende de una serie de factores, como los requisitos del sistema, las limitaciones de tiempo y presupuesto, y las preferencias del equipo de desarrollo que trabajará dentro del proyecto.

Método DevOps

Integración Continua/Entrega Continua (CI/CD):

Esta práctica permite automatizar el proceso de prueba y despliegue de software, asegurando que las nuevas características, arreglos y actualizaciones se entreguen de manera rápida y confiable.

La CI/CD es esencial para un desarrollo ágil y para mantener un ciclo de retroalimentación constante con el usuario final. Además, apoya la filosofía de microservicios al permitir despliegues independientes por servicio.



Planificación:

Requisitos: Realizar sesiones con stakeholders utilizando herramientas como JIRA o Trello para gestionar las historias de usuario y los epics.
Alcance: Definir el MVP (Producto Mínimo Viable) y futuras iteraciones utilizando metodologías ágiles.
Arquitectura y Tecnología: Seleccionar una arquitectura basada en microservicios utilizando Docker para contenedores y Kubernetes para orquestación.
Cronograma: Utilizar MS Project o GanttProject para desarrollar un diagrama de Gantt.

Codificación:

Base de Código: Establecer repositorios en GitHub y seguir la metodología Gitflow.
Funcionalidades: Programar utilizando lenguajes como Python o Node.js para servicios backend.
Modularidad: Adoptar patrones de diseño que promuevan la reutilización de código.
Control de Versiones: Emplear Git para el control de versiones con revisiones de código mediante pull requests.

Construcción (Build):

Compilación: Configurar pipelines de CI/CD con Jenkins o GitLab CI para automatizar los builds.
Integración Continua: Utilizar SonarQube para análisis estático de calidad de código.
Dependencias: Gestionarlas con herramientas como npm para Node.js.

Pruebas (Test):

Casos de Prueba: Implementar pruebas automáticas con frameworks como Selenium para pruebas de UI.
Integración y Sistema: Automatizar pruebas de integración y de sistema con TestNG o Cucumber.
CI/CD: Integrar las pruebas en los pipelines que se ejecuten en cada commit.

Despliegue:

Automatización: Utilizar Ansible o Terraform para automatizar el despliegue de infraestructuras.
Reproducibilidad: Implementar Infraestructura como Código (IaC) para entornos consistentes.
Escalabilidad: Desplegar en la nube con proveedores como AWS, Azure o GCP usando servicios como AWS ECS o Google Kubernetes Engine.

Operación:

Gestión de Infraestructura: Utilizar herramientas como Chef o Puppet para la configuración de servidores.
Backups: Automatizar backups utilizando servicios de nube como AWS RDS o herramientas como Bacula.

Monitoreo:

Estado del Sistema: Configurar ELK Stack (Elasticsearch, Logstash, Kibana) para el análisis de logs.
Alertas: Utilizar Alertmanager con Prometheus para gestionar y enviar notificaciones de alertas.
Rendimiento y Uso: Analizar métricas de uso con Google Analytics para la interfaz de usuario y APMs como New Relic para aplicaciones.

Lanzamiento (Release):

Planificación de Lanzamientos: Coordinar versiones con calendarios de lanzamiento en herramientas como Calendly.
Marketing y Soporte: Preparar materiales de lanzamiento y formación para diseño y Zoom para webinars.
Documentación: documentar las versiones y los cambios.

Iteración y Mejora Continua:

Feedback: Recoger comentarios de usuarios a través de herramientas como SurveyMonkey o directamente en la plataforma con Intercom.
Mejora: Analizar datos de uso y rendimiento para iterar sobre el producto utilizando metodologías Lean.
Desarrollo: Planificar sprints de desarrollo con Scrum y en plataformas, como JIRA.

Diagrama ER

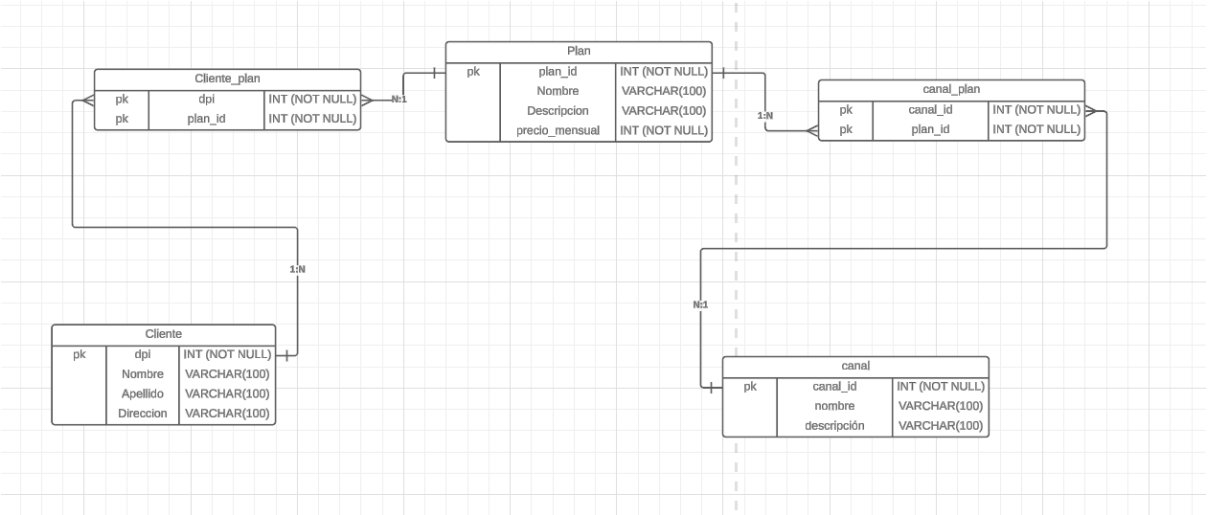


Tabla Cliente

Esta tabla almacena información sobre los clientes de la empresa. Los campos de la tabla son los siguientes:

- dpi: Identificador único del cliente.
- nombre: Nombre del cliente.
- apellido: Apellido del cliente.
- dirección: Dirección del cliente.

Tabla Plan

Esta tabla almacena información sobre los planes de televisión por cable que ofrece la empresa. Los campos de la tabla son los siguientes:

- plan_id: Identificador único del plan.
- nombre: Nombre del plan.
- descripcion: Descripción del plan.
- precio_mensual: Precio mensual del plan.

Tabla Cliente_plan

Esta tabla relaciona a los clientes con los planes a los que están suscritos. Los campos de la tabla son los siguientes:

- cliente_id: Identificador del cliente.
- plan_id: Identificador del plan.

Las relaciones entre las tablas son las siguientes:

- Una relación N:1 entre la tabla Cliente y la tabla Cliente_plan. Esta relación indica que un cliente puede estar suscrito a uno o más planes.
- Una relación 1:N entre la tabla Cliente_plan y la tabla Plan. Esta relación indica que un plan puede estar suscrito por uno o más clientes.

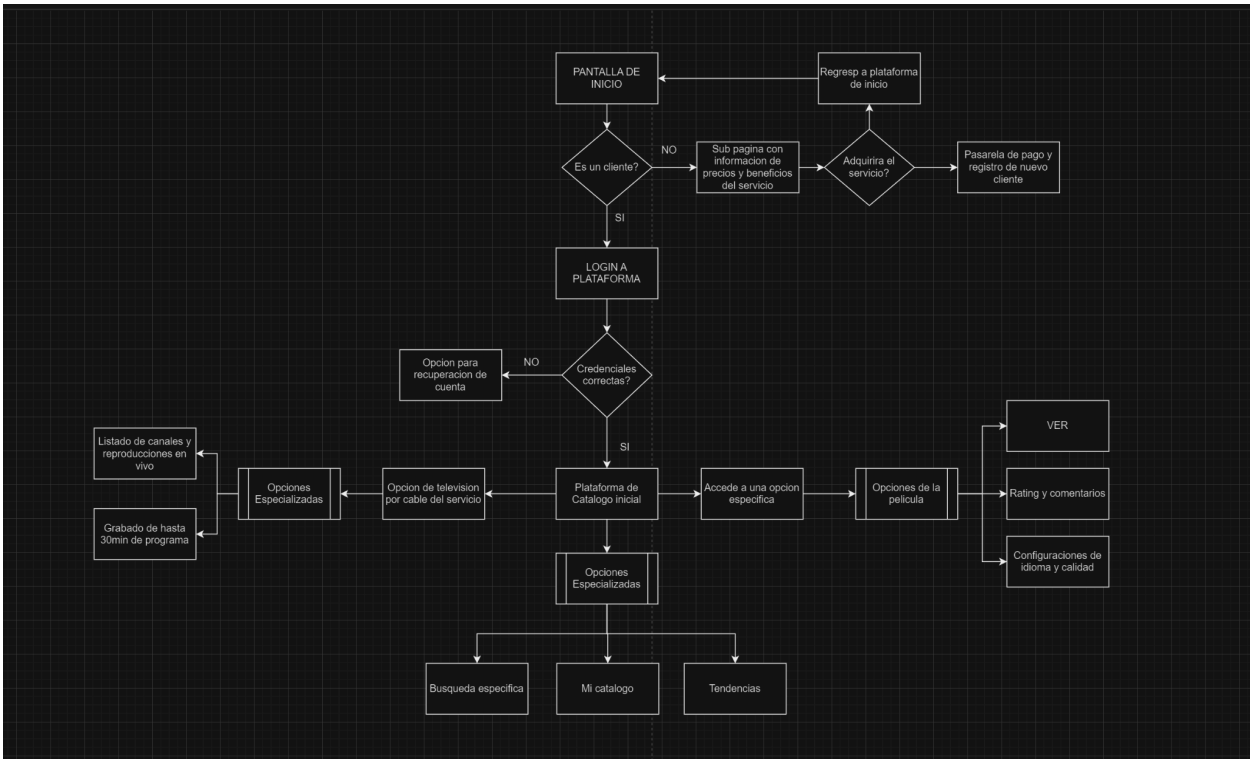
Tabla Canal

Esta tabla almacena información sobre los canales.

- canal_id: identificador único del canal
- nombre: nombre del canal
- descripción: nombre del canal

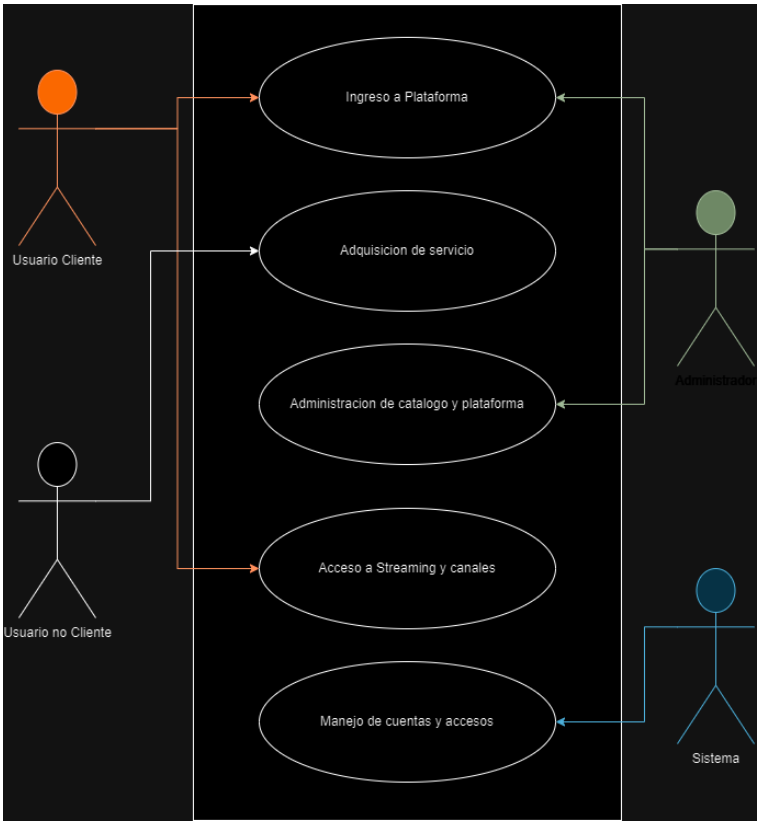
Diseño de módulos y flujos

Diagrama de Flujo del servcio



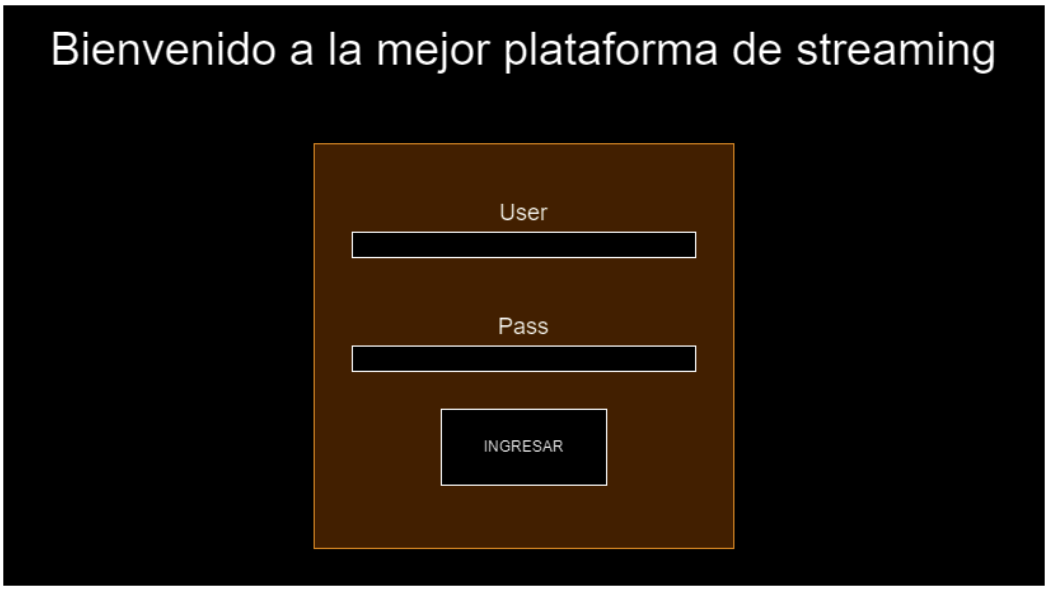
A continuación se presenta el diagrama de flujo del servicio de cable y streaming el cual va dirigido a la explicacion del mismo. Para presentar este servicio de forma especifica, se hacen las siguientes explicaciones de modulos y descripciones de flujos bajo el siguiente esquema de clases de bajo nivel.

Para entender la posibilidad de usos de la plataforma y servicios se muestra el siguiente diagrama de casos de uso.



A continuación se describen de forma específica algunos de los módulos a utilizar dentro del servicio.

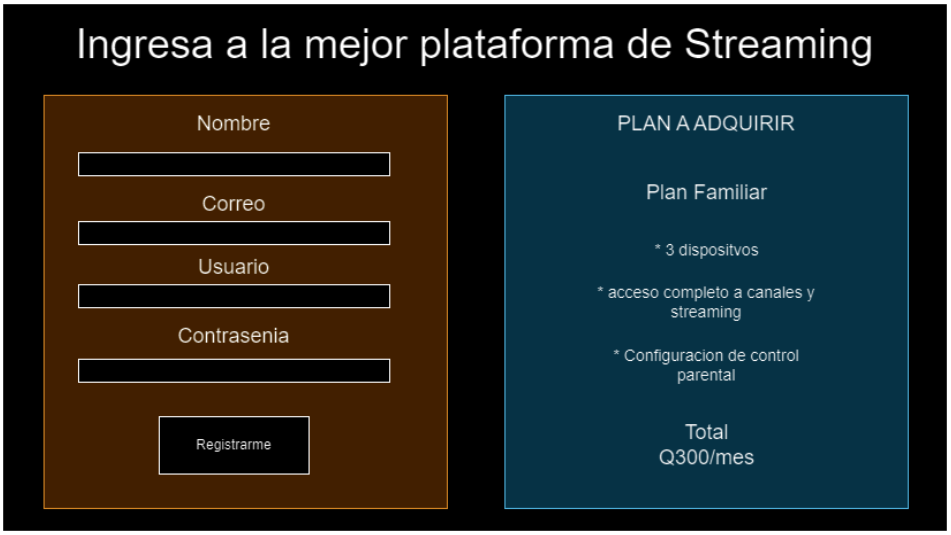
Login



Este modulo se manejara por ingreso y validacion de usuario y contraseña cifrada, un ejemplo del modelo de datos a utilizar es:

{
“user”: “hola”,
“pass”: “hola”
}

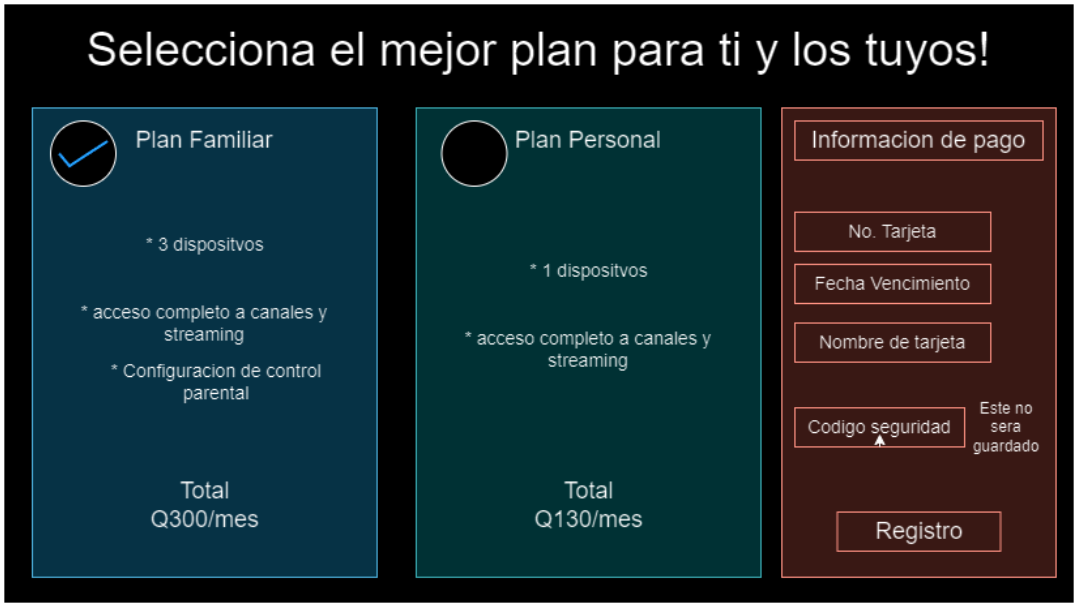
Registro



Para el sistema de registro, el usuario tuvo que haber escogido previamente el plan deseado para adquisicion junto a la informacion de pago, dejando unicamente datos de registro en este apartado, el registro de datos esperado es.

{
“Nombre”: “hola”,
“Correo”: “hola”,
“Usuario”: “hola”,
“Contrasenia”: “hola”
}

Adquisición de servicio



En este apartado el usuario podrá escoger el plan deseado para su adquisición y configurar los métodos de pago que utilizará.

{

“Plan”: 1,

“No Tarjeta”: 12345678945,

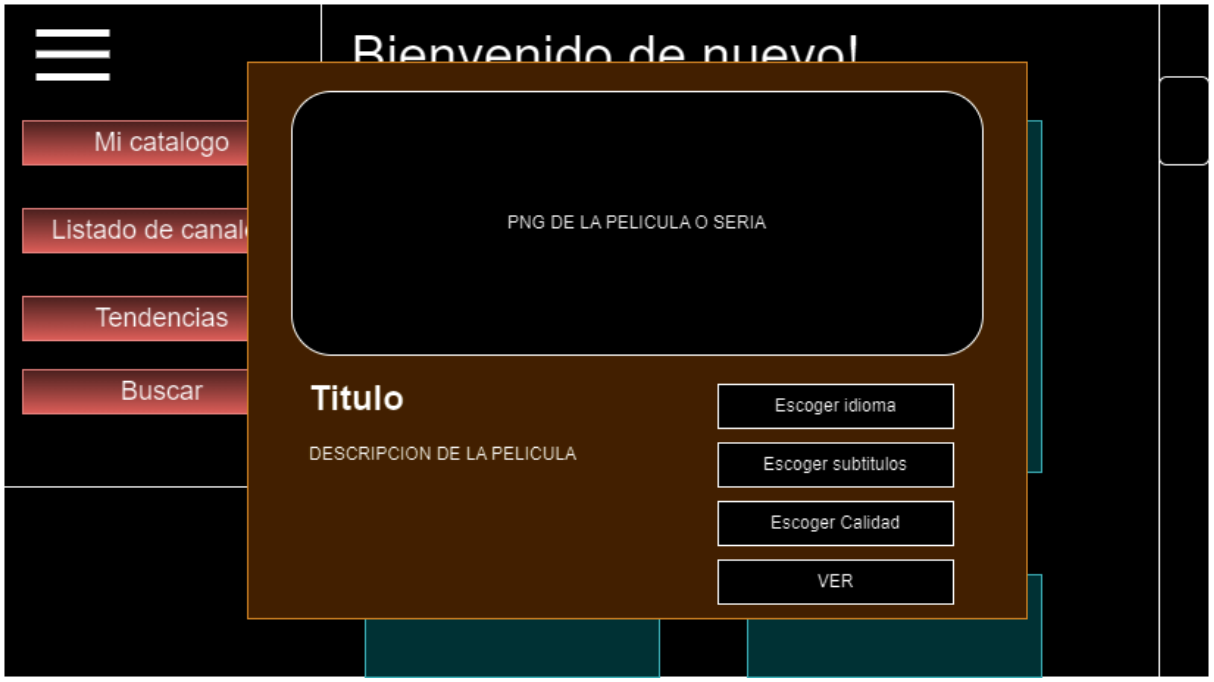
“FechaV”: “12/24”

}

Catálogo Principal con funcionalidades específicas

En este apartado el usuario podrá visualizar el catálogo completo de opciones para escoger, así como un submenu desplegable con funcionalidades específicas. Este no necesita un esquema de datos saliente.

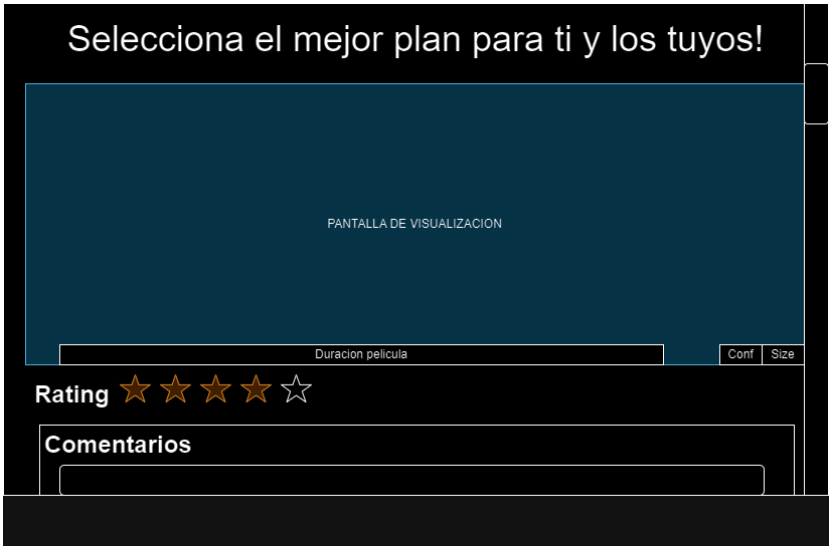
Opciones de prevista para opción escogida



En este apartado se muestra una preconfiguracion de visualización de la pelicula o serie seleccionada para su visualización, los datos salientes para su funcionalidad son:

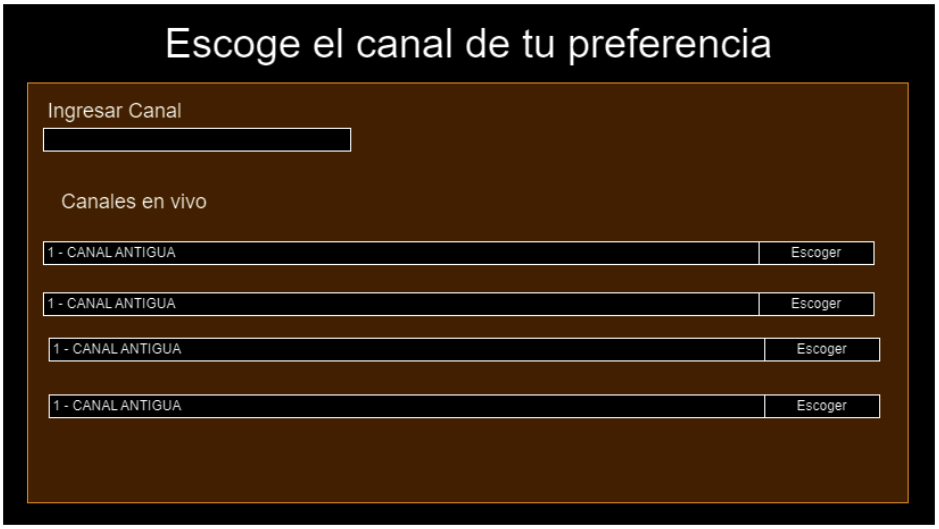
```
{
        “idioma”: 1,
        “subtitulos”: 2,
        “Calidad”: 3
}
```

Pantalla de visualización de streaming



En este apartado se podrá visualizar la opción seleccionada, así como ver un sistema de calificación y comentarios en la parte inferior del mismo. No necesita un esquema de datos salientes para su utilización.

Listado de canales



Este apartado mostrará un listado de canales disponibles dentro del sistema, en el cual podrá ingresar por búsqueda específica o escoger dentro del listado en vivo lo deseado.

```
{
        “canal”: 52
}
```

Diagrama Gantt

Explicación de las actividades

- Hito 1: Requisitos definidos: Este hito se alcanza cuando los estudiantes han definido los requisitos funcionales y no funcionales del sistema.
- Hito 2: Modelo de datos diseñado: Este hito se alcanza cuando los estudiantes han diseñado el modelo de datos que se utilizará para almacenar la información del sistema.
- Hito 3: Arquitectura del sistema diseñada: Este hito se alcanza cuando los estudiantes han diseñado la arquitectura del sistema, que incluye la estructura de los componentes del sistema y las relaciones entre ellos.
- Hito 4: Sistema implementado: Este hito se alcanza cuando los estudiantes han implementado el sistema utilizando un lenguaje de programación y una plataforma de desarrollo.
- Hito 5: Sistema probado: Este hito se alcanza cuando los estudiantes han probado el sistema para garantizar que cumpla con los requisitos.

Observaciones

- Las fechas se han ajustado para que el proyecto se complete en un plazo de 30 días.
- Las actividades se han reorganizado para que las tareas más importantes se completen primero.
- Los hitos se han mantenido en las mismas fechas para proporcionar puntos de referencia claros para el progreso del proyecto.

Actividades adicionales

Las actividades adicionales que se pueden agregar al diagrama de Gantt son las siguientes:

- Desarrollo de la documentación del usuario: Esta actividad consiste en crear la documentación que los usuarios utilizarán para aprender a utilizar el sistema.
- Desarrollo de la documentación técnica: Esta actividad consiste en crear la documentación que los desarrolladores utilizarán para mantener y actualizar el sistema.
- Capacitación de los usuarios: Esta actividad consiste en enseñar a los usuarios cómo utilizar el sistema.

Diagrama de gantt
Proyecto: Análisis y diseño de un sistema de gestión para un servicio de cable
Fecha de incio: 28 de diciembre de 2023
Fecha de finalización: 28 de enero 2024