1. **Referencias Bibliográficas: Links**

<https://www.redalyc.org/journal/160/16057381018/16057381018.pdf>

<https://sedici.unlp.edu.ar/handle/10915/119967>

<https://www.e-revistes.uji.es/index.php/adcomunica/article/view/5027>

1. **Requerimientos Funcionales y No Funcionales**

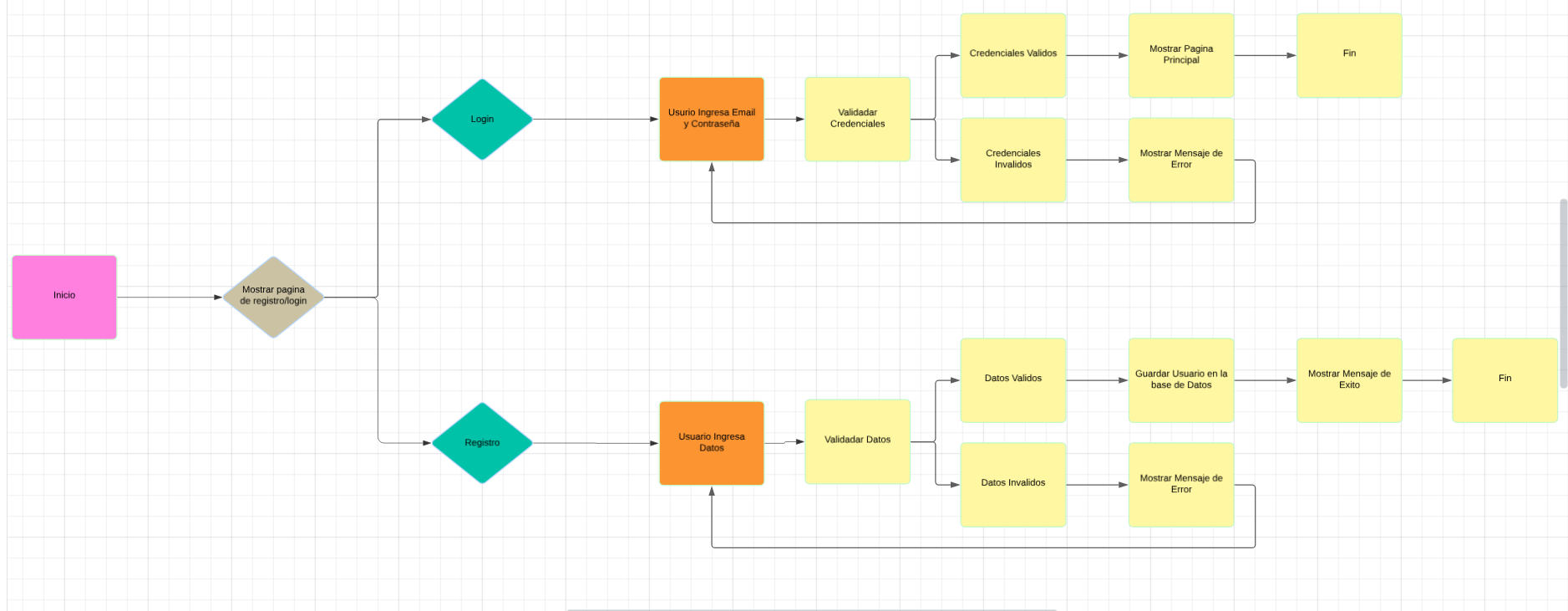
|  |  |
| --- | --- |
| Funcionales | No Funcionales |
| Permitir a los usuarios registrarse y crear cuentas. | La plataforma debe ser capaz de soportar al menos 10 millones de usuarios concurrentes. |
| Permitir a los usuarios iniciar sesión y cerrar sesión. | La plataforma debe tener una disponibilidad del 99.9%. |
| Permitir a los usuarios buscar películas y series. | Las páginas deben cargarse en menos de 2 segundos. |
| Permitir a los usuarios reproducir videos en streaming. | Los videos deben reproducirse en calidad HD (720p) o superior. |
| Proveer recomendaciones personalizadas basadas en el historial del usuario. | La plataforma debe estar disponible en múltiples idiomas. |
| Permitir a los usuarios crear y gestionar listas de reproducción. | La plataforma debe ser compatible con múltiples dispositivos (PC, móvil, Smart TV). |
| Ofrecer diferentes planes de suscripción. | Los datos del usuario deben estar encriptados para garantizar la seguridad. |
| Permitir a los usuarios descargar contenido para ver offline. | La plataforma debe cumplir con las regulaciones de privacidad de datos, como GDPR. |
| Proveer una sección de ayuda y soporte técnico. | Las actualizaciones de la plataforma deben realizarse sin interrumpir el servicio. |
| Permitir la evaluación y calificación del contenido por parte de los usuarios. | La plataforma debe ser escalable para manejar incrementos en la carga de usuarios. |

1. **Atributos y Servicios**

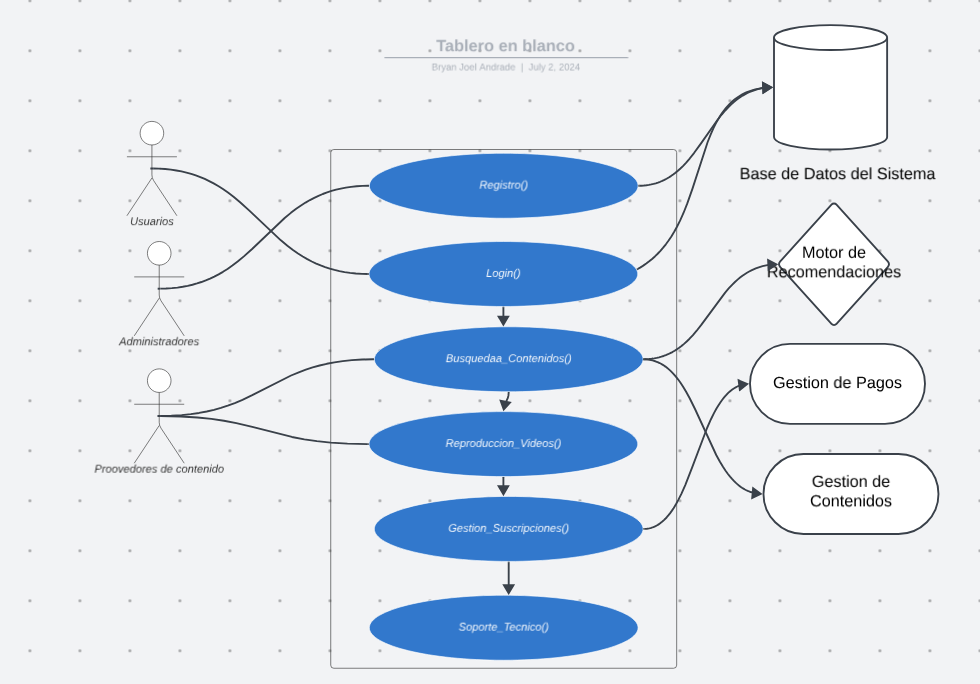
|  |  |
| --- | --- |
| Atributos | Servicios. |
| usuario | Gestion\_Usuarios() |
| pelicula\_serie | Gestion\_Contenidos() |
| suscripcion | Reproduccion\_Videos() |
| reproducción | Gestion\_Suscripciones() |
|  | Soporte\_Tecnico() |

1. **Diagramas: flujo, frontera y clases**

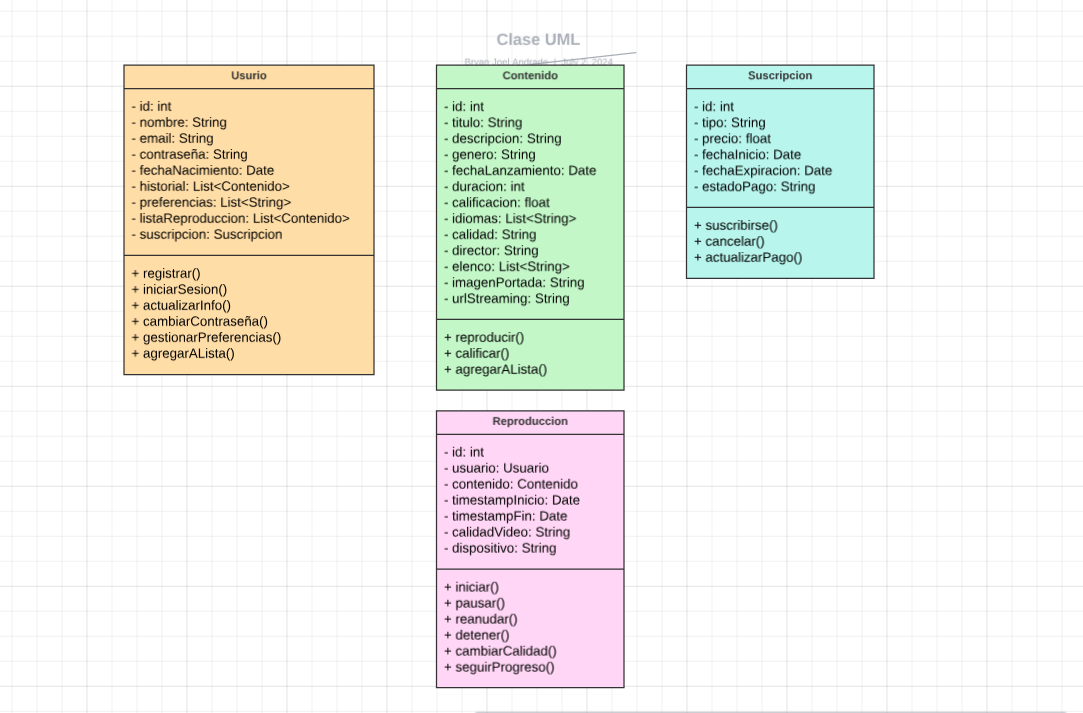
**Diagrama de Flujo**

****

**Diagrama de Frontera**

****

**Diagrama de Clases**

****

1. **Diseño de Pseudocódigo**

**Registro Usuario**

Procedimiento RegistrarUsuario(nombre, email, contraseña, fechaNacimiento)

Si UsuarioExiste(email) Entonces

Mostrar "El usuario ya existe."

Sino

usuarioNuevo <- CrearUsuario(nombre, email, contraseña, fechaNacimiento)

GuardarUsuario(usuarioNuevo)

Mostrar "Registro exitoso."

FinSi

FinProcedimiento

Función UsuarioExiste(email)

usuario <- BuscarUsuarioPorEmail(email)

Si usuario es NULL Entonces

Retornar Falso

Sino

Retornar Verdadero

FinSi

FinFunción

Función CrearUsuario(nombre, email, contraseña, fechaNacimiento)

usuario <- Nuevo Usuario

usuario.nombre <- nombre

usuario.email <- email

usuario.contraseña <- Hash(contraseña)

usuario.fechaNacimiento <- fechaNacimiento

usuario.historial <- ListaVacia

usuario.preferencias <- ListaVacia

usuario.listaReproduccion <- ListaVacia

usuario.suscripcion <- NULL

Retornar usuario

FinFunción

Procedimiento GuardarUsuario(usuario)

BaseDeDatos.Insertar("usuarios", usuario)

FinProcedimiento

**Inicio de Sesion**

Procedimiento IniciarSesion(email, contraseña)

usuario <- BuscarUsuarioPorEmail(email)

Si usuario es NULL Entonces

Mostrar "Usuario no encontrado."

SinoSi VerificarContraseña(usuario, contraseña) Entonces

CrearSesion(usuario)

Mostrar "Inicio de sesión exitoso."

Sino

Mostrar "Contraseña incorrecta."

FinSi

FinProcedimiento

Función VerificarContraseña(usuario, contraseña)

Retornar Hash(contraseña) = usuario.contraseña

FinFunción

Procedimiento CrearSesion(usuario)

sesion <- Nueva Sesion

sesion.usuario <- usuario

sesion.timestamp <- ObtenerTimestampActual()

GuardarSesion(sesion)

FinProcedimiento

Procedimiento GuardarSesion(sesion)

BaseDeDatos.Insertar("sesiones", sesion)

FinProcedimiento

**Busqueda de Contenidos**

Procedimiento BuscarContenido(criterio)

resultados <- BaseDeDatos.Buscar("contenidos", criterio)

MostrarResultados(resultados)

FinProcedimiento

Procedimiento MostrarResultados(resultados)

Para cada contenido en resultados

Mostrar contenido.titulo, contenido.descripcion, contenido.genero

FinPara

FinProcedimiento

**Reproduccion de Videos**

Procedimiento ReproducirVideo(idUsuario, idContenido)

usuario <- BuscarUsuarioPorId(idUsuario)

contenido <- BuscarContenidoPorId(idContenido)

Si contenido es NULL Entonces

Mostrar "Contenido no encontrado."

SinoSi UsuarioTieneAcceso(usuario, contenido) Entonces

IniciarReproduccion(usuario, contenido)

Sino

Mostrar "Acceso denegado."

FinSi

FinProcedimiento

Función UsuarioTieneAcceso(usuario, contenido)

Si usuario.suscripcion es NULL Entonces

Retornar Falso

SinoSi usuario.suscripcion.estadoPago = "activo" Entonces

Retornar Verdadero

Sino

Retornar Falso

FinSi

FinFunción

Procedimiento IniciarReproduccion(usuario, contenido)

reproduccion <- Nueva Reproduccion

reproduccion.usuario <- usuario

reproduccion.contenido <- contenido

reproduccion.timestampInicio <- ObtenerTimestampActual()

GuardarReproduccion(reproduccion)

Mostrar "Reproduciendo: " + contenido.titulo

FinProcedimiento

Procedimiento GuardarReproduccion(reproduccion)

BaseDeDatos.Insertar("reproducciones", reproduccion)

FinProcedimiento

**Gestion de Suscripciones**

Procedimiento Suscribirse(idUsuario, tipoSuscripcion)

usuario <- BuscarUsuarioPorId(idUsuario)

suscripcion <- CrearSuscripcion(tipoSuscripcion)

usuario.suscripcion <- suscripcion

GuardarUsuario(usuario)

Mostrar "Suscripción exitosa."

FinProcedimiento

Función CrearSuscripcion(tipo)

suscripcion <- Nueva Suscripcion

suscripcion.tipo <- tipo

suscripcion.precio <- ObtenerPrecio(tipo)

suscripcion.fechaInicio <- ObtenerFechaActual()

suscripcion.fechaExpiracion <- CalcularFechaExpiracion(suscripcion.fechaInicio, tipo)

suscripcion.estadoPago <- "activo"

Retornar suscripcion

FinFunción

Función ObtenerPrecio(tipo)

Si tipo = "basico" Entonces

Retornar 10.0

SinoSi tipo = "estandar" Entonces

Retornar 15.0

SinoSi tipo = "premium" Entonces

Retornar 20.0

FinSi

FinFunción

Función CalcularFechaExpiracion(fechaInicio, tipo)

Si tipo = "mensual" Entonces

Retornar fechaInicio + 30 dias

SinoSi tipo = "anual" Entonces

Retornar fechaInicio + 365 dias

FinSi

FinFunción

1. **Planteamiento de Algoritmo**

**Paradigma de Programacion:** POO

**Lista de librerías:**

from flask import Flask, request, jsonify, session

from flask\_sqlalchemy import SQLAlchemy

from flask\_bcrypt import Bcrypt

from flask\_login import LoginManager, UserMixin, login\_user, logout\_user, login\_required, current\_user

from datetime import datetime, timedelta

**Modulos de algoritmos:** PREREQUISITOS4:

Registrar\_usuario();

Iniciar\_sesion();

Buscar\_contenidos();

Iniciar\_reproduccion();

Suscribirse();

Obtener\_precio();

Calcular\_fecha\_expiracion();

1. **Desarrollo: Implementación del Algoritmo**
2. **Testing: Pruebas**
3. Puesta en producción