UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERÍA

Escuela Profesional de Ingeniería de Sistemas

Proyecto “DataFiller”

Curso: *Pruebas y Calidad de Software*

Docente: *Mag. Patrick Cuadros Quiroga*

Integrantes:

SEBASTIAN NICOLAS FUENTES AVALOS (2022073902)
MAYRA FERNANDA CHIRE RAMOS (2021072620)
GABRIELA LUZKALID GUTIERREZ MAMANI (2022074263)

Tacna – Perú
2025

CONTROL DE VERSIONES					
Versión	Hecha por	Revisada por	Aprobada por	Fecha	Motivo
1.0	MPV	ELV	ARV	10/10/2020	Versión Original

Sistema *DataFiller*
Documento de Arquitectura de Software

Versión *{1.0}*

CONTROL DE VERSIONES					
Versión	Hecha por	Revisada por	Aprobada por	Fecha	Motivo
1.0	MPV	ELV	ARV	10/10/2020	Versión Original

INDICE GENERAL

Contenido

1. INTRODUCCIÓN

1.1. Propósito (Diagrama 4+1)

Este documento presenta una visión integral y detallada de la arquitectura del sistema DataFiller, una plataforma web diseñada para automatizar la generación de datos de prueba realistas para bases de datos SQL y NoSQL. El propósito principal de DataFiller es proporcionar una arquitectura que cumpla con los requisitos funcionales y no funcionales del sistema, garantizando eficiencia, seguridad y una experiencia de usuario fluida. Dada la naturaleza de la aplicación, se ha elegido el estilo arquitectónico de 3 capas MVC (Modelo-Vista-Controlador), el cual permite una separación clara entre los datos, la interfaz de usuario y la lógica de control. Este documento adopta el modelo de vistas arquitectónicas 4+1, que ofrece diversas perspectivas del sistema, facilitando la comprensión y participación de los diferentes stakeholders según sus intereses. La arquitectura propuesta está diseñada para soportar la generación de datos sintéticos que respeten las relaciones entre tablas y las restricciones de integridad, además de facilitar la implementación del sistema de planes (gratuito y premium).

1.2. Alcance

El proyecto DataFiller abarcará el desarrollo de una plataforma web que permitirá a usuarios generar datos de prueba realistas para bases de datos, con las siguientes funcionalidades e inclusiones:

- Análisis automático de scripts SQL y NoSQL para detectar la estructura de tablas, relaciones y restricciones sin requerir conocimientos técnicos avanzados por parte del usuario.
- Generación de datos sintéticos realistas que respeten las relaciones entre tablas y las restricciones de integridad, con cantidades variables según el plan (10 registros por tabla en plan gratuito, cantidad ampliada en plan premium).
- Implementación de un sistema de planes con limitaciones diferenciadas:
    - Plan gratuito: 3 consultas diarias, 10 registros por tabla, formato SQL
    - Plan premium: Consultas ilimitadas, mayor cantidad de registros, todos los formatos, datos personalizados por industria
- Desarrollo de un sistema de autenticación de usuarios y gestión de suscripciones con integración a pasarela de pagos para el plan premium (S/9.99 mensual).
- Interfaz web intuitiva que permita a los usuarios pegar scripts, visualizar resultados y descargar datos generados.
- Sistema de soporte por correo electrónico para todos los usuarios, con atención prioritaria para usuarios premium.

1.3. Definición, siglas y abreviaturas

SQL: Lenguaje de Consulta Estructurado
NoSQL: Not Only SQL
BD: Base de Datos
SCPT: Scripts
IA: Inteligencia Artificial
API: Application Programming Interface
MVC: Modelo-Vista-Controlador
QA: Quality Assurance (Aseguramiento de Calidad)
DBA: Database Administrator (Administrador de Base de Datos)

GDPR: General Data Protection Regulation (Reglamento General de Protección de Datos)

LPDP: Ley de Protección de Datos Personales

PCI-DSS: Payment Card Industry Data Security Standard

B/C: Beneficio/Costo

VAN: Valor Actual Neto

TIR: Tasa Interna de Retorno

JSON: JavaScript Object Notation

1.4. Organización del documento

La estructura del documento sigue el modelo de vistas arquitectónicas 4+1, describiendo el sistema desde las perspectivas de:

- Casos de uso: Requerimientos y funcionalidades
- Lógica: Estructura del sistema MVC
- Implementación: Organización del código fuente
- Procesos: Aspectos dinámicos
- Despliegue: Infraestructura cloud y servicios en la nube para el hosting y acceso al sistema

2. OBJETIVOS Y RESTRICCIONES ARQUITECTÓNICAS

2.1. Priorización de requerimientos

ID	Descripción	Prioridad
RF01	Análisis automático de scripts SQL para detectar estructuras	Alta
RF02	Generación de datos sintéticos realistas que respeten relaciones y restricciones	Alta
RF03	Interfaz web intuitiva para pegar scripts, visualizar y descargar datos	Alta
RF04	Sistema de autenticación de usuarios	Media
RF05	Sistema de planes con limitaciones diferenciadas (gratuito/premium)	Media
RF06	Integración con pasarela de pagos para suscripciones premium	Baja

1.1.1. Requerimientos Funcionales

ID	Descripción	Prioridad
RF01	Análisis automático de scripts SQL para detectar estructuras	Alta
RF02	Generación de datos sintéticos realistas que respeten relaciones y restricciones	Alta
RF03	Interfaz web intuitiva para pegar scripts, visualizar y descargar datos	Alta
RF04	Sistema de autenticación de usuarios	Media
RF05	Sistema de planes con limitaciones diferenciadas (gratuito/premium)	Media
RF06	Integración con pasarela de pagos para suscripciones premium	Baja
RF07	Sistema de soporte por correo electrónico con prioridad según plan	Baja
RF08	Datos personalizados por industria para usuarios premium	Baja

1.1.2. Requerimientos No Funcionales – Atributos de Calidad

ID. Requerimiento	**Nombre del Requisito**	**Descripción de Requisito**	**Prioridad**
RF-001	Seguridad	El sistema debe garantizar la seguridad de los datos mediante encriptación de contraseñas	Alta

RF-002	Rendimiento	El sistema debe garantizar tiempos de respuesta rápidos al generar datos de prueba, incluso cuando los esquemas de base de datos sean grandes o complejos.	Alta
RF-003	Disponibilidad	Mantener la plataforma disponible y accesible la mayor parte del tiempo.	Alta
RF-004	Usabilidad	Diseñar una interfaz intuitiva y fácil de usar para los usuarios, permitiendo cargar esquemas de base de datos, personalizar configuraciones y generar datos con facilidad.	Media
RF-005	Mantenibilidad	Facilitar la mantenibilidad del sistema mediante código limpio, documentación adecuada y pruebas robustas.	Media
RF-006	Disponibilidad	Acceso al sistema 24/7 con mínimo tiempo de inactividad	Media

2.2. Restricciones

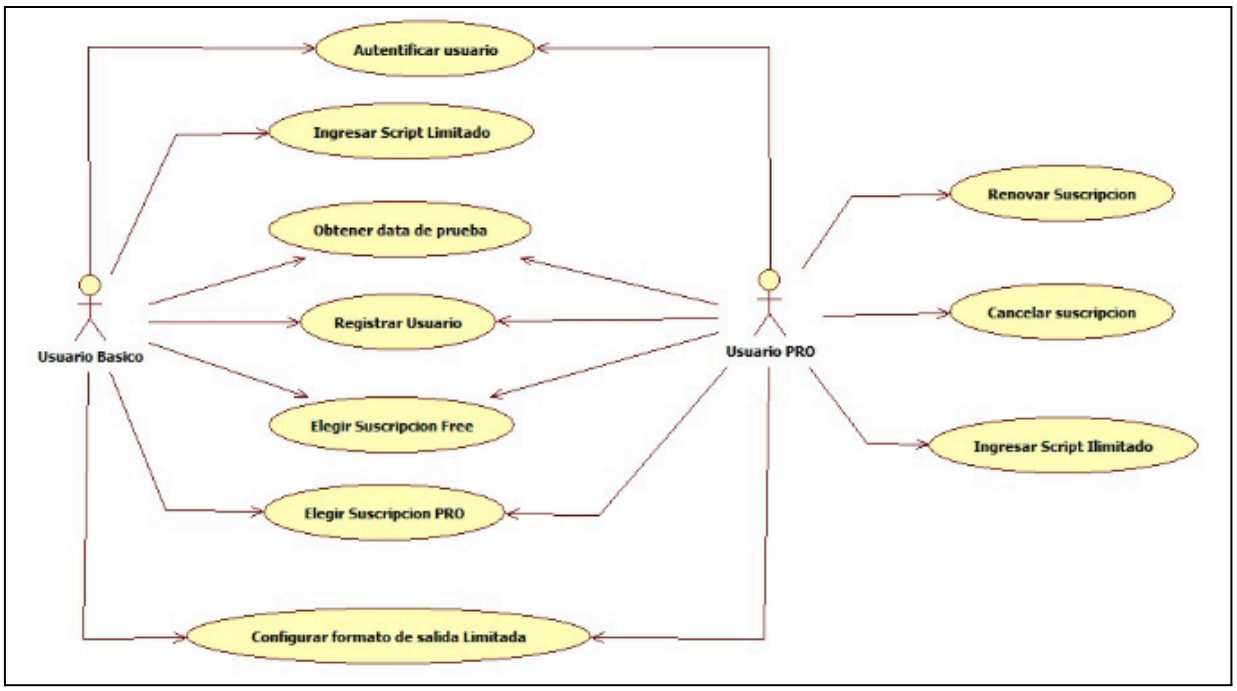
- **Generación de Datos Sintéticos Realistas**: No se generarán ni utilizarán datos personales reales en la plataforma, cumpliendo con el Reglamento General de Protección de Datos (GDPR) y la Ley de Protección de Datos Personales (LPDP).
- **Limitaciones en el Uso de Datos para Testeo en Entornos Sensibles**: Los usuarios no deben utilizar la plataforma para generar datos sensibles que puedan violar la privacidad en sectores como salud, finanzas o datos de tarjetas de crédito, conforme a leyes como HIPAA y PSD2.

- **Cumplimiento con Normativas de Pago y Transacciones**: El sistema de gestión de pagos para el Plan Premium debe cumplir con la Norma de Seguridad de Datos para la Industria de Tarjetas de Pago (PCI-DSS).
- **Tecnología**: El desarrollo se realizará utilizando PHP versión 8 para el backend, y HTML5, CSS3, JavaScript y Bootstrap para el frontend.
- **Base de Datos**: Se utilizará MySQL 8 para gestionar los datos del sistema.
- **Infraestructura**: El sistema se desplegará en un servidor dedicado con Windows Server (Elastika).

3. REPRESENTACIÓN DE LA ARQUITECTURA DEL SISTEMA

3.1. Vista de Caso de uso

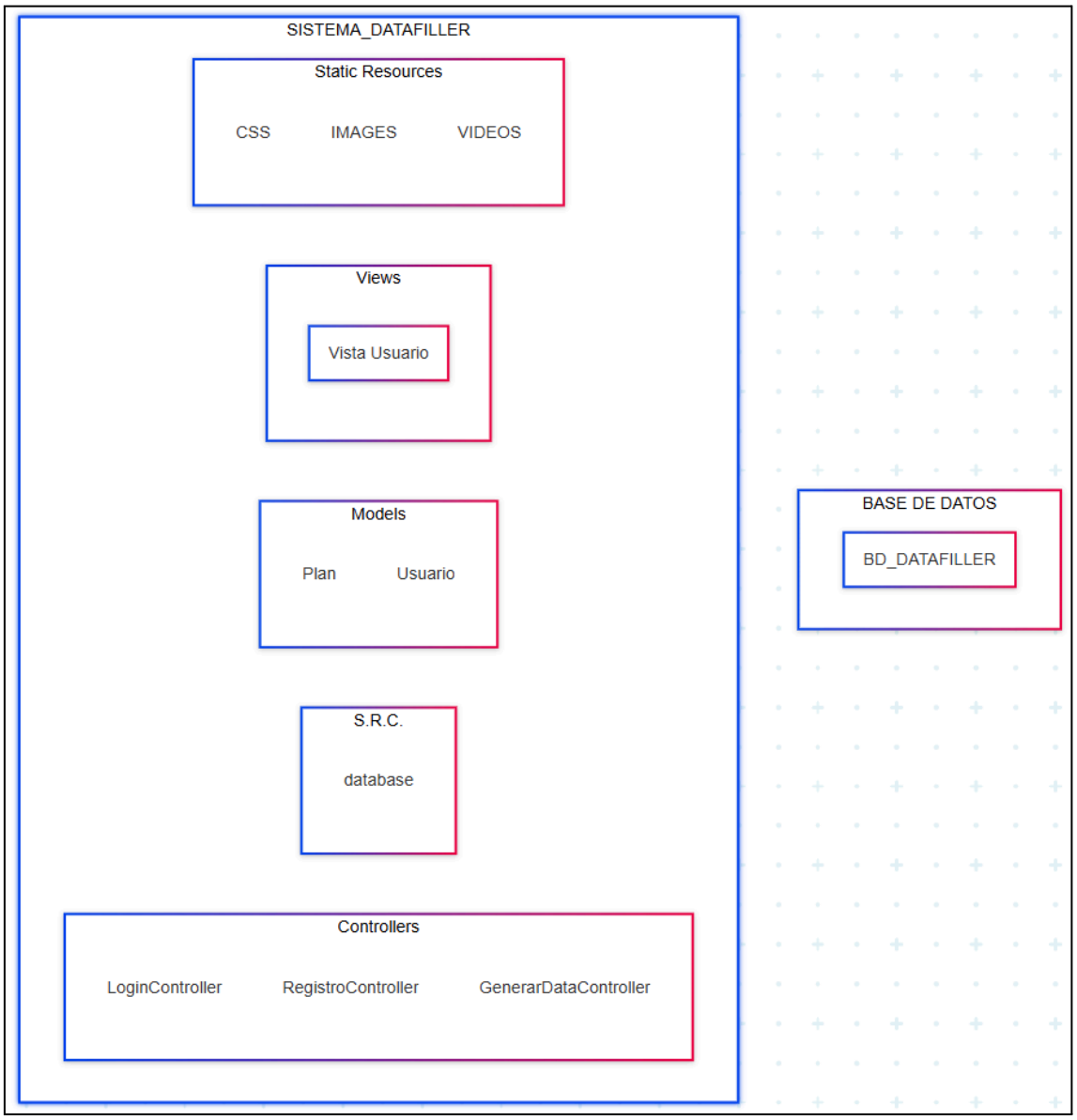
1.1.3. Diagramas de Casos de uso



3.2. Vista Lógica

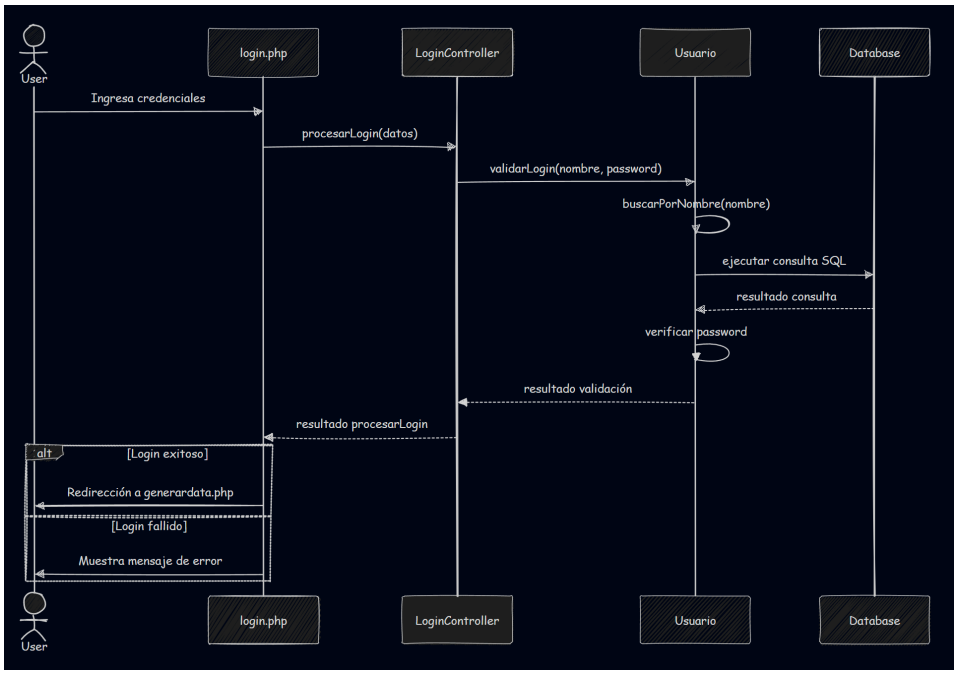
[La vista lógica se encarga de representar los requerimientos funcionales del sistema. Esta sección describe las partes del diseño del modelo significativas para la arquitectura, tales como subsistemas y paquetes.]

3.2.1. Diagrama de Subsistemas (paquetes)

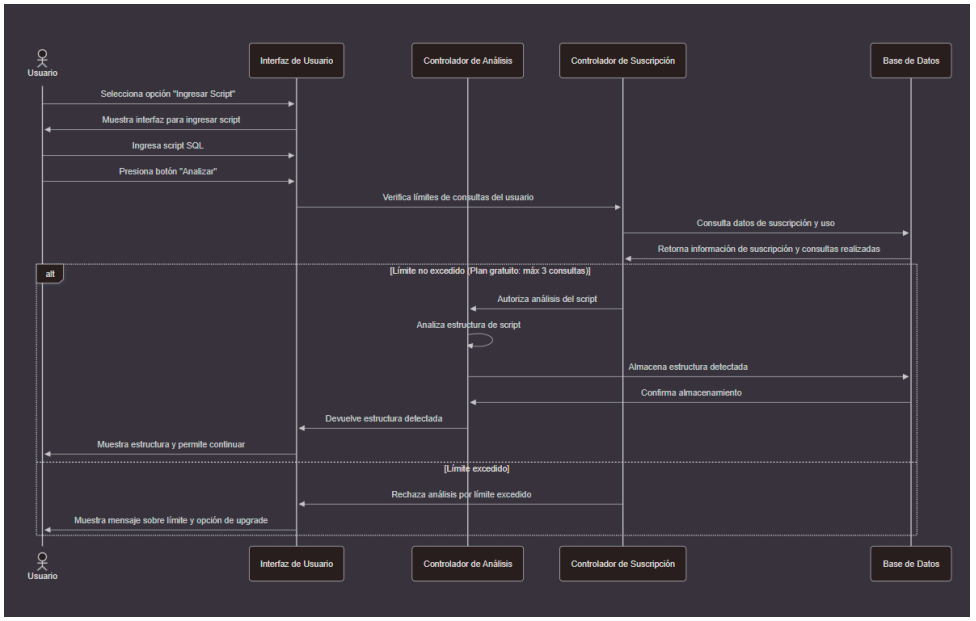


3.2.2. Diagrama de Secuencia (vista de diseño)

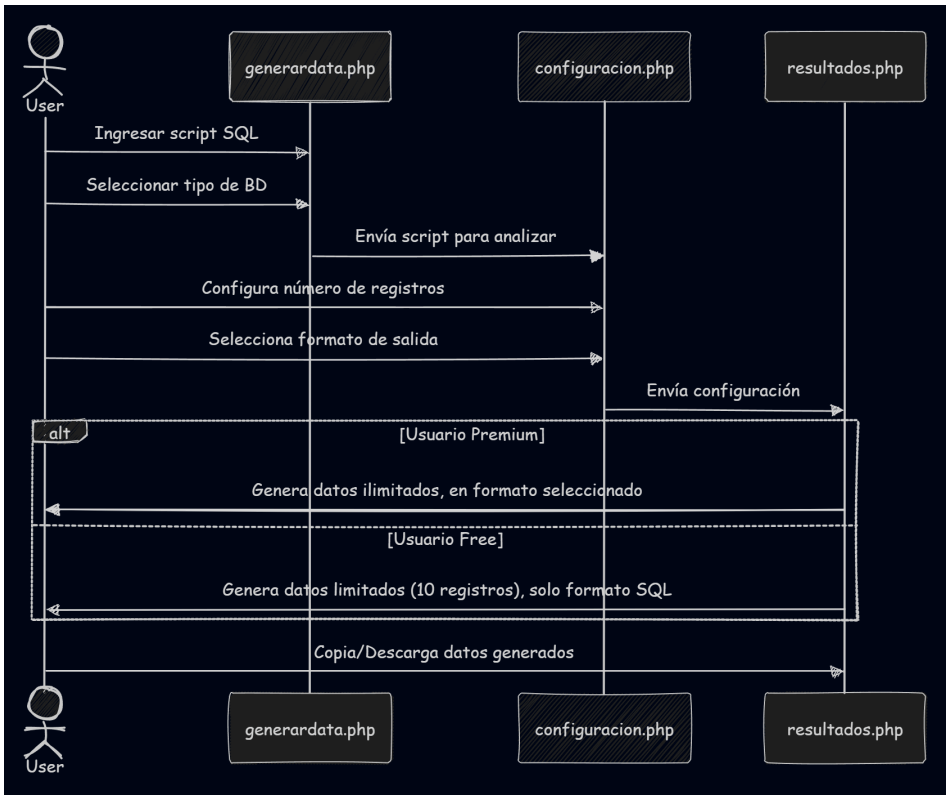
Autentificar Usuario



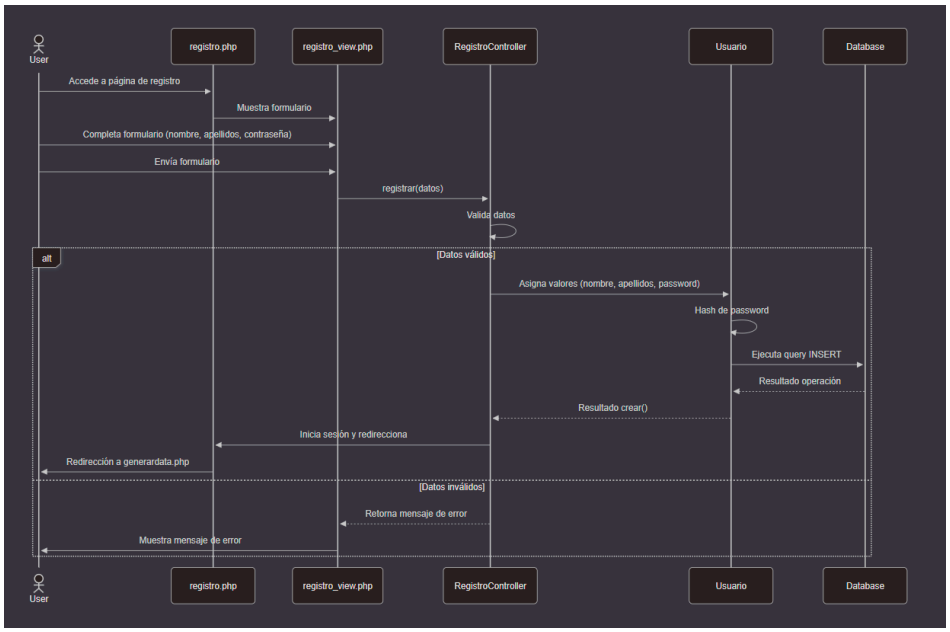
Ingresar Script Limitado



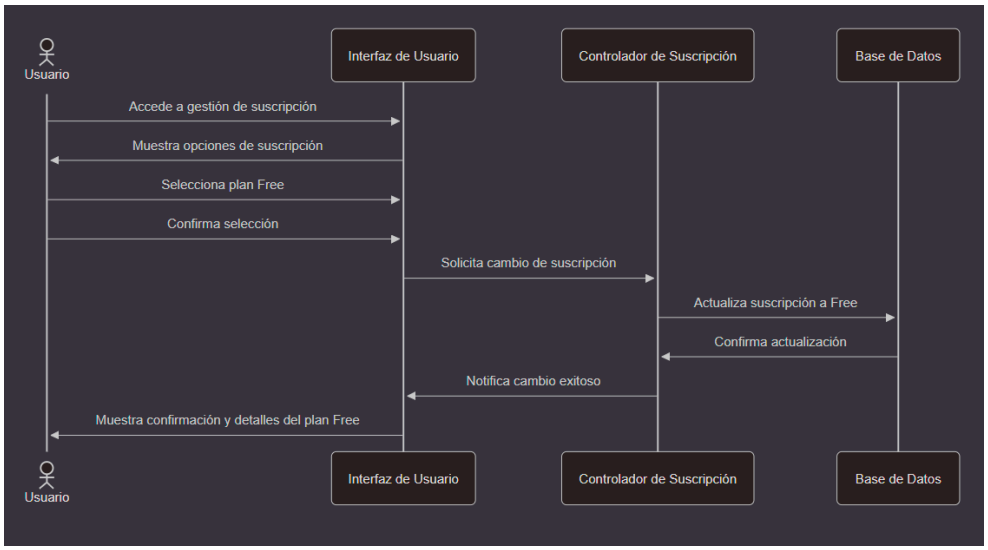
Obtener Data de Prueba



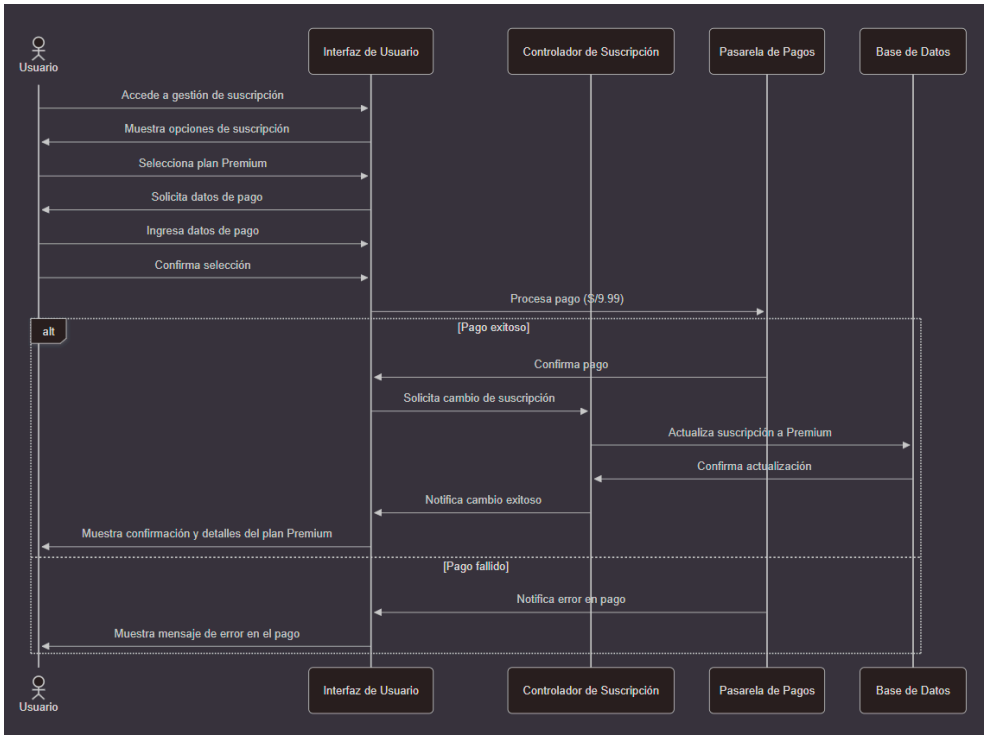
Registrar usuario



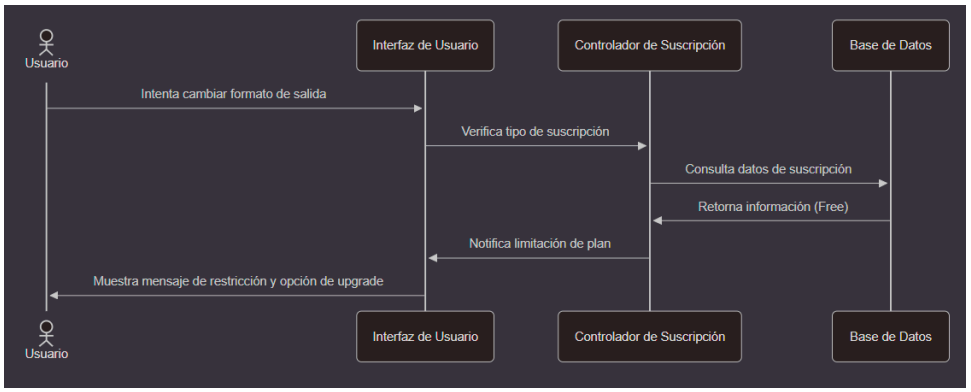
Elegir Suscripcion Free



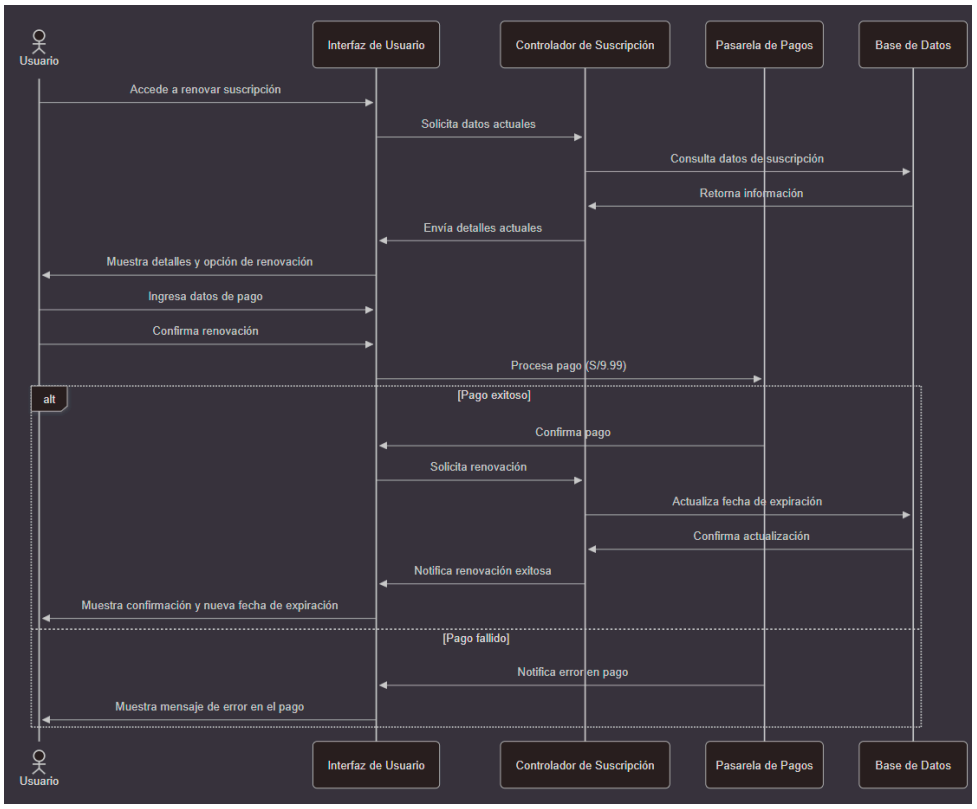
Elegir Suscripcion PRO



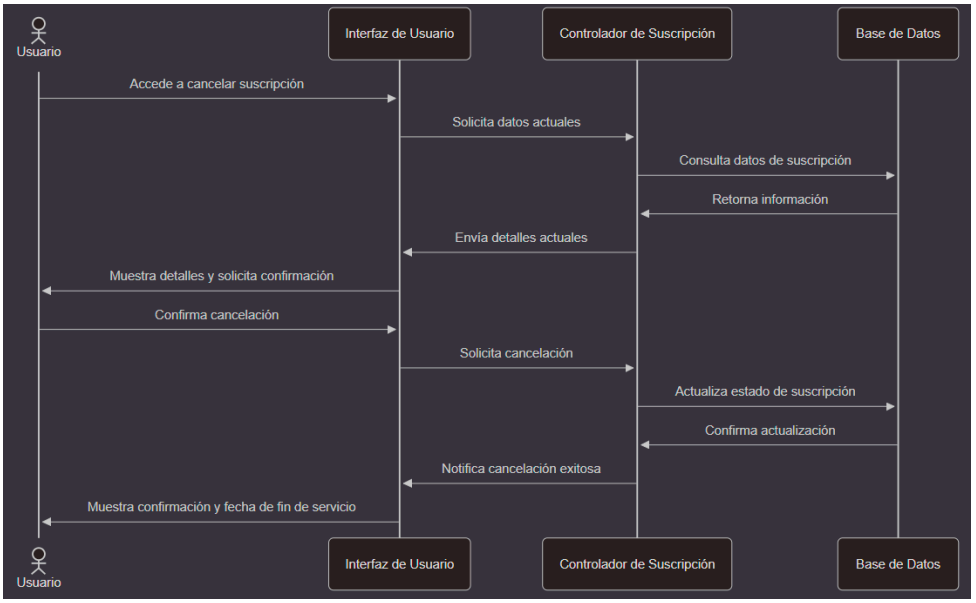
Configurar Formato de Salida Limitada



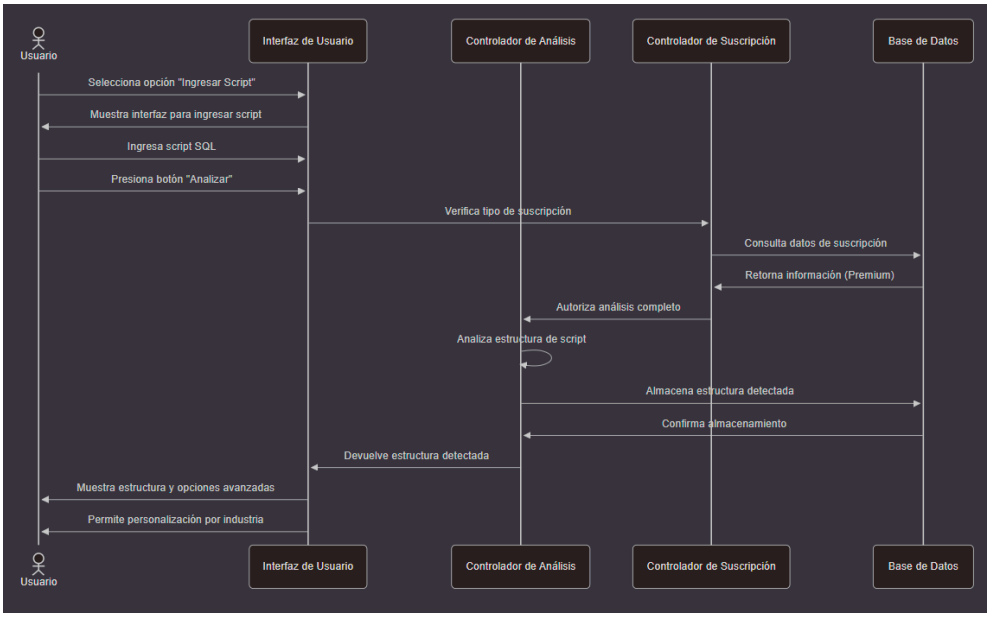
Renovar Suscripción



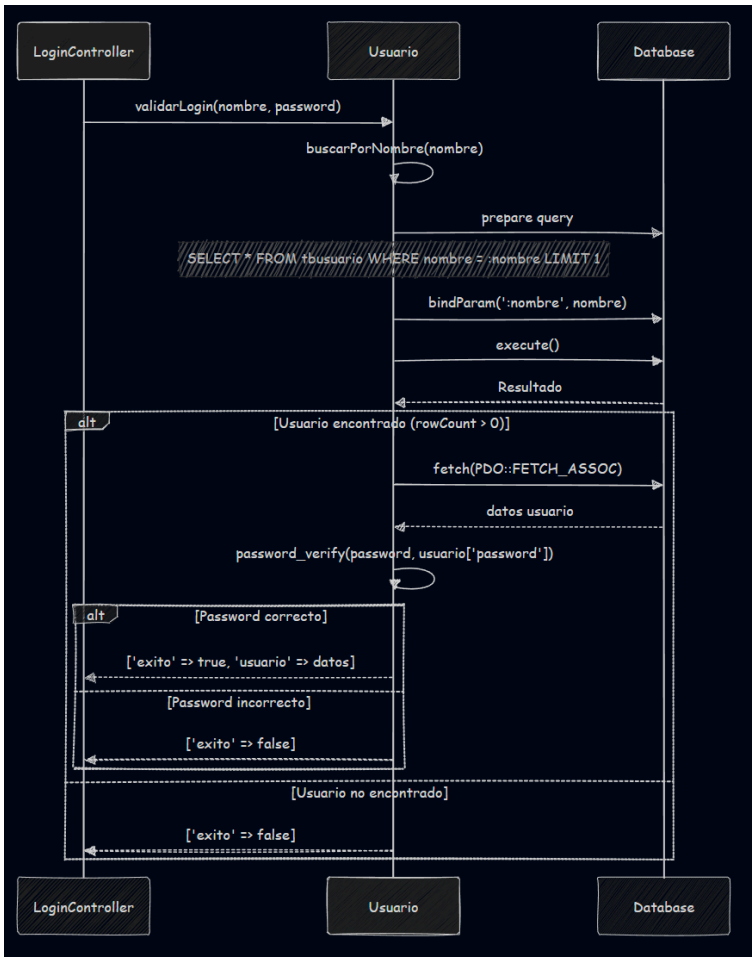
Cancelar Suscripción



Ingresar Script Ilimitado



Validar Login



3.2.3. Diagrama de Colaboración (vista de diseño)

3.2.4. Diagrama de Objetos

Diagrama de Objetos del CUS Autenticar Usuario

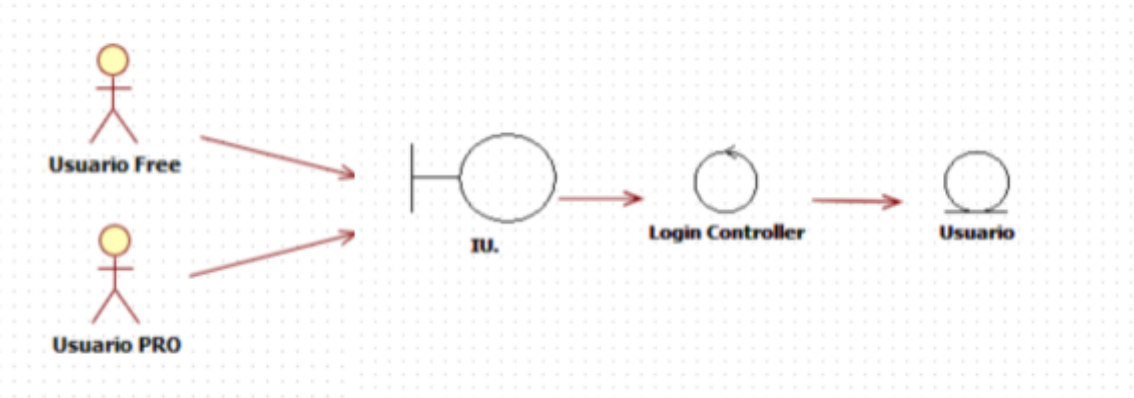


Diagrama de Objetos del CUS Elegir Suscripción FREE

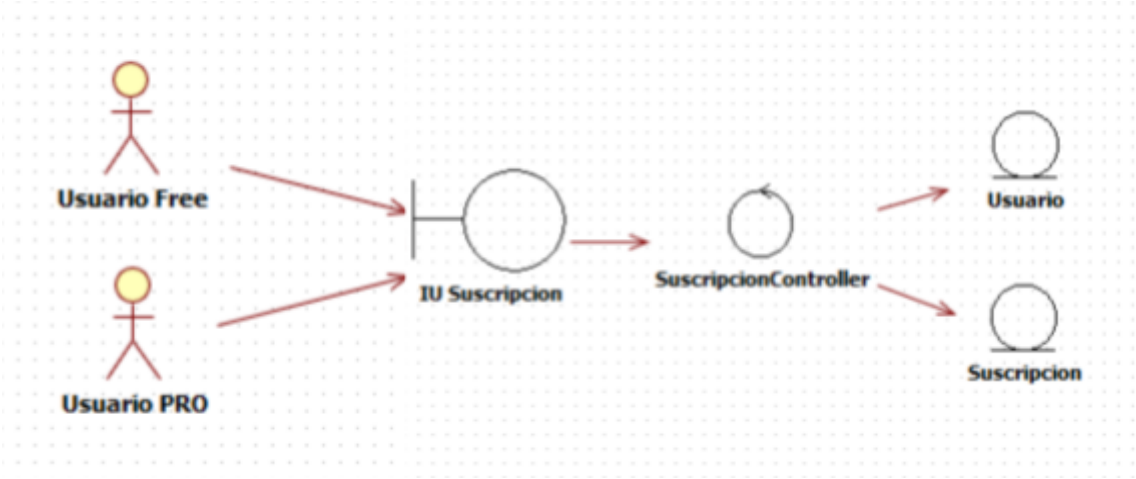
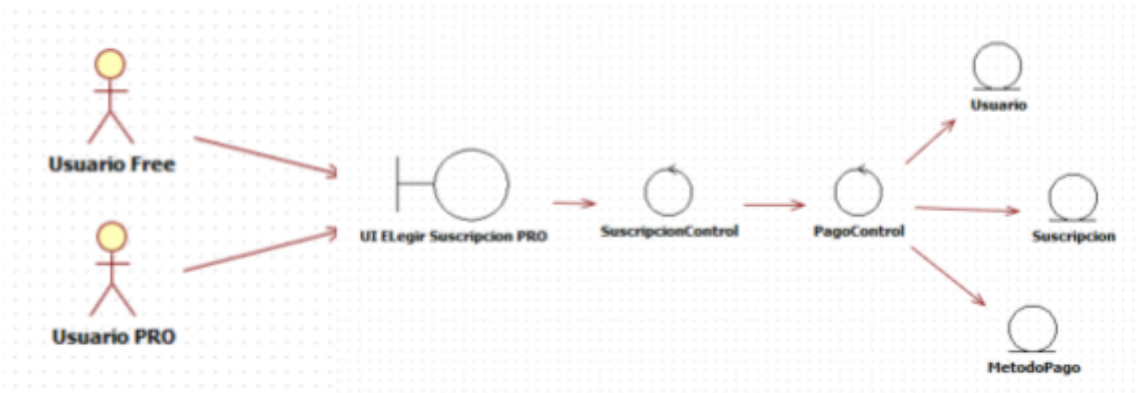
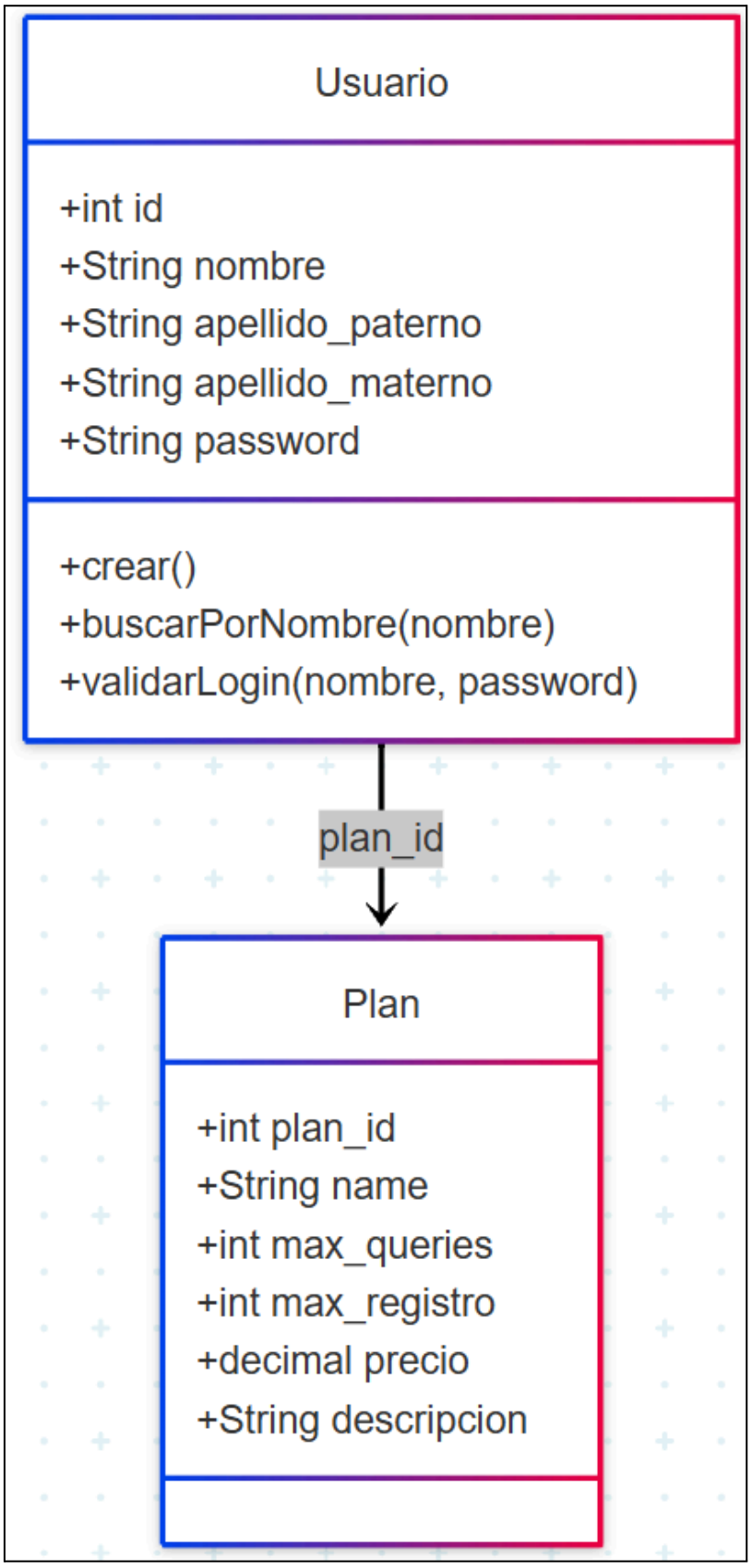


Diagrama de Objetos del CUS Elegir Suscripción PRO

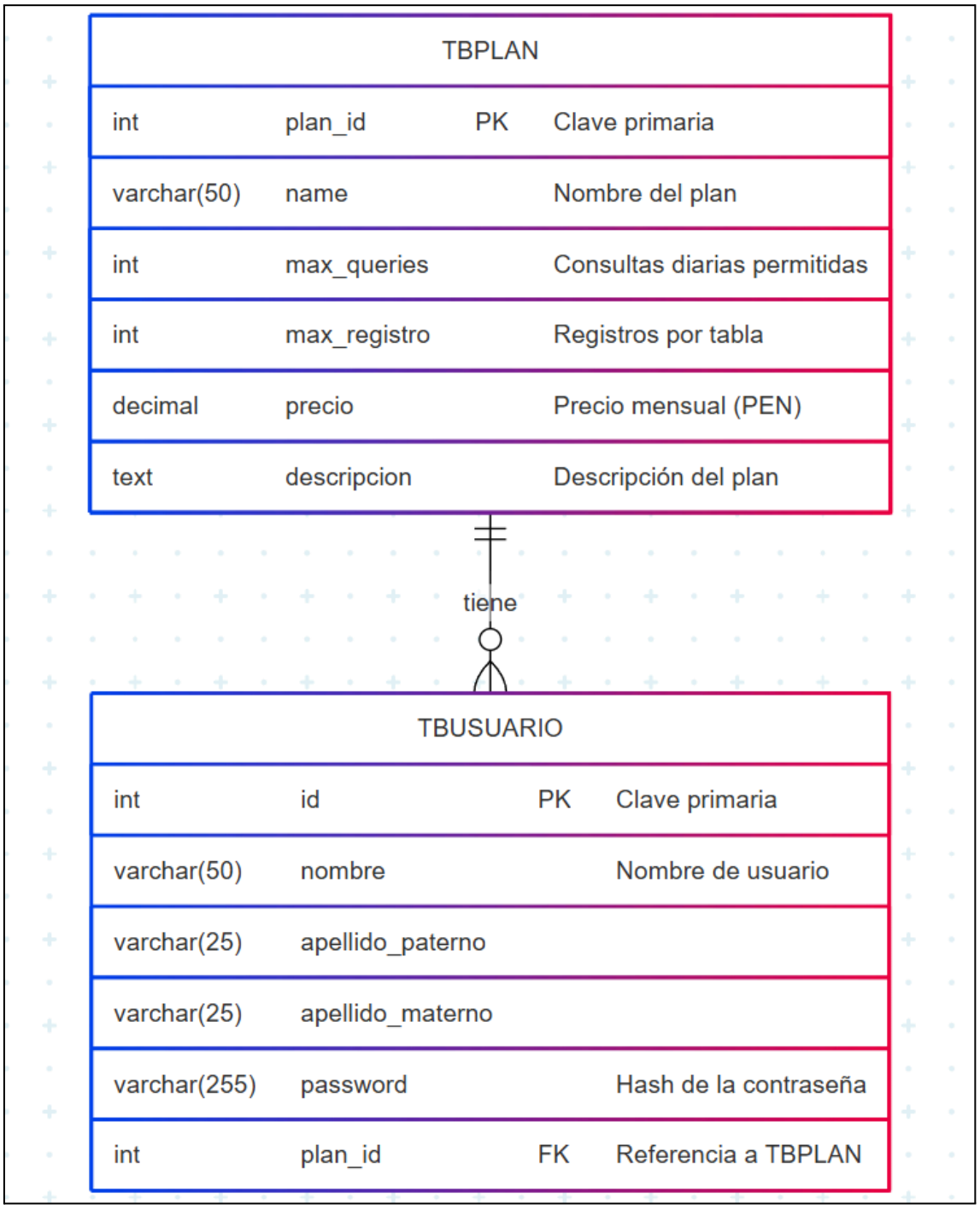


3.2.5. Diagrama de Clases

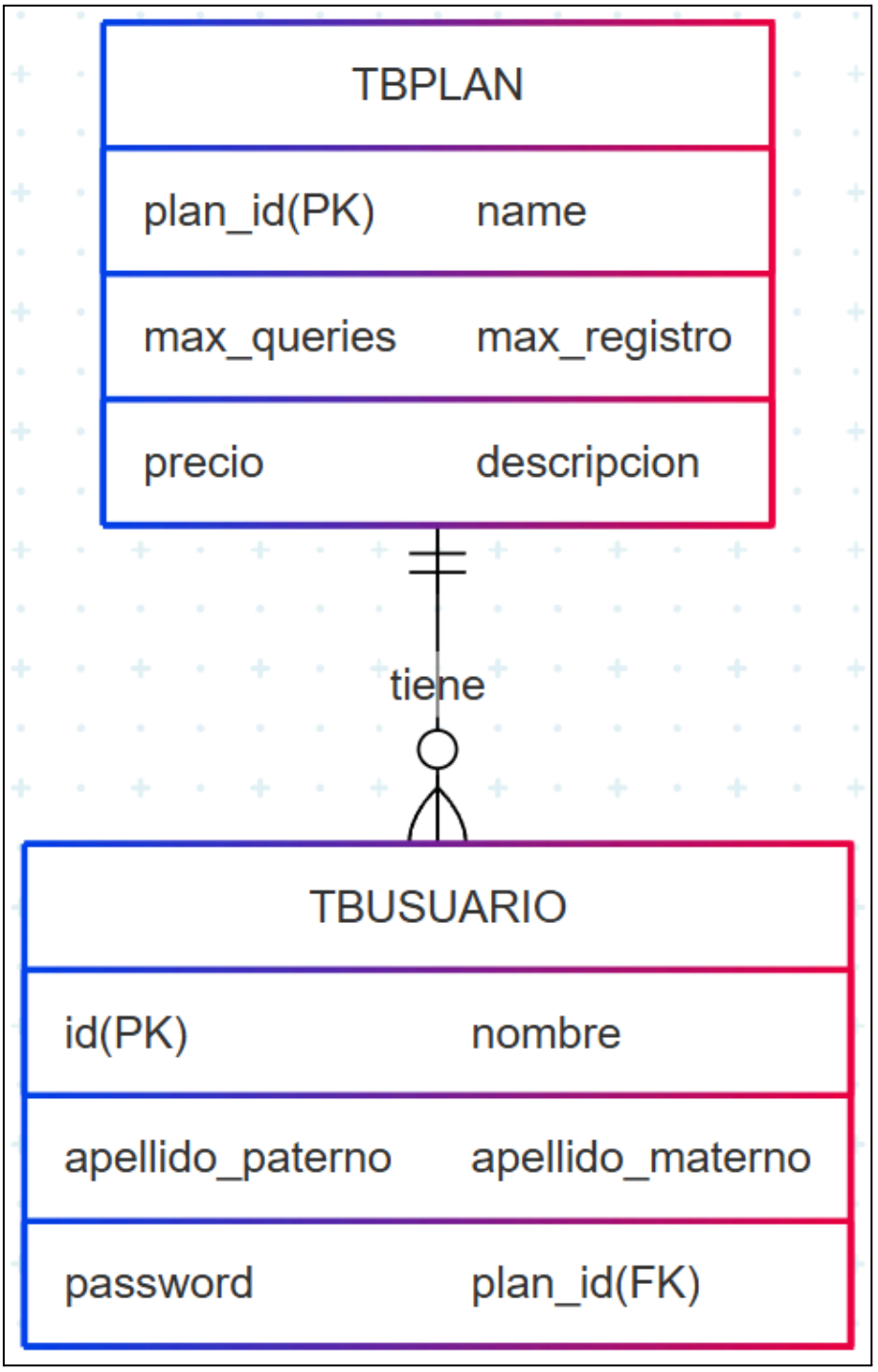


3.2.6. Diagrama de Base de datos (reacional o no relacional)

- Diagrama de Base de datos Físico

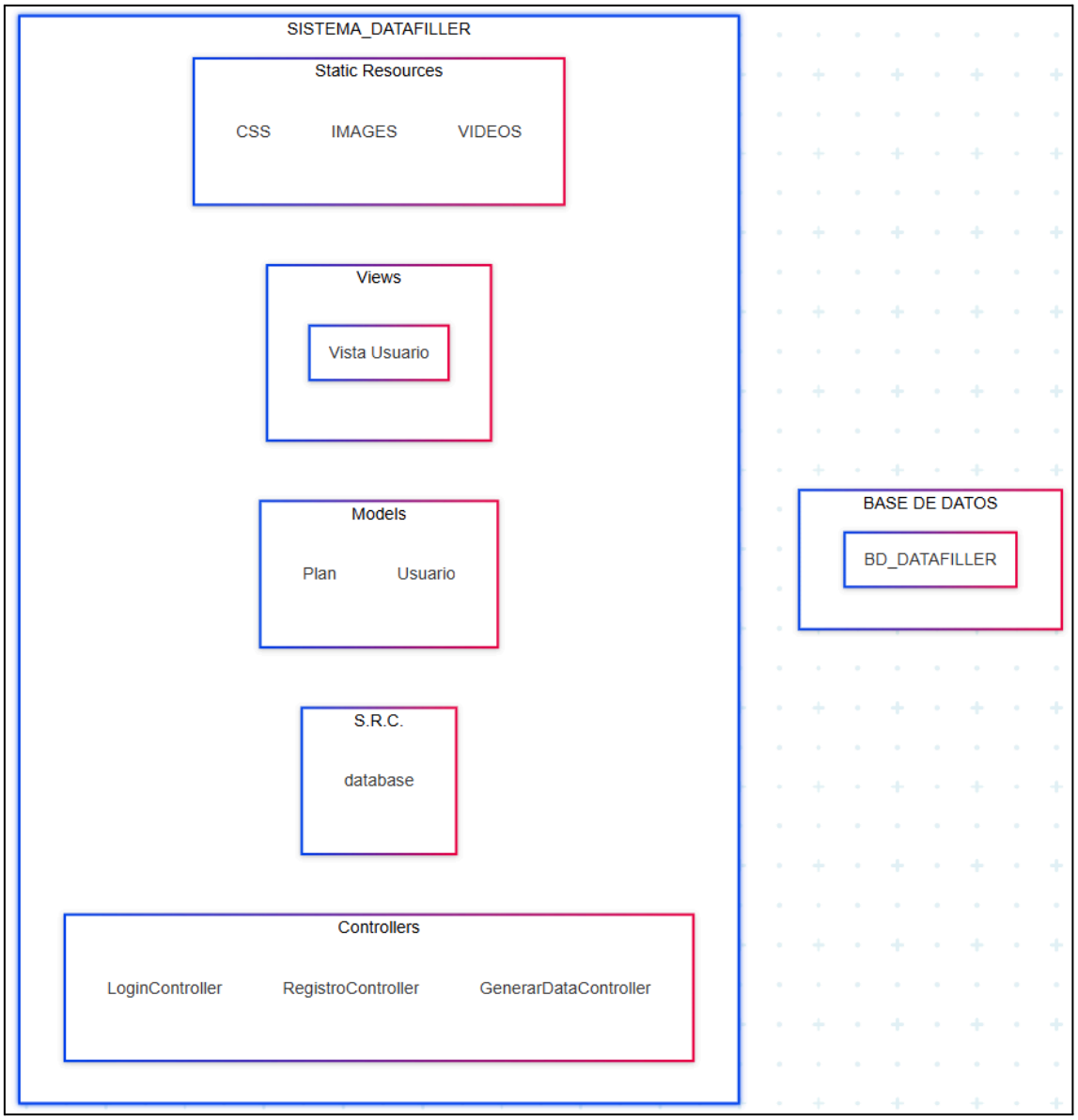


- Diagrama de Base de datos Lógico

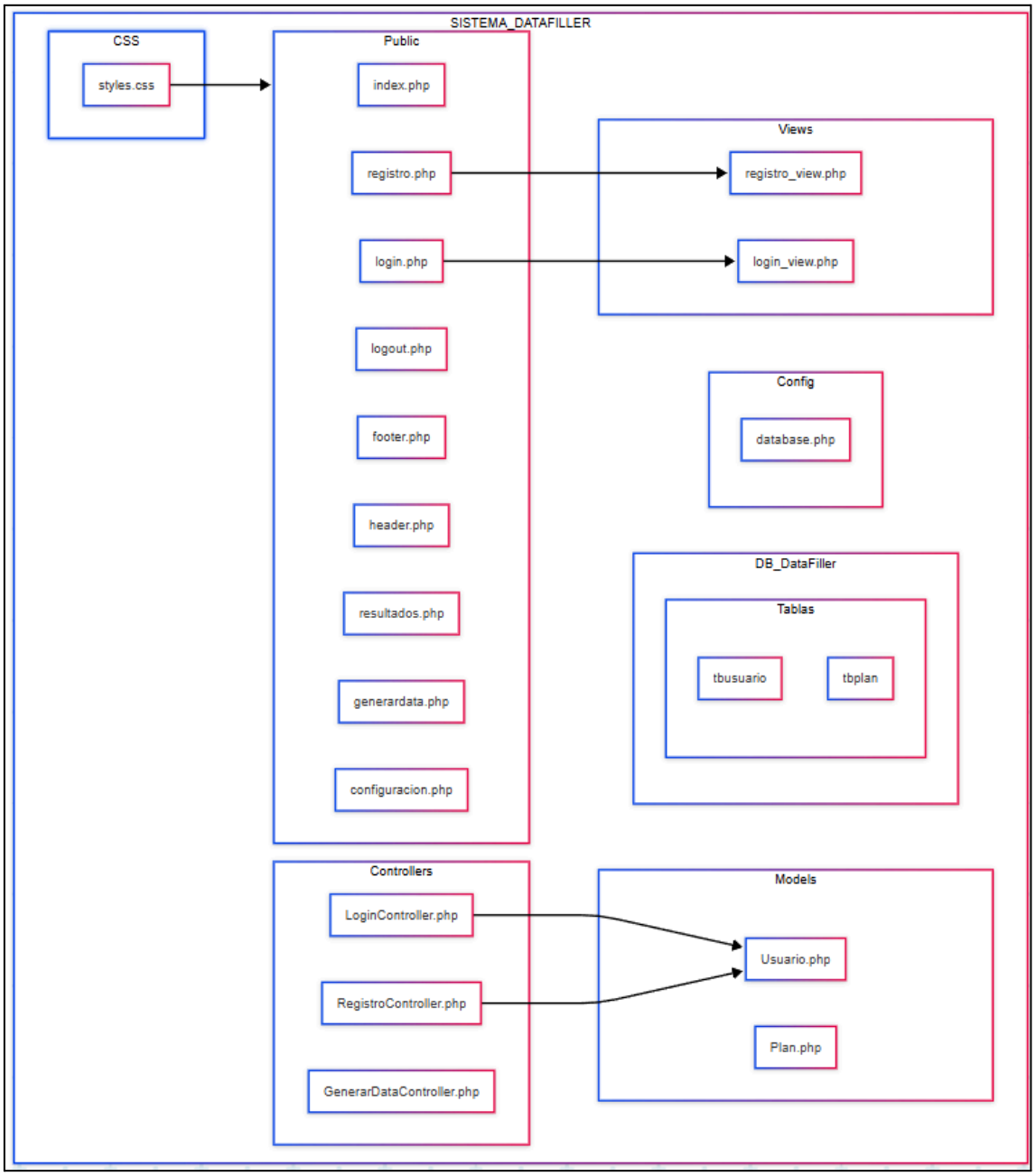


3.3. Vista de Implementación (vista de desarrollo)

3.3.1. Diagrama de arquitectura software (paquetes)



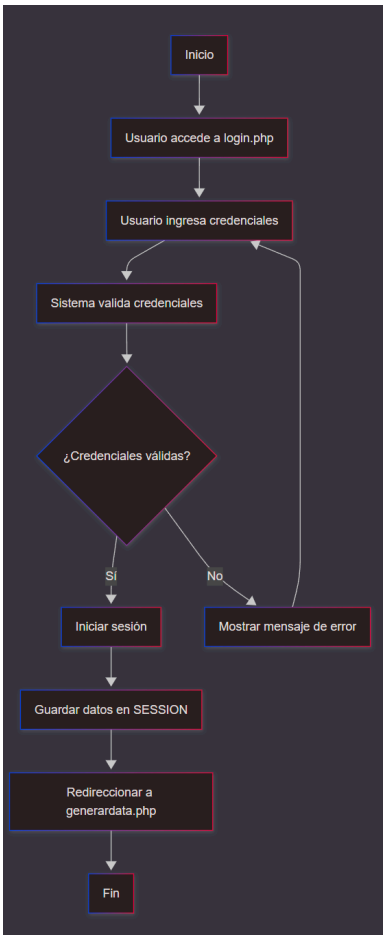
3.3.2. Diagrama de arquitectura del sistema (Diagrama de componentes)



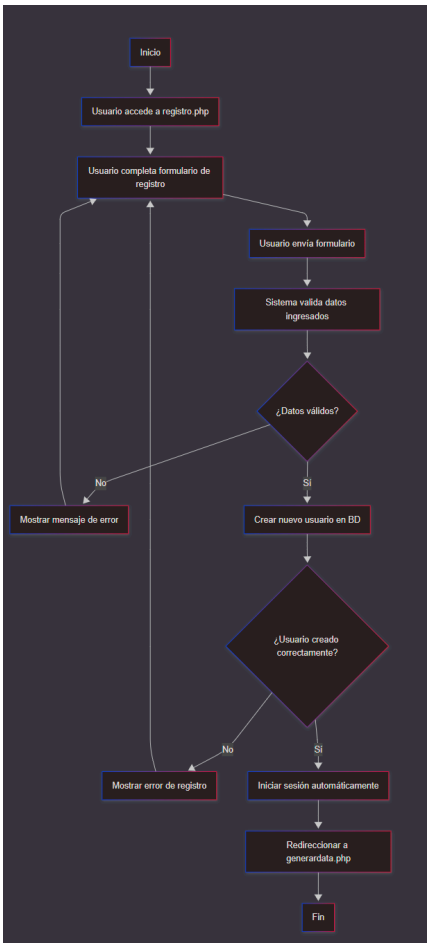
3.4. Vista de procesos

3.4.1. Diagrama de Procesos del sistema (diagrama de actividad)

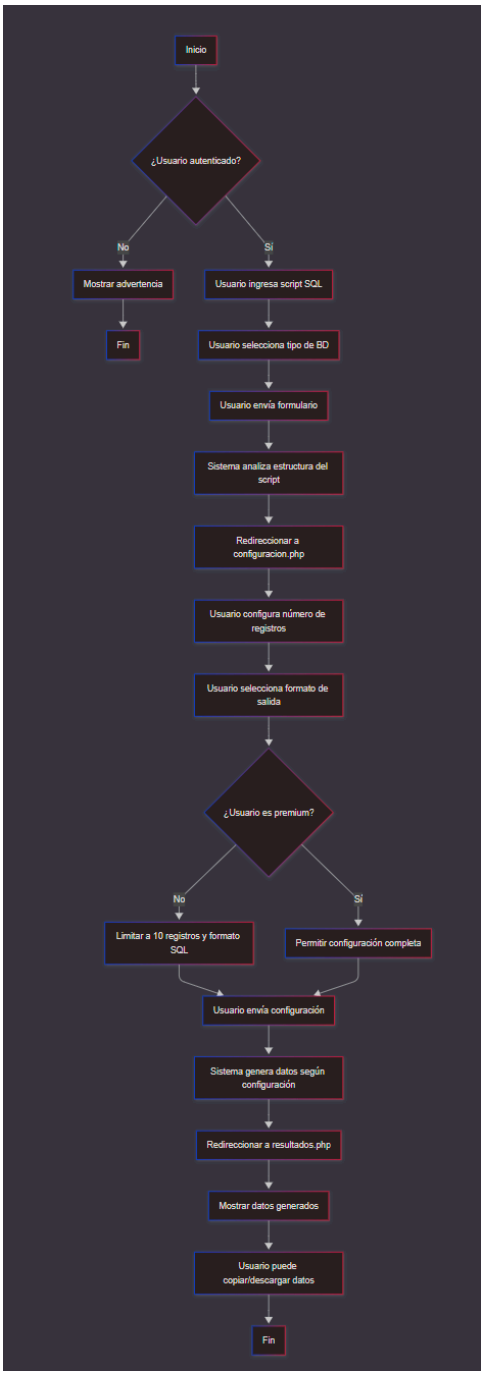
Proceso de Login



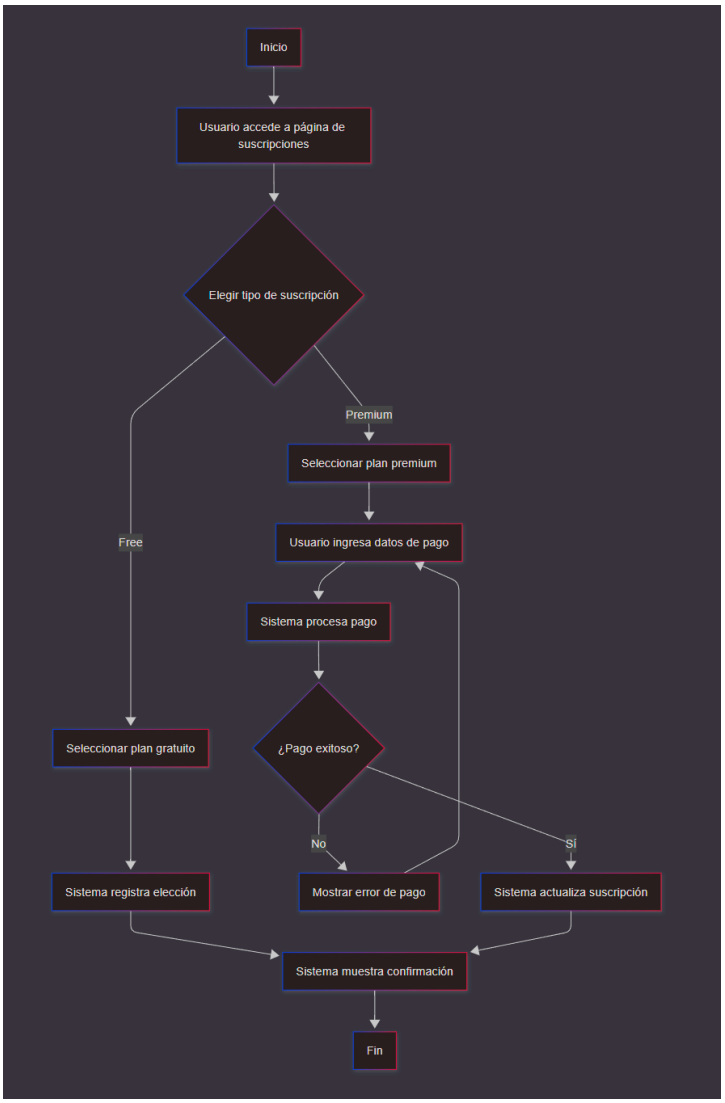
Proceso de Registro



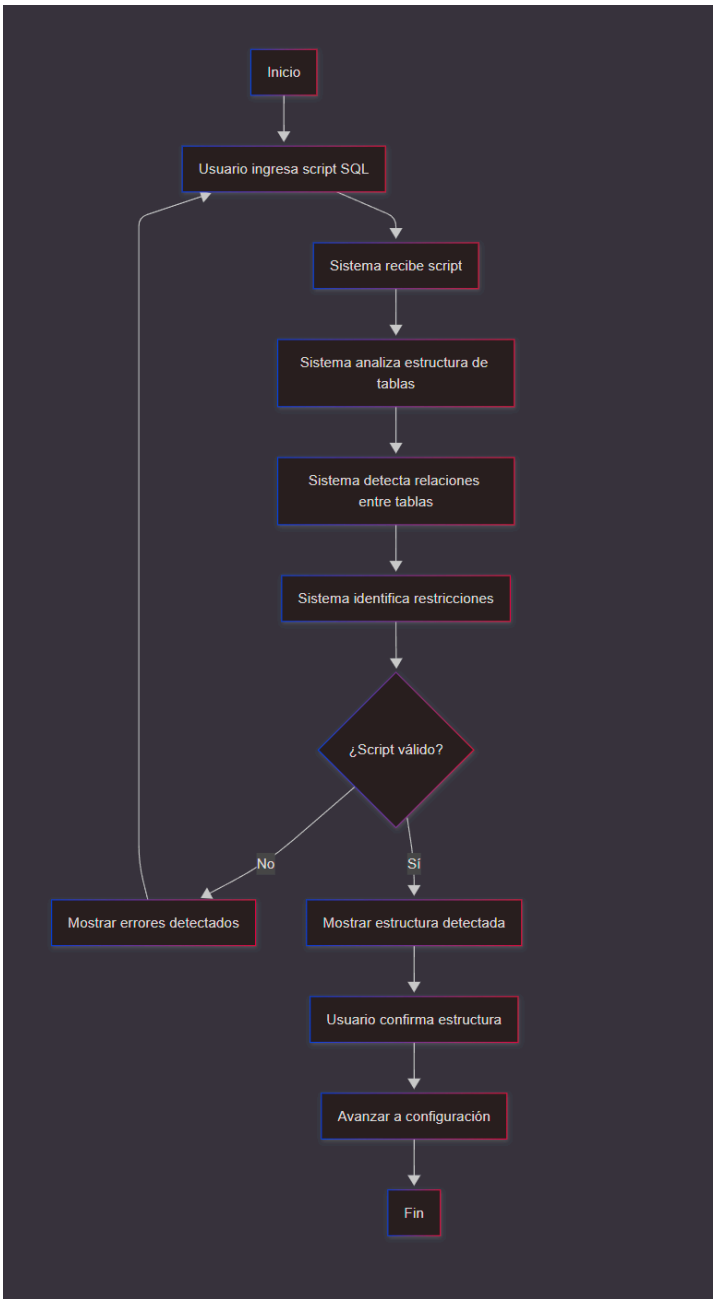
Generación de Datos



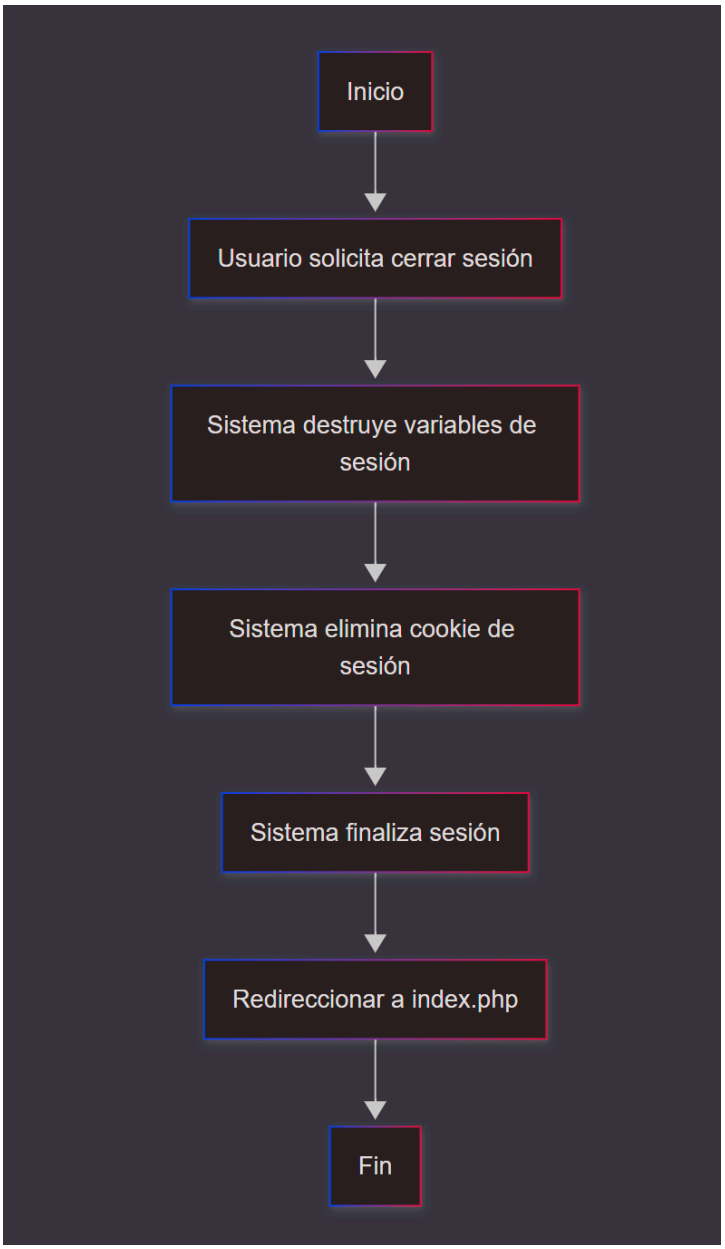
Gestión de Suscripción



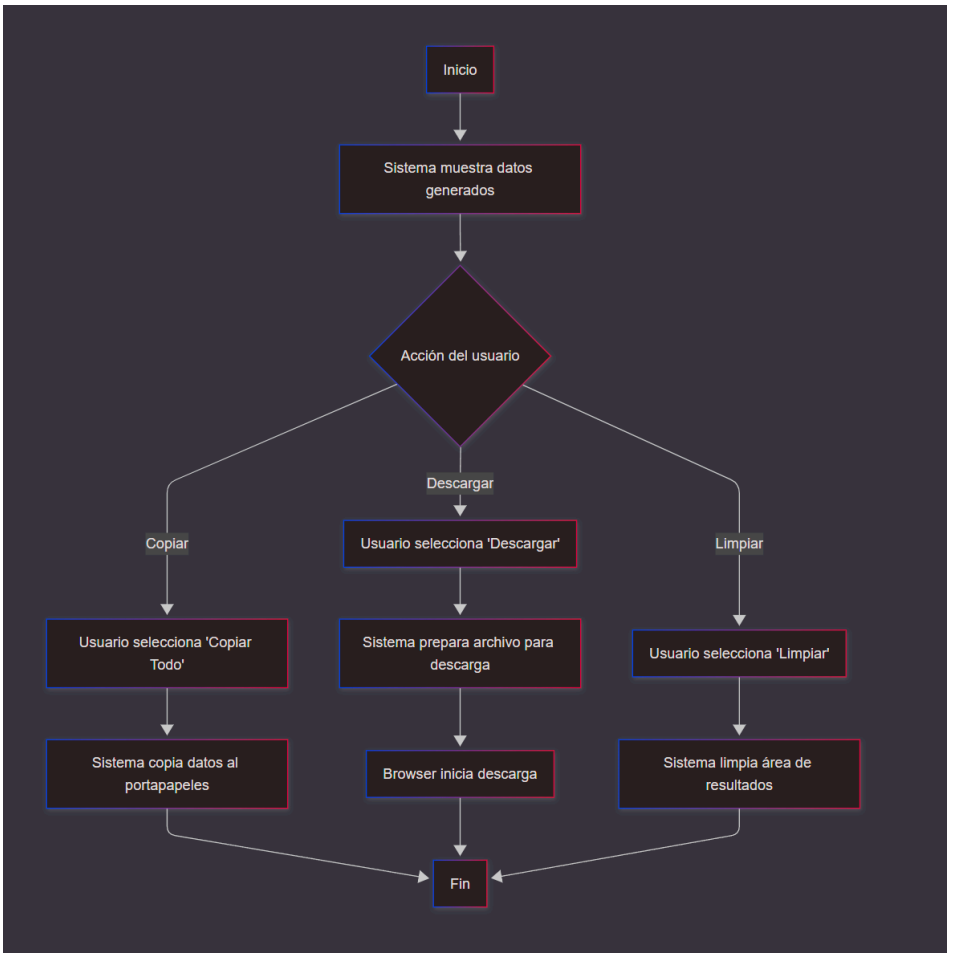
Análisis de Script SQL



Cierre de Sesión

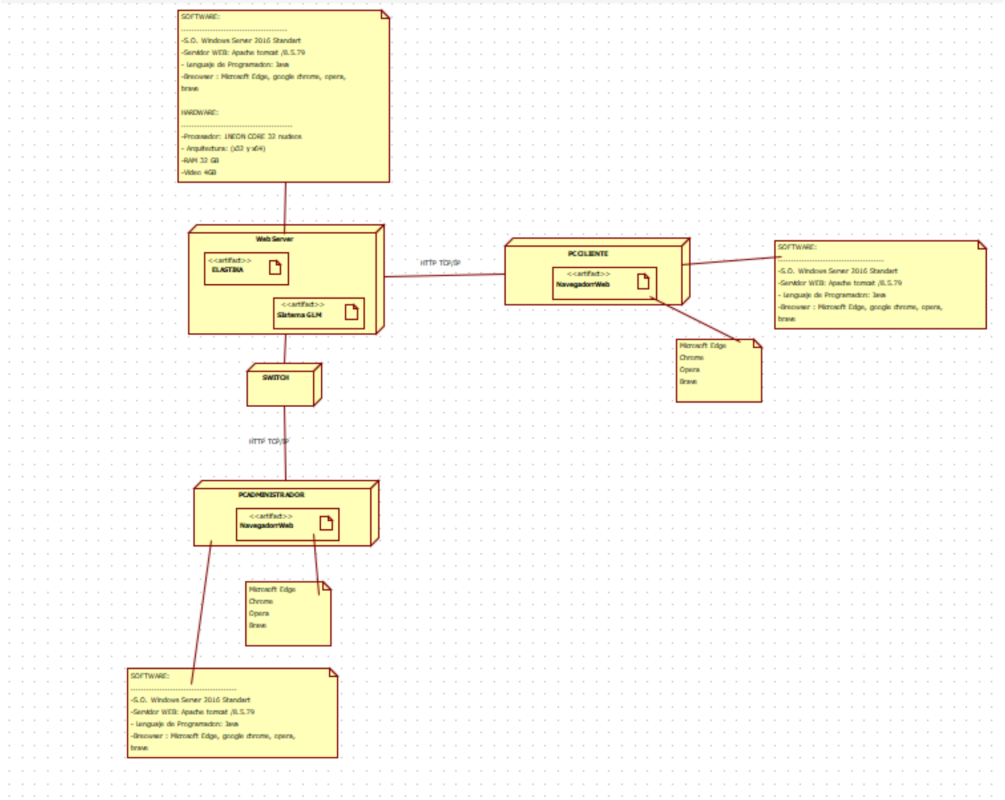


Gestión de Resultados



3.5. Vista de Despliegue (vista física)

3.5.1. Diagrama de despliegue



4. ATRIBUTOS DE CALIDAD DEL SOFTWARE

Escenario de Funcionalidad

DataFiller se enfoca en proporcionar un conjunto completo de funcionalidades para la generación de datos de prueba. La funcionalidad se evalúa mediante:

1. ***Completitud**: El sistema debe ser capaz de analizar correctamente al menos el 95% de los scripts SQL y NoSQL estándar presentados por los usuarios.*
2. ***Precisión**: Los datos generados deben cumplir con todas las restricciones definidas en el script original, incluyendo claves primarias, claves foráneas, restricciones de unicidad y reglas de validación.*
3. ***Adaptabilidad**: El sistema debe adaptarse a diferentes tipos de estructuras de bases de datos, desde simples tablas independientes hasta esquemas complejos con múltiples niveles de relaciones.*

4. ***Seguridad**: Toda la funcionalidad debe implementarse con estrictas medidas de seguridad, asegurando que los datos generados sean sintéticos y no comprometan información sensible.*

Escenario de Usabilidad

La usabilidad es un factor crítico para DataFiller, ya que está diseñado para ser utilizado por profesionales con diversos niveles de conocimiento técnico:

1. ***Facilidad de aprendizaje**: Un usuario nuevo debe ser capaz de generar su primer conjunto de datos de prueba en menos de 10 minutos después de registrarse, sin necesidad de consultar documentación extensa.*
2. ***Eficiencia de uso**: El sistema debe permitir que los usuarios frecuentes generen datos de prueba en menos de 5 pasos y en un tiempo promedio inferior a 2 minutos (excluyendo el tiempo de análisis para scripts muy complejos).*
3. ***Memorabilidad**: Los usuarios que regresan después de un período sin utilizar el sistema deben poder recordar fácilmente cómo utilizarlo sin necesidad de reaprendizaje.*
4. ***Tasa de errores**: La interfaz debe prevenir errores comunes del usuario, como configuraciones incompatibles o restricciones imposibles de cumplir, proporcionando alertas claras y sugerencias de corrección.*
5. ***Satisfacción**: El sistema debe mantener una calificación de satisfacción del usuario superior al 80% según las encuestas de retroalimentación.*

Escenario de confiabilidad

La confiabilidad es esencial para garantizar que DataFiller sea una herramienta consistente y fiable para el desarrollo y pruebas de software:

1. ***Disponibilidad**: El sistema debe mantener un tiempo de actividad de al menos 99.5% durante horas laborables (8 AM - 8 PM, de lunes a viernes).*

2. ***Fiabilidad de resultados****: Los datos generados deben ser consistentes con las especificaciones y restricciones en el 100% de los casos, sin violar reglas de integridad.*
3. ***Recuperación ante fallos****: En caso de interrupciones durante el proceso de generación, el sistema debe ser capaz de recuperarse y reanudar la operación sin pérdida de datos o configuraciones del usuario.*
4. ***Protección de datos****: El sistema debe garantizar que ningún dato ingresado por los usuarios sea accesible por otros usuarios, manteniendo una estricta separación de información.*
5. ***Integridad de transacciones****: Todas las operaciones críticas, como pagos y generación de datos, deben completarse íntegramente o no realizarse en absoluto (transacciones atómicas).*

Escenario de rendimiento

El rendimiento es crucial para la experiencia del usuario, especialmente cuando se trabaja con grandes volúmenes de datos:

1. **Tiempo de respuesta**: El análisis inicial de scripts debe completarse en menos de 5 segundos para scripts estándar (hasta 20 tablas).
2. **Escalabilidad**: La generación de datos debe mantener un rendimiento aceptable incluso para grandes volúmenes, generando al menos 1000 registros por segundo para estructuras simples.
3. **Concurrencia**: El sistema debe soportar al menos 100 usuarios concurrentes realizando tareas de generación de datos sin degradación significativa del rendimiento.
4. **Eficiencia de recursos**: El consumo de memoria y CPU debe optimizarse para minimizar costos de infraestructura, utilizando no más del 70% de los recursos del servidor durante picos de actividad.
5. **Tiempo de descarga**: Los datos generados deben prepararse para descarga en un formato comprimido que optimice la velocidad de transferencia, con un tiempo máximo de 10 segundos para conjuntos de datos de hasta 10MB.

Escenario de mantenibilidad

La mantenibilidad asegura que el sistema pueda evolucionar y adaptarse a nuevos requisitos:

1. ***Modularidad**: El sistema debe estar organizado en módulos independientes con interfaces bien definidas, permitiendo modificar un componente sin afectar a otros.*
2. ***Extensibilidad**: Debe ser posible añadir soporte para nuevos tipos de bases de datos o formatos de exportación sin reescribir componentes existentes.*
3. ***Facilidad de pruebas**: Cada componente debe contar con pruebas unitarias y de integración que cubran al menos el 80% del código.*
4. ***Documentación**: El código debe estar documentado siguiendo estándares de la industria, facilitando su comprensión por nuevos desarrolladores.*
5. ***Simplicidad**: Se debe priorizar diseños simples y directos sobre soluciones complejas cuando ambas satisfacen los requisitos funcionales.*

Otros Escenarios

1. *Protección de datos: Toda la información transmitida entre el cliente y el servidor debe cifrarse mediante HTTPS.*
2. *Autenticación segura: El sistema debe implementar mecanismos robustos de autenticación, incluyendo contraseñas seguras y protección contra ataques de fuerza bruta.*
3. *Autorización: El acceso a funcionalidades debe estar estrictamente controlado según el tipo de plan del usuario.*
4. *Cumplimiento normativo: La plataforma debe cumplir con regulaciones de protección de datos como GDPR*