UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERÍA

Escuela Profesional de Ingeniería de Sistemas

Plataforma Avanzada para la Generación Automática de Diagramas UML para la empresa Tech Solutions

Curso: Patrones de Software

Docente: Mag. Patrick Cuadrados

Integrantes:

- Alexis Jeanpierre Martínez Vargas (2019063638)
- Juan José David Pérez Vizcarra (2019063636)
- Jhon Thomas Ticona Chambi (2018062232)

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

Plataforma Avanzada para la Generación Automática de Diagramas UML para la empresa Tech Solutions

Documento de Arquitectura de Software

Versión *1.0*

Presentado Por:

Ticona Chambi, Jhon

Documentador

2025

CONTROL DE VERSIONES					
Versión	Hecha por	Revisada por	Aprobada por	Fecha	Motivo
1.0	JTC	AMV, JTC	AMV	26/04/2025	Versión Inicial

INDICE GENERAL

1. INTRODUCCIÓN

1.1. Propósito

El propósito de este documento es presentar la arquitectura del sistema "Plataforma Avanzada para la Generación Automática de Diagramas UML" para la empresa Tech Solutions, utilizando el modelo de vistas 4+1. Este enfoque permite describir la arquitectura desde diferentes perspectivas, asegurando que cumpla con los requisitos funcionales y no funcionales identificados en el documento SRS.

1.2. Alcance

Este documento se centrará en:

- La vista lógica del sistema, detallando los componentes principales (módulo de generación UML, módulo de colaboración).
- Aspectos clave de las demás vistas, como la vista de procesos para la edición en tiempo real y la vista física para la infraestructura básica.
- Se omitirán detalles de bajo nivel, como configuraciones específicas de servidores o implementaciones de plugins.

1.3. Definición, siglas y abreviaturas

- UML Unified Modeling Language: Lenguaje estándar para modelado visual de sistemas software.
- JSON/YAML Formatos de intercambio de datos utilizados para importar/exportar modelos de diagramas.
- Git Sistema de control de versiones integrado para generar diagramas basados en cambios de código.
- RUP Rational Unified Process: Metodología utilizada para el desarrollo del sistema.
- RF Requerimiento Funcional (RF-01 para selección de diagramas).
- Rnf Requerimiento No Funcional (Rnf01 para rendimiento).
- CUS Caso de Uso del Sistema ("Generar UML").
- BR Regla de Negocio (RN01 para registro de usuarios).

1.4. Organización del documento

2. OBJETIVOS Y RESTRICCIONES ARQUITECTONICAS

2.1. Priorización de requerimientos

2.1.1. Requerimientos Funcionales

Nombre	Código	Descripción	Sistema	Prioridad
RF-01	Selección de Opciones	Elegir el tipo de diagrama UML a generar: Clases, Secuencia, Casos de Uso, Componentes.	Plataforma Web	Alta
RF-02		Soporte para múltiples lenguajes de programación (C#, Java, Python, etc.).	Plataforma Web	Alta
RF-03	Generación de UML	Convertir automáticamente el código pegado en un diagrama UML con vista previa en tiempo real.	Plataforma Web	Alta
RF-04	Edición y Personalización	Permitir la edición manual del diagrama generado, añadiendo o eliminando nodos, relaciones o notas.	Plataforma Web	Alta
RF-05	Colaboración	Permitir compartir diagramas UML en tiempo real con otros usuarios para edición conjunta.	Plataforma Web	Media
RF-06	Exportación	Guardar diagramas en diferentes formatos: PNG, SVG, PDF o en código PlantUML/Mermaid.	Plataforma Web	Alta
RF-07	Historial y Versionado	Permitir la reversión a versiones anteriores del diagrama para recuperar cambios previos.	Plataforma Web	Media
RF-08	Comentarios y Notas	Posibilidad de añadir anotaciones o notas técnicas a los elementos del diagrama UML.	Plataforma Web	Baja

RF-09	**Validaciones**	Alertar si el código pegado tiene errores de sintaxis que impiden la generación del diagrama.	Plataforma Web	Alta
RF-10	**Seguridad y Acceso**	Implementar autenticación de usuarios con roles de acceso (Invitado, Usuario, Administrador).	Plataforma Web	Alta

2.1.2. Requerimientos No Funcionales – Atributos de Calidad

Nro. Rnf	Requerimientos no Funcional	Descripción del Requerimiento No Funcional
Rnf01	Rendimiento	El sitio web debe tener un tiempo de carga máximo de 2 segundos para garantizar una experiencia de usuario fluida.
		El sistema debe ser capaz de manejar hasta 1000 usuarios concurrentes sin degradación significativa del rendimiento.
Rnf02	Seguridad	El sitio web debe implementar medidas de seguridad, como cifrado SSL, para proteger los datos de los usuarios.
		Debe haber una política de contraseñas seguras que requiera contraseñas fuertes para los usuarios registrados.
Rnf03	Disponibilidad	El sitio web debe estar disponible las 24 horas del día, los 7 días de la semana, con un tiempo de inactividad planificado mínimo.
Rnf04	Usabilidad	El diseño y la interfaz de usuario del sitio web deben ser intuitivos y fáciles de usar para niños, con elementos visuales atractivos.
		El sitio debe ser accesible desde dispositivos móviles y tabletas, además de computadoras de escritorio.
Rnf05	Compatibilidad	El sitio web debe ser compatible con los principales navegadores web, como Chrome, Firefox, Edge y Safari.

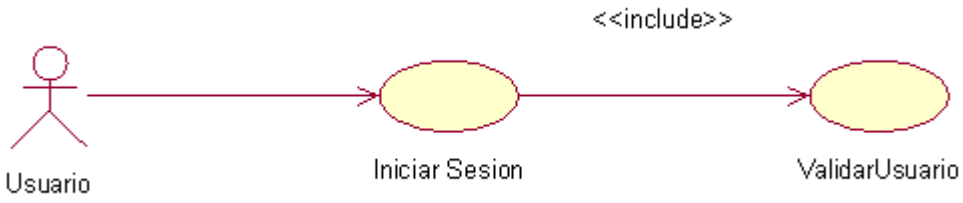
2.2. Restricciones

3. REPRESENTACIÓN DE LA ARQUITECTURA DEL SISTEMA

3.1. Vista de Caso de uso

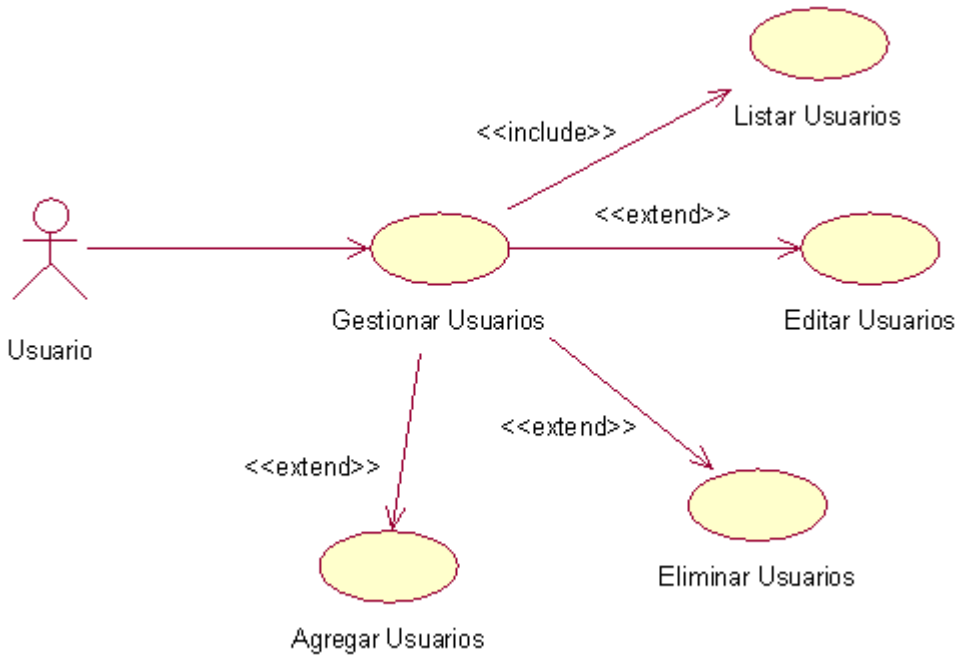
3.1.1. Diagramas de Casos de uso

Figura 1: Diagrama de Caso de Uso del Módulo Iniciar Sesión incluyendo la acción de validar Usuario



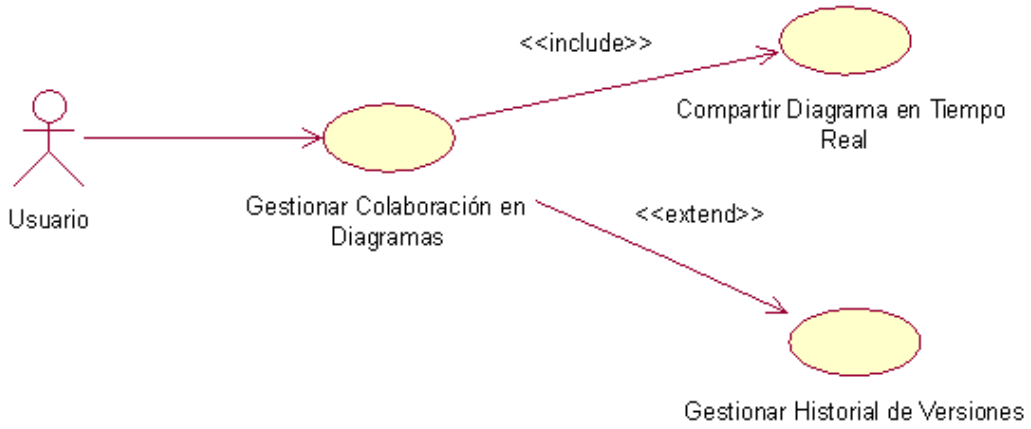
Fuente: Elaboración Propia

Figura 7: Diagrama de Caso de Uso del Modulo Gestionar Usuario que incluye la acción de Listar usuario y extensión de acciones de tipo Editar Usuarios, Agregar Usuarios y Eliminar Usuarios.



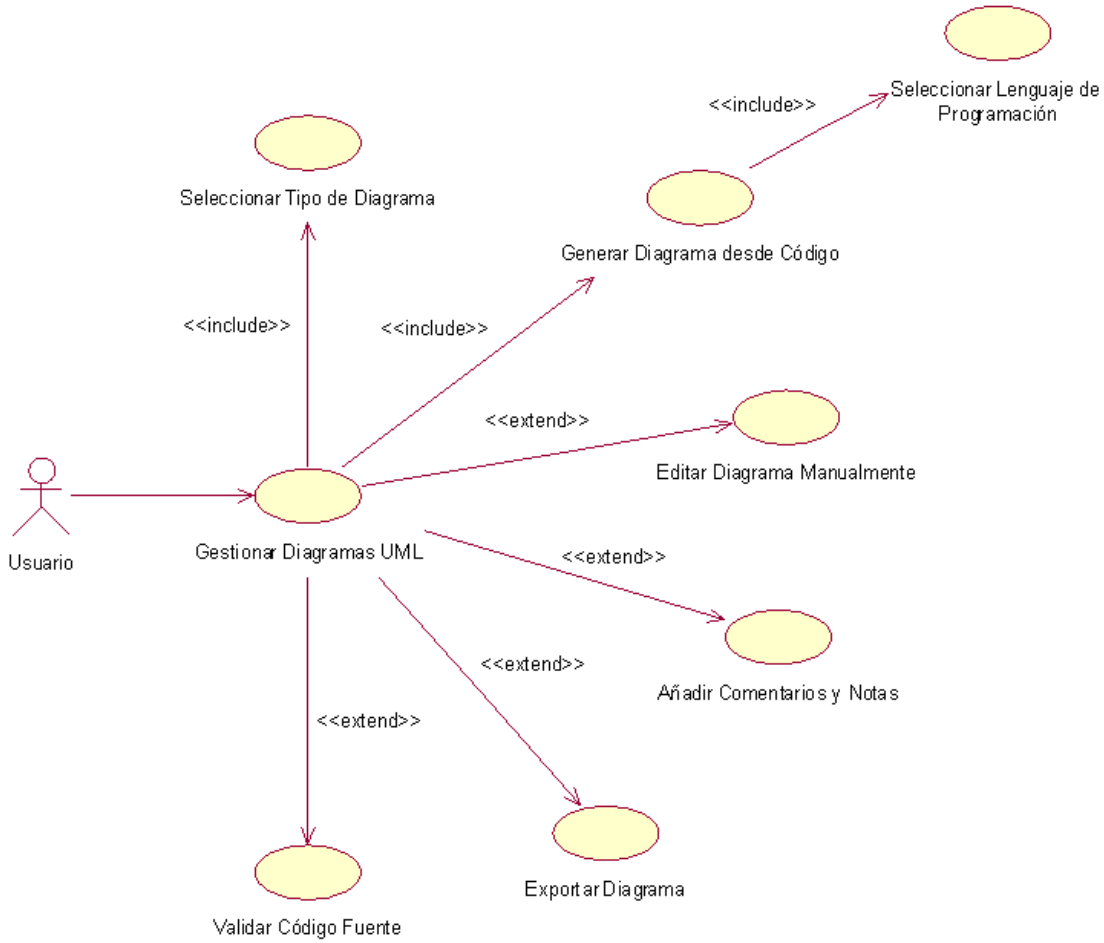
Fuente: Elaboración Propia

Figura 8: Diagrama de Caso de Uso del Módulo Gestionar Colaboración en Diagramas que incluye la acción de Compartir Diagrama en Tiempo Real y la extensión de Gestionar Historial de Versiones.



Fuente: Elaboración Propia

Figura 9: Diagrama de Caso de Uso del Módulo Gestionar Diagramas UML que incluye las acciones de Seleccionar Tipo de Diagrama y Generar Diagrama desde Código este a su vez incluye Seleccionar Lenguaje de Programación por otor lado en las extensiones son Editar Diagrama Manualmente, Añadir Comentarios, Exportar Diagrama y Validar Código Fuente.

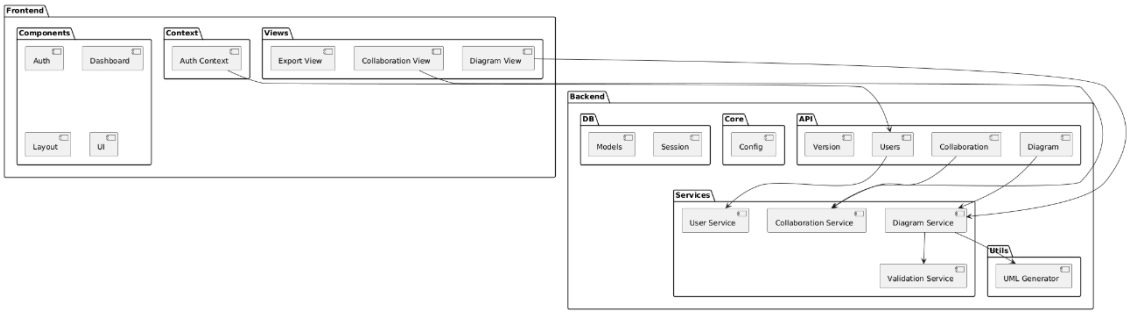


Fuente: Elaboración Propia

3.2. Vista Lógica

3.2.1. Diagrama de Subsistemas (paquetes)

Este diagrama organiza el sistema en módulos lógicos (paquetes) según su funcionalidad, mostrando las dependencias entre ellos para garantizar una arquitectura escalable y mantenible. Se estructura en tres capas principales:



3.2.2. Diagrama de Secuencia (vista de diseño)

Diagrama de secuencia de colaboración

Diagrama de secuencia de generación de UML

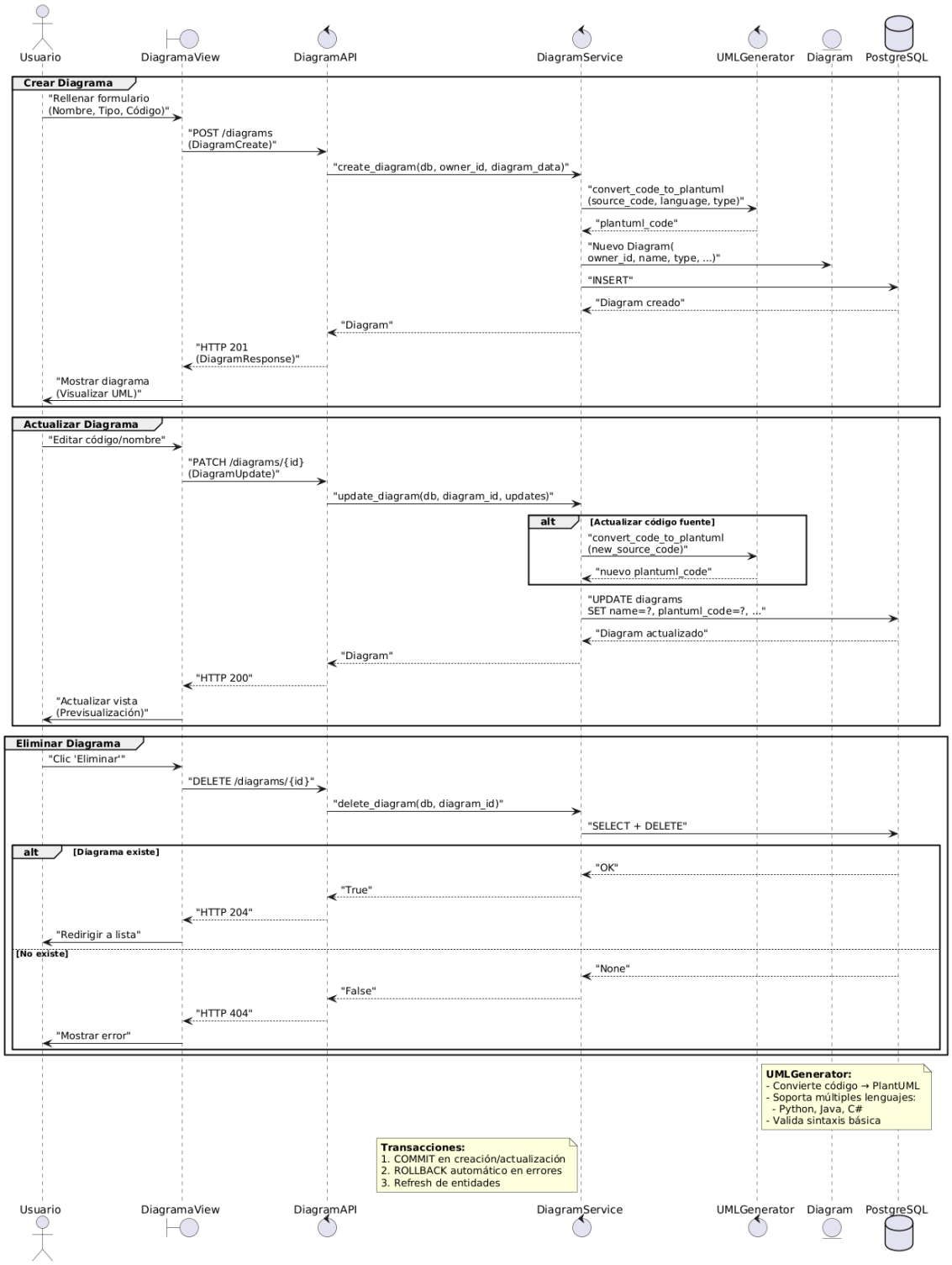


Diagrama de secuencia de autenticación

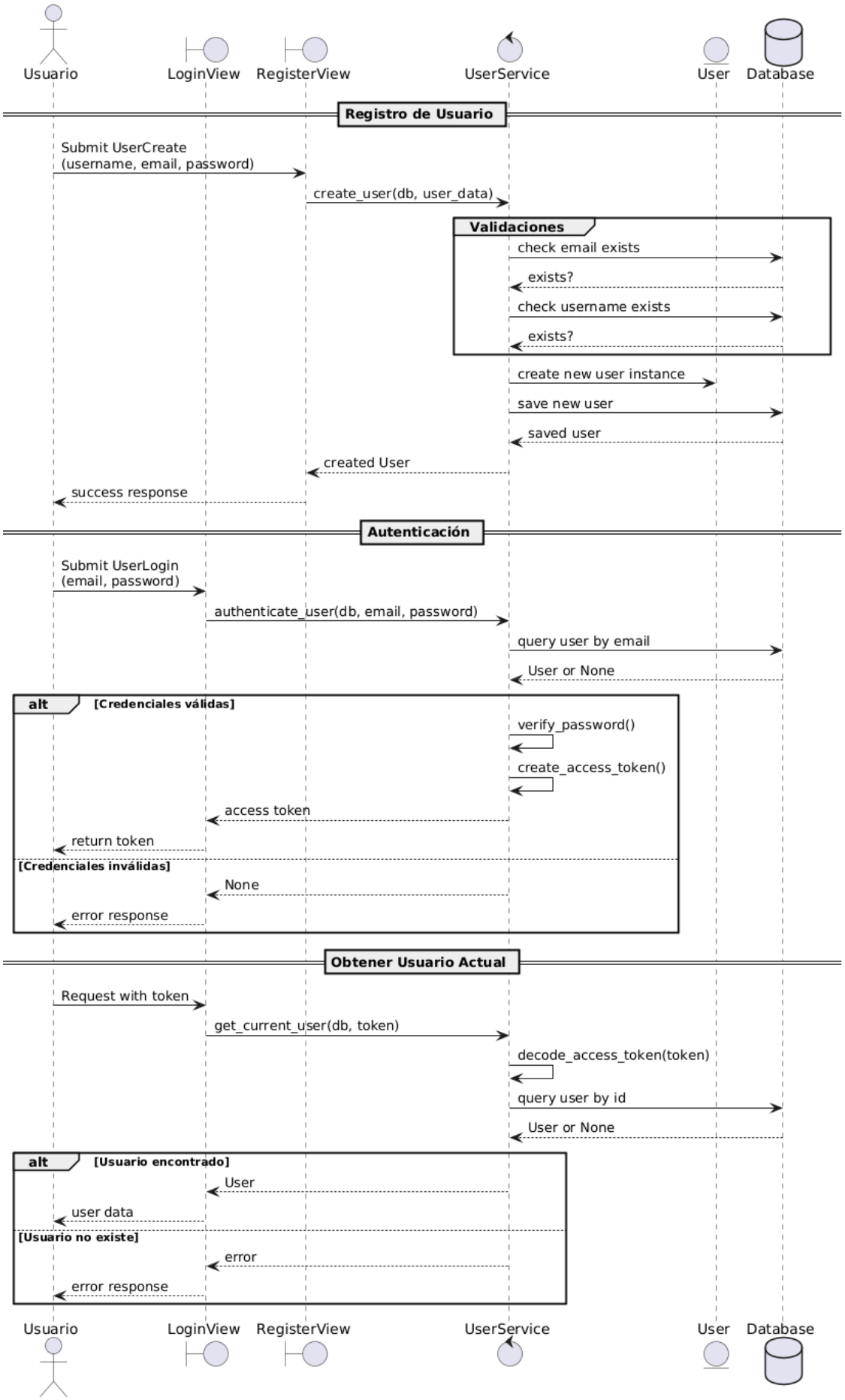

Usuario
LoginView
RegisterView
UserService
User
Database
Registro de Usuario
Submit UserCreate
(username, email, password)
create_user(db, user_data)
Validaciones
check email exists
exists?
check username exists
exists?
create new user instance
save new user
saved user
created User
success response
Autenticación
Submit UserLogin
(email, password)
authenticate_user(db, email, password)
query user by email
User or None
alt
[Credenciales válidas]
verify_password()
create_access_token()
access token
return token
[Credenciales inválidas]
None
error response
Obtener Usuario Actual
Request with token
get_current_user(db, token)
decode_access_token(token)
query user by id
User or None
alt
[Usuario encontrado]
User
user data
[Usuario no existe]
error
error response
Usuario
LoginView
RegisterView
UserService
User
Database

Diagrama de secuencia de versiona miento

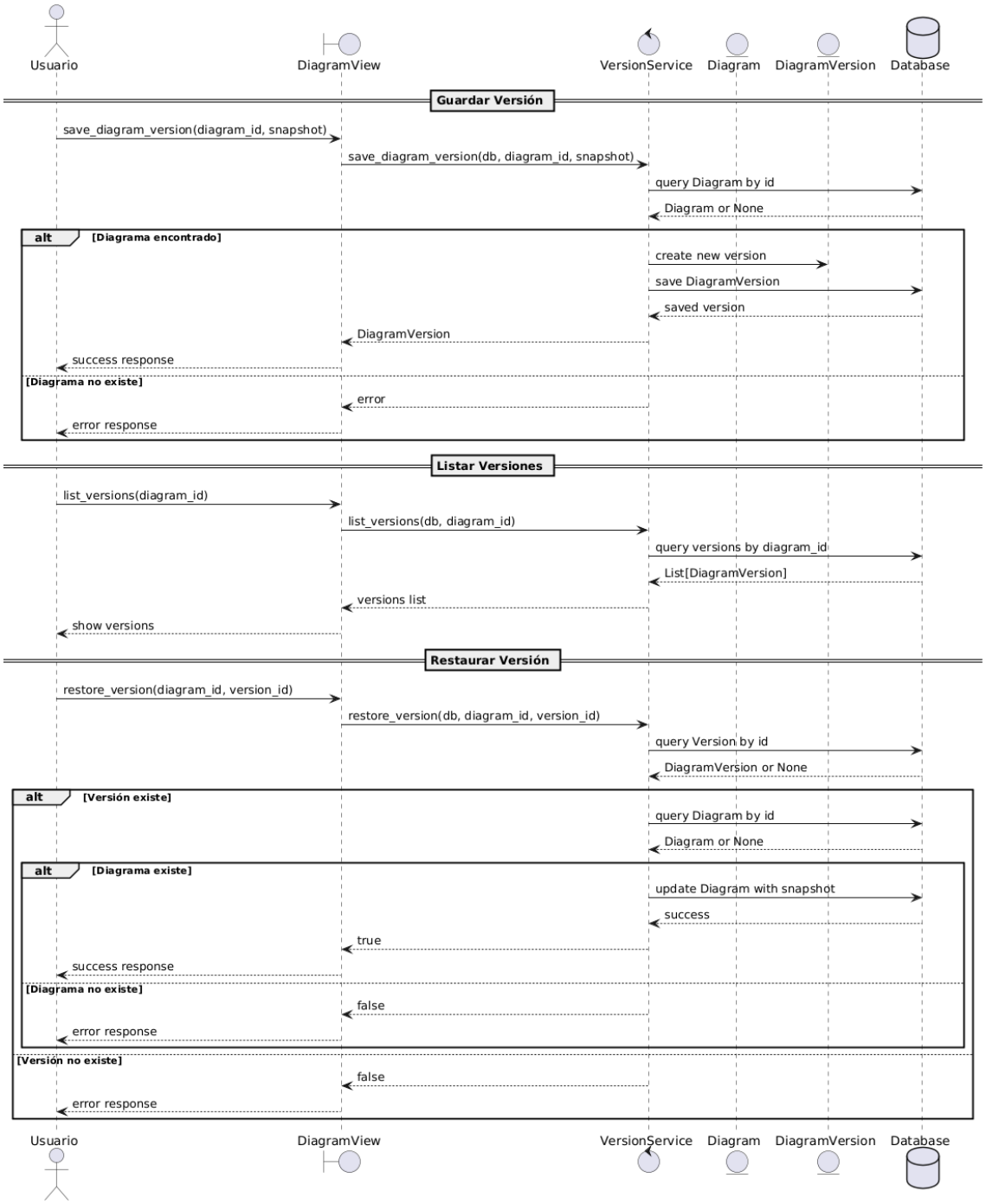
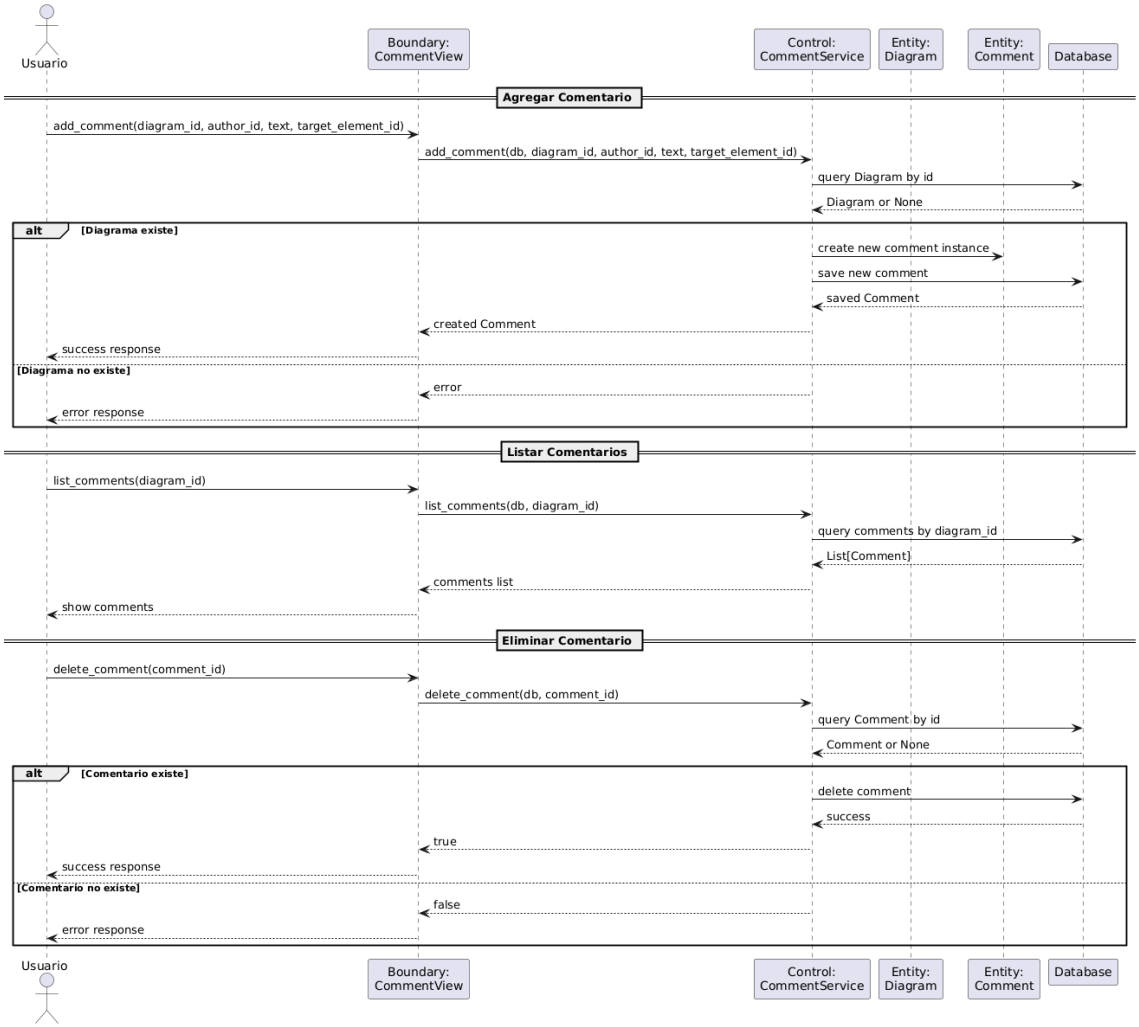
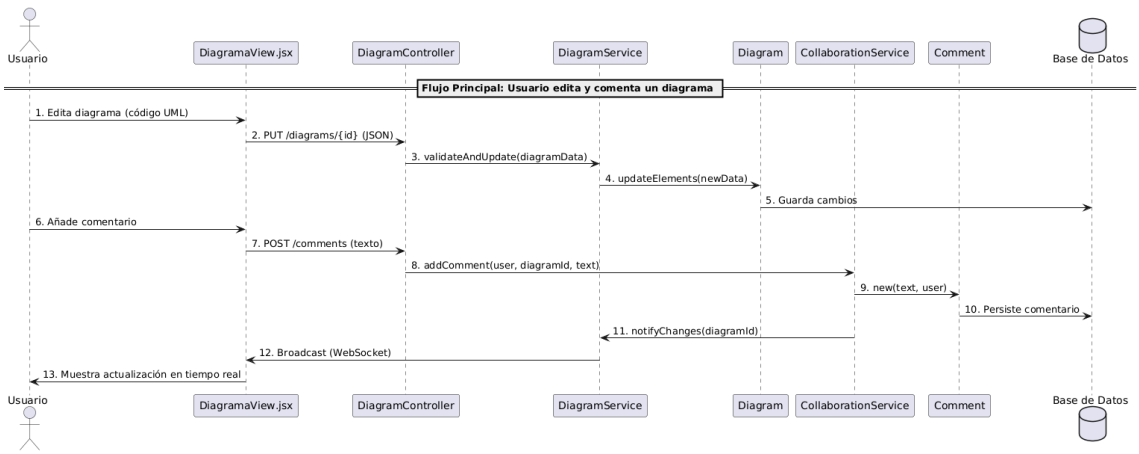


Diagrama de secuencia de Comentarios

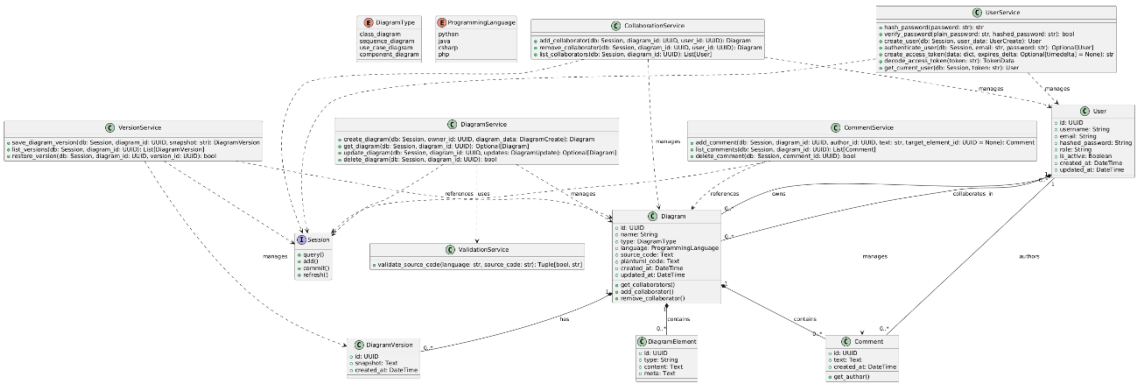


3.2.3. Diagrama de Colaboración (vista de diseño)



3.2.4. Diagrama de Clases

El Diagrama de Clases es una representación estática de la estructura del sistema, donde se modelan las clases principales, sus atributos, métodos y las relaciones entre ellas (herencia, asociación, composición, etc.). Este diagrama es fundamental para el diseño orientado a objetos, ya que define la arquitectura lógica del software antes de su implementación.



3.2.5. Diagrama de Base de datos (relacional o no relacional)

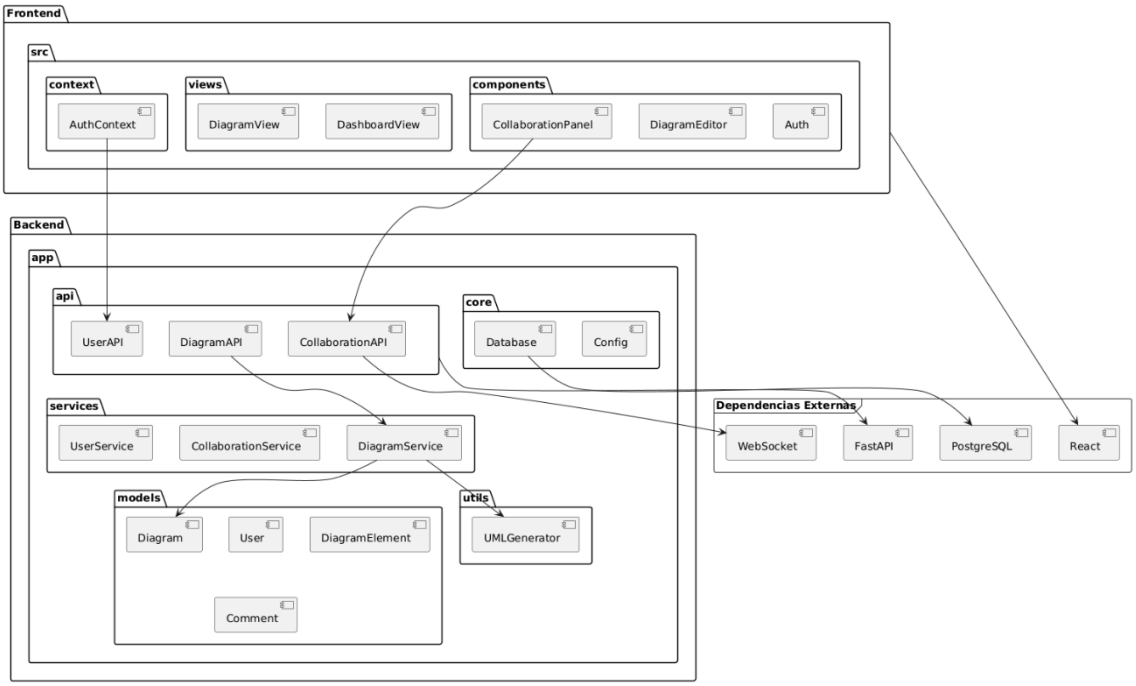
Representa el esquema de datos (relacional o NoSQL), incluyendo tablas como Diagram, DiagramVersion y Comment, con claves primarias/foráneas y restricciones, para guiar la implementación física de la persistencia.

3.3. Vista de Implementación (vista de desarrollo)

3.3.1. Diagrama de arquitectura software (paquetes)

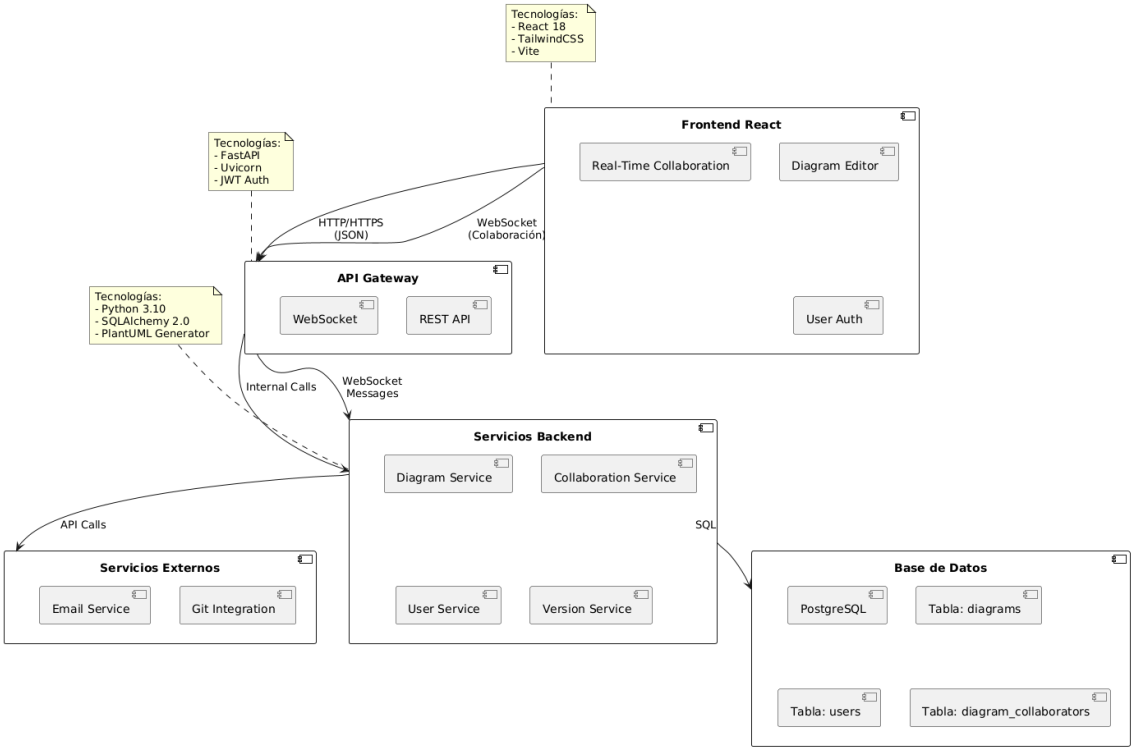
Este diagrama representa la estructura modular del sistema, organizando los componentes principales en paquetes lógicos (ej: autenticación, diagramación, comentarios) y sus dependencias. Muestra cómo

interactúan los módulos entre sí (ej: el paquete comentarios depende de diagramación), facilitando la comprensión de responsabilidades y acoplamiento. Incluye capas como presentación, lógica de negocio y persistencia, alineándose con patrones de diseño. Su propósito es guiar el desarrollo, mantenimiento y escalabilidad del código.



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

Este diagrama representa los componentes software reutilizables del sistema (ej: servicios, APIs, librerías) y sus interfaces de comunicación, mostrando cómo se integran entre sí y con sistemas externos (como bases de datos o autenticación de terceros). Cada componente se define por su función técnica (ej: CommentService, AuthAPI) y se conecta mediante interfaces estandarizadas (REST, gRPC, eventos). Su objetivo es visualizar la arquitectura técnica, facilitando el despliegue, la escalabilidad y el reemplazo de tecnologías.



3.4. Vista de procesos

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

Diagrama de actividades - colaboración

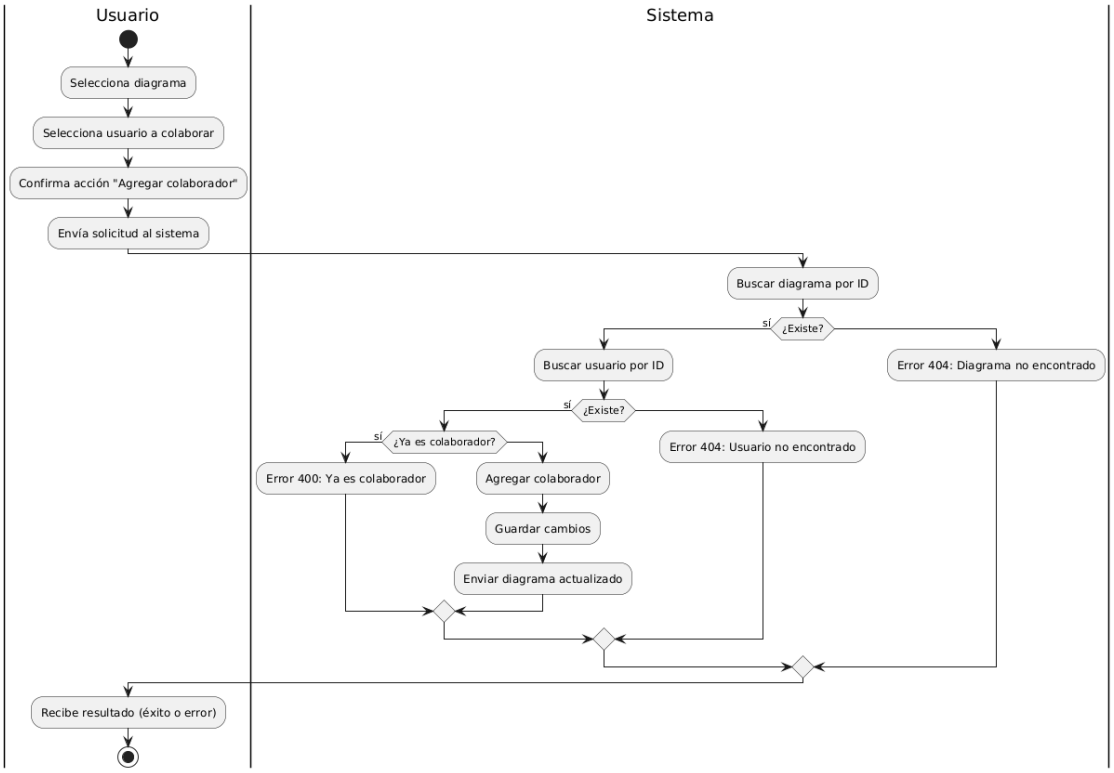


Diagrama de actividades – Generar diagrama UML

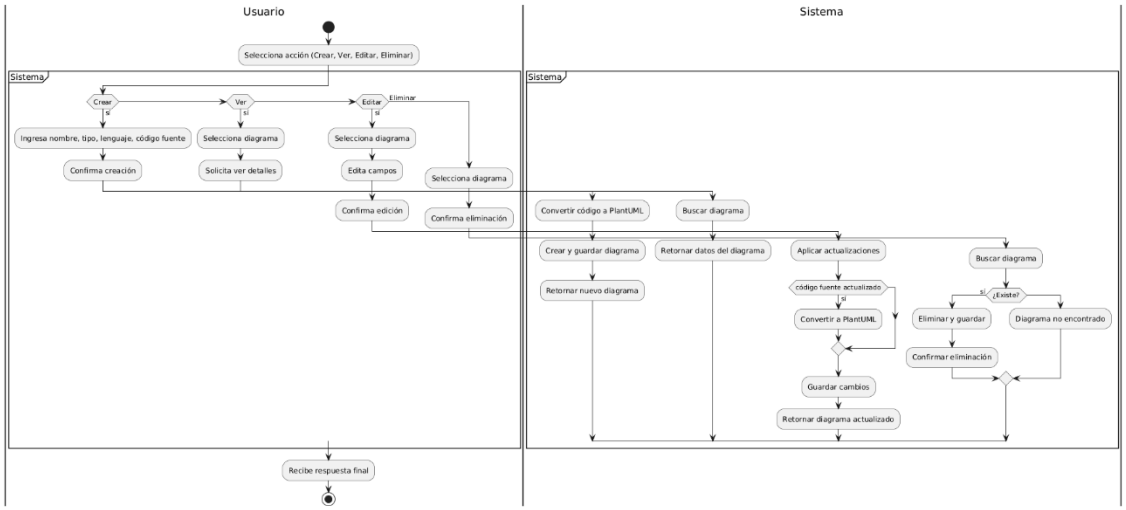


Diagrama de actividades - Autenticación

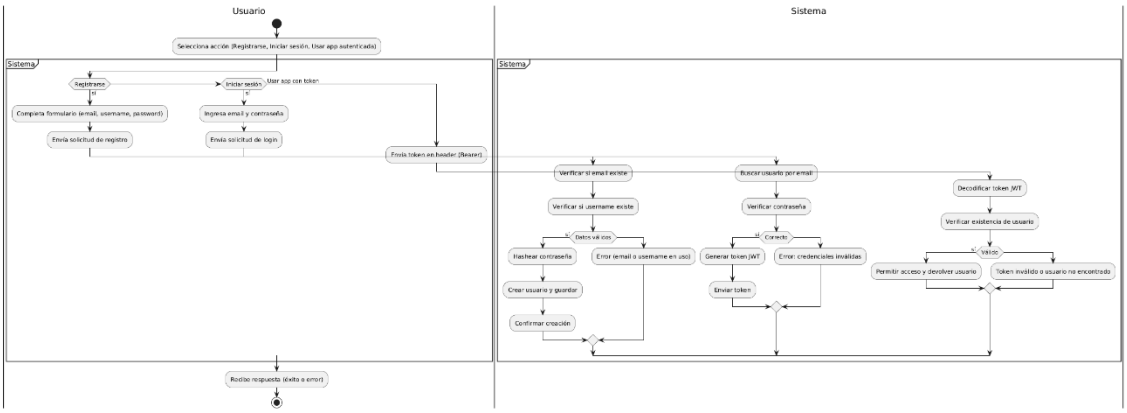


Diagrama de actividades de Versiones

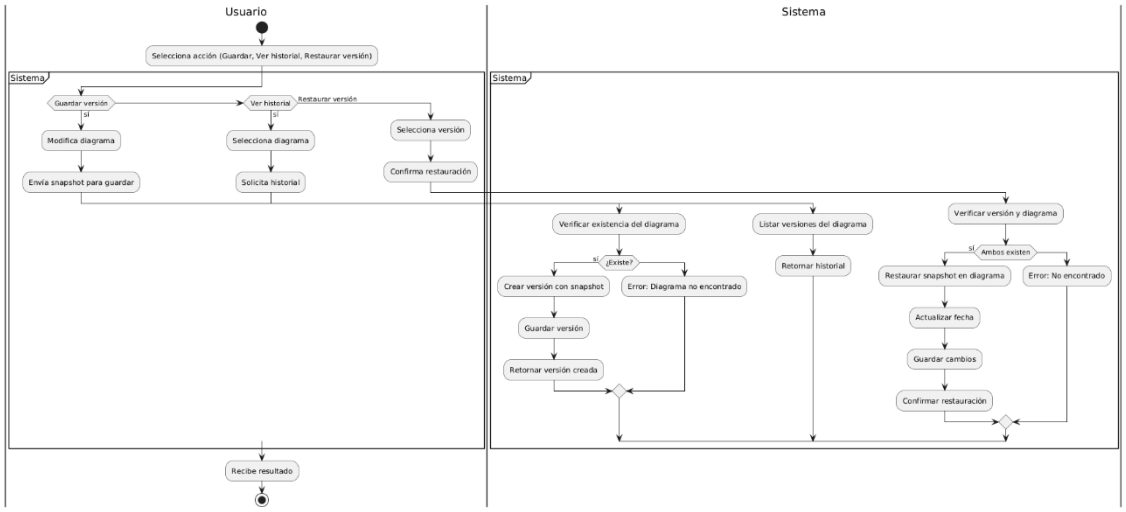
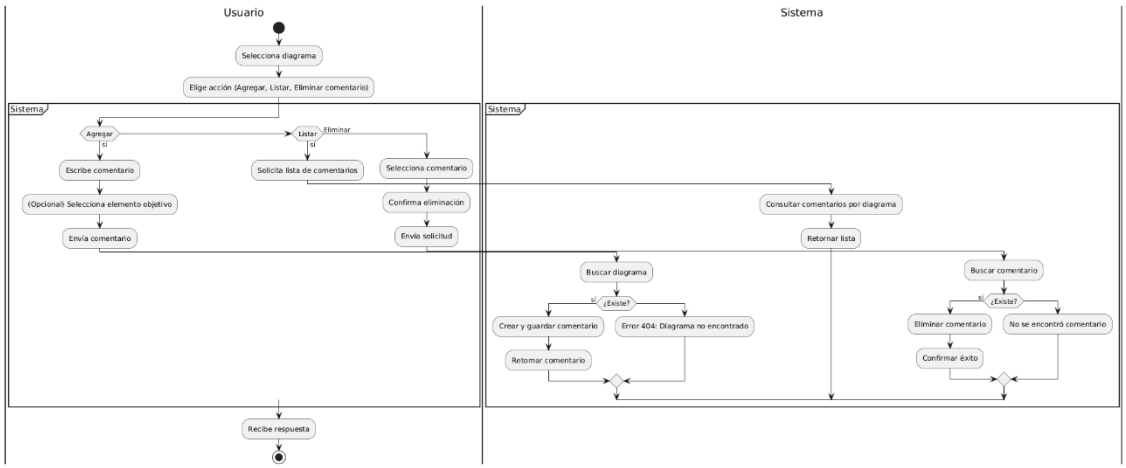


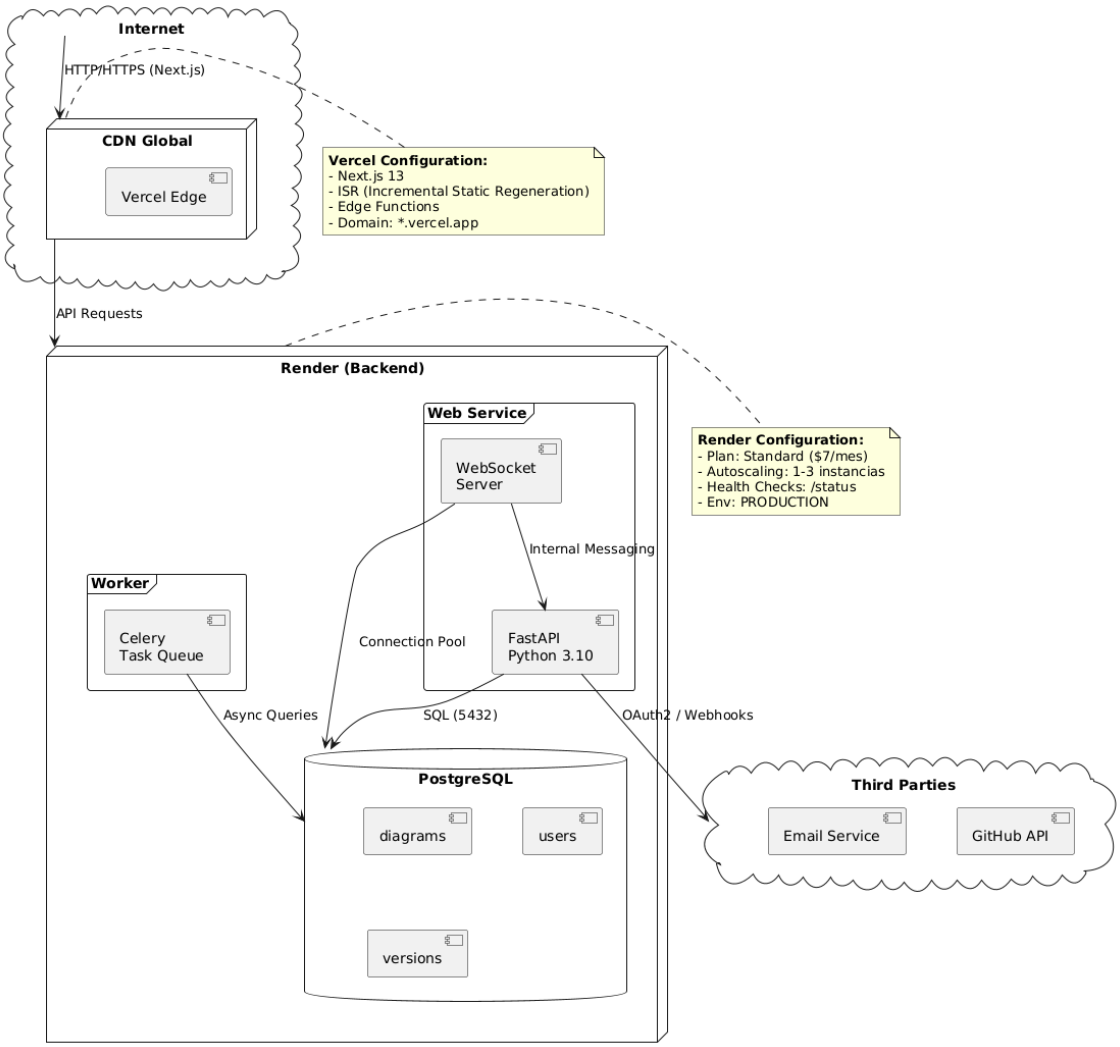
Diagrama de actividades – Comentarios



3.5. Vista de Despliegue (vista física)

3.5.1. Diagrama de despliegue

Este diagrama describe la infraestructura física y virtual donde se ejecuta el sistema, mostrando servidores (nodos), contenedores, servicios en la nube y dispositivos involucrados. Especifica cómo los componentes de software (frontend, backend, bases de datos) se distribuyen en estos entornos, incluyendo tecnologías concretas y protocolos de comunicación (HTTP, TCP/IP). Su propósito es guiar la instalación, escalabilidad y gestión operativa del sistema en producción, reflejando requisitos de disponibilidad, seguridad y rendimiento.



4. ATRIBUTOS DE CALIDAD DEL SOFTWARE

Los Atributos de Calidad (Quality Attributes, QAs) son propiedades medibles y evaluables que describen el comportamiento del sistema más allá de su funcionalidad principal. Mientras que los requerimientos funcionales indican qué hace un sistema, los QAs indican cómo lo hace, y qué tan bien cumple con las expectativas de los stakeholders en aspectos como rendimiento, seguridad, usabilidad, y mantenibilidad.

Según [Wojcik, 2013], los QAs son considerados requerimientos no funcionales. Un sistema puede funcionar correctamente en términos funcionales, pero aún así fallar si no cumple con ciertos atributos de calidad.

A continuación, se describen los principales escenarios asociados a cada atributo de calidad:

Escenario de Funcionalidad

Evalúa las capacidades, características y alcance del sistema.

Considera:

- Cobertura de funciones.
- Generalidad y completitud de operaciones.
- Seguridad lógica de las funciones ofrecidas.

Escenario de Usabilidad

Define la facilidad de uso e interacción del sistema por parte del usuario final.

Incluye:

- Facilidad de aprendizaje.
- Facilidad de uso cotidiano.
- Adaptabilidad a usuarios y tareas.
- Reducción de errores y ayuda al usuario.
- Confianza y satisfacción del usuario.

Escenario de Confiabilidad / Seguridad

Mide la capacidad del sistema para mantener la integridad, disponibilidad y confidencialidad de los datos.

Abarca:

- Prevención: minimizar vulnerabilidades.
- Precaución: detección anticipada de amenazas.
- Reacción: respuesta ante incidentes.
- Enfoques: físico, lógico y humano.

Escenario de Rendimiento

Evalúa la eficiencia del sistema bajo carga.

Métricas clave:

- Tiempo de respuesta.
- Velocidad de procesamiento.
- Uso de recursos (memoria, CPU, red).
- Capacidad de manejo de eventos concurrentes.

Escenario de Mantenibilidad

Se refiere a la facilidad con que el sistema puede ser modificado, ampliado o adaptado.

Subatributos:

- Extensibilidad.
- Adaptabilidad.
- Facilidad de diagnóstico y corrección.