

Logotipo

Descripción generada automáticamente

**“****SOFTWARE PARA DISEÑAR DIAGRAMAS DE ENTIDAD DE BASE DE DATOS DE MANERA COLABORATIVA EN TIEMPO REAL”**

**MATERIA** : INGENIERIA DE SOFTWARE 1

**GRUPO** : SB

**DOCENTE** :ING. MARTINEZ CANEDO ROLANDO

ANTONIO

**NOMBRE** :

**REGISTRO** :

**FECHA** :30/09/2025

Gestión II - 2025

Santa Cruz De La Sierra – Bolivia

Contenido

[1. INTRODUCCIÓN 5](#_Toc208844950)

[2. FUNDAMENTACIÓN TEORICA 6](#_Toc208844951)

[2.1. Desarrollo de software basado en componentes 6](#_Toc208844952)

[2.2. Arquitectura del software 7](#_Toc208844953)

[2.3. Colaboración en tiempo real 7](#_Toc208844954)

[Socket 7](#_Toc208844955)

[2.4. Canvas para el diseño de diagramas de entidad de base de datos 8](#_Toc208844956)

[2.5. Estándar de codificación 9](#_Toc208844957)

[Espaciado y sangría 9](#_Toc208844958)

[Comentarios 9](#_Toc208844959)

[Nombres de variables y funciones 9](#_Toc208844960)

[Manejo adecuado de errores 9](#_Toc208844961)

[2.6. Modelado UML y Diagramas de Clase 10](#_Toc208844962)

[2.7. Tecnologías de Software Moderno 10](#_Toc208844963)

[2.8. Ingeniería de software asistido por computadora 11](#_Toc208844964)

[2.8.1. CASE y la Evolución de la Asistencia por Computadora 11](#_Toc208844965)

[3. OBJETIVOS 12](#_Toc208844966)

[3.1. Objetivo general 12](#_Toc208844967)

[3.2. Objetivo especifico 12](#_Toc208844968)

[4. Alcance 13](#_Toc208844969)

[5. PROCESO DE DESARROLLO DE SOFTWARE 13](#_Toc208844970)

[5.1. Captura de requisitos 13](#_Toc208844971)

[Casos de Uso Detallados - Diagram To SpringBoot 14](#_Toc208844972)

[CU1: Crear cuenta / Registrarse 14](#_Toc208844973)

[CU2: Iniciar sesión / Autenticarse 15](#_Toc208844974)

[CU3: Crear nuevo proyecto/pizarra 15](#_Toc208844975)

[CU4: Unirse a sala / Abrir pizarra 16](#_Toc208844976)

[CU5: Crear clase UML en canvas 17](#_Toc208844977)

[CU6: Editar clase 17](#_Toc208844978)

[CU7: Crear/editar relación UML 18](#_Toc208844979)

[CU8: Eliminar objeto 19](#_Toc208844980)

[CU9: Guardar/auto-guardar diagrama 19](#_Toc208844981)

[CU10: Sincronización en tiempo real 20](#_Toc208844982)

[CU11: Compartir sala / Generar enlace de invitación 21](#_Toc208844983)

[CU12: Generar diagrama con IA 21](#_Toc208844984)

[CU13: Exportar a Spring Boot 22](#_Toc208844985)

[Historias de Usuario - Diagram To SpringBoot 23](#_Toc208844986)

[5.2. Análisis 26](#_Toc208844987)

[5.3. Diseño 26](#_Toc208844988)

[diseño de la arquitectura de software 26](#_Toc208844989)

[Diagrama de paquetes 27](#_Toc208844990)

[Diagrama de Componentes 27](#_Toc208844991)

[Diagrama de despliegue 28](#_Toc208844992)

[Figura 1 Arquitectura cliente-servidor con un único servidor 8](#_Toc164961812)

[Figura 2 Arquitectura cliente servidor 14](#_Toc164961813)

[Figura 3 Diagrama de paquetes 15](#_Toc164961814)

[Figura 4 Diagrama de despliegue 15](#_Toc164961815)

[Figura 5 Inicio del software 22](#_Toc164961816)

[Figura 6 Login del software 23](#_Toc164961817)

[Figura 7 Pizarra 25](#_Toc164961818)

[Figura 8 Componentes de un diagrama de secuencia. 26](#_Toc164961819)

# INTRODUCCIÓN

En el ámbito del desarrollo de software, la comunicación y la colaboración entre los miembros del equipo son aspectos fundamentales para alcanzar el éxito. Una herramienta que facilite y optimice estos procesos puede marcar una diferencia significativa en la eficiencia y la calidad del trabajo final. En este contexto, surge la necesidad de un software especializado en el diseño colaborativo en tiempo real de diagramas de secuencia.

Los diagramas de entidades de base de datos son una herramienta gráfica ampliamente utilizada para modelar la interacción entre objetos en un sistema. Su representación visual permite comprender de forma clara y concisa las entidades que se manejaran en la base de datos. Sin embargo, la creación tradicional de estos diagramas suele ser un proceso individual y estático, lo que puede limitar la fluidez del trabajo colaborativo y la incorporación de ideas de diversos miembros del equipo.

Es por ello que el desarrollo de un software para el diseño colaborativo en tiempo real de diagramas de entidades de base de datos se presenta como una propuesta innovadora y de gran valor para el ámbito del desarrollo de software. Esta herramienta permitiría a los equipos trabajar de forma simultánea en el mismo diagrama, visualizando en tiempo real las modificaciones y aportes de cada participante.

# FUNDAMENTACIÓN TEORICA

## Desarrollo de software basado en componentes

La ingeniería de software basada en componentes (CBSE), también conocida como desarrollo basado en componentes (CBD), es una rama de la ingeniería de software que enfatiza la separación de asuntos, separation of concerns (SoC), por lo que se refiere a la funcionalidad de amplio rango disponible a través de un sistema de software dado. Es un acercamiento basado en la reutilización para definir, implementar, y componer componente de software débilmente acoplados en sistemas. Esta práctica persigue un amplio grado de beneficios tanto en el corto como el largo plazo, para el software en sí mismo y para las organizaciones que patrocinan tal software.

Los ingenieros de software consideran los componentes como parte de la plataforma inicial para la orientación a servicios. Los componentes juegan este rol, por ejemplo, en servicios de web y, más recientemente, en las arquitecturas orientadas a servicios (SOA), por el que un componente es convertido por el servicio web en un servicio y consiguientemente hereda otras características más allá de las de un componente ordinario.

Los componentes pueden producir o consumir eventos y pueden ser usados para las arquitecturas dirigida por eventos (EDA).

Otro aspecto a tener en cuenta en el DSBC, es el poder integrar lo mejor de las tecnologías para realizar aplicaciones personalizadas, ajustadas a los requisitos de los clientes; lo cual le permite a los desarrolladores y/o a la empresa adquirir las tecnologías que más se adapten a sus particularidades, sin incurrir en gastos de licenciamiento o soporte y actualización de grandes soluciones, aunque muchas de estas tecnologías se encuentran bajo la premisa de Freeware y GNU (General Public License), lo cual añade otra gran ventaja.

## Arquitectura del software

La arquitectura de Software hace referencia a la estructura y la relación entre las diferentes partes de un software y sus propiedades visibles externas.

En suma, una arquitectura de Software está compuesta por más arquitecturas de datos articuladas entre sí. Esta es la razón por la que abarca tantos elementos y herramientas para llevarse a cabo.

Su principal objetivo radica en ofrecer cierta calidad al sistema de administración de datos, a partir de su desempeño, ahorro de tiempo, su disponibilidad y usabilidad, la capacidad de modificarse y adecuarse a las nuevas necesidades del sistema, entre otros atributos de calidad.

Implantar una arquitectura de software capacitada para los intereses de una compañía deriva en la reducción de costos, mejora en la toma de decisiones, efectividad de rutas de acción y proyección acertada.

## Colaboración en tiempo real

La colaboración en tiempo real se refiere a la capacidad de trabajar en un proyecto o tarea de forma simultánea con otros usuarios, de manera que las modificaciones realizadas por cada participante se reflejen instantáneamente en la vista de los demás. Esta modalidad de trabajo colaborativo se ha convertido en una herramienta fundamental en diversos ámbitos, especialmente en el desarrollo de software, la gestión de proyectos y la creación de contenido.

### Socket

Un socket es un mecanismo de comunicación entre dos aplicaciones que se ejecutan en diferentes computadoras. Permite establecer una conexión bidireccional entre los procesos, posibilitando el intercambio de datos de forma fluida y eficiente. Los sockets son ampliamente utilizados en aplicaciones de red, como la mensajería instantánea, los juegos en línea y el intercambio de archivos.

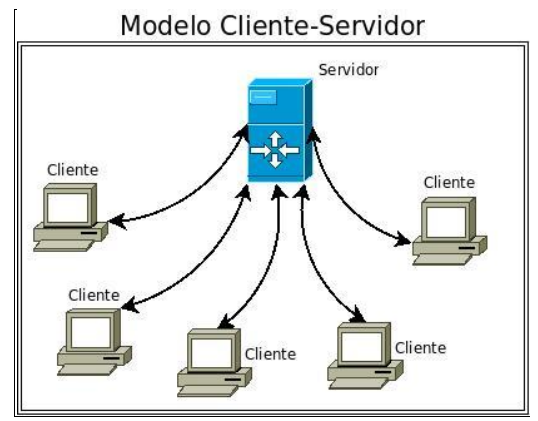


Figura 1 Arquitectura cliente-servidor con un único servidor

## Canvas para el diseño de diagramas de entidad de base de datos

El HTML5 Canvas es un elemento HTML que permite crear gráficos y animaciones directamente en el navegador web. Su uso se ha extendido ampliamente en el desarrollo web moderno, debido a su flexibilidad y capacidad para generar gráficos dinámicos e interactivos. En el contexto del diseño de diagramas de secuencia, Canvas se presenta como una herramienta viable y eficiente para crear diagramas interactivos y personalizables.

Canvas se integra fácilmente con HTML y JavaScript, lo que permite crear diagramas de secuencia que se integran a la perfección con páginas web existentes. Además, JavaScript puede ser utilizado para agregar funcionalidades adicionales a los diagramas, como la animación, la interacción con el usuario y la generación de informes.

## Estándar de codificación

Este documento establece un conjunto de lineamientos para la escritura de código JavaScript en proyectos. La adopción de estos estándares busca promover la legibilidad, mantenibilidad, consistencia y calidad del código, facilitando la colaboración entre desarrolladores y el mantenimiento del mismo a lo largo del tiempo.

### Espaciado y sangría

* Utilizar una sangría consistente de 2 o 4 espacios para mejorar la legibilidad del código.
* Las llaves deben colocarse en la misma línea que la declaración de control (if, for, etc.) y en la siguiente línea para el bloque de código correspondiente.

### Comentarios

* Los comentarios deben usarse para explicar secciones complejas del código o para documentar la funcionalidad de las funciones.
* Se recomienda utilizar un estilo de comentario consistente, como comentarios de bloque o comentarios de línea.

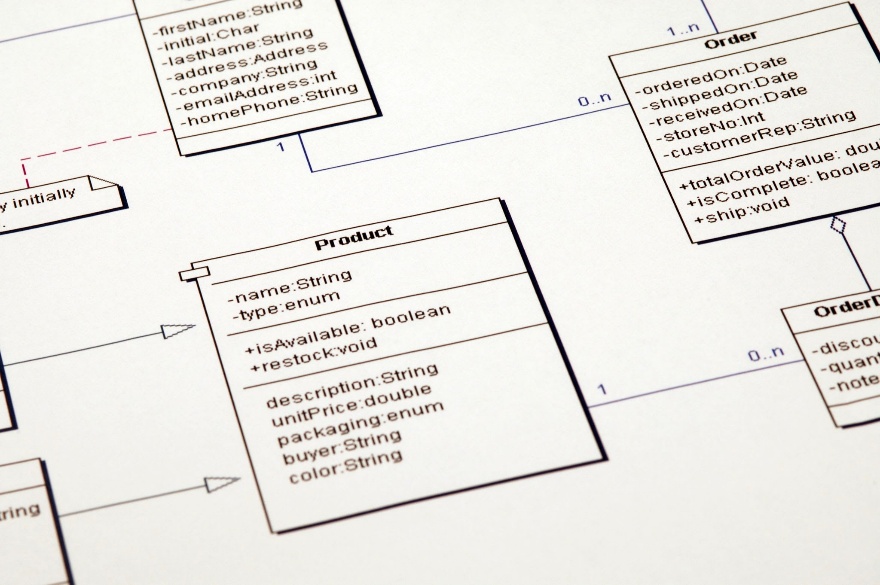
### Nombres de variables y funciones

* Se deben usar nombres de variables y funciones descriptivos que indiquen claramente su propósito. Evite usar nombres abreviados o crípticos.
* Se recomienda utilizar la convención de nomenclatura camel Case para variables y funciones.

### Manejo adecuado de errores

* Se debe implementar un manejo adecuado de errores en todas las partes del código.
* Esto incluye la captura de errores, el registro de errores y la gestión de errores de forma apropiada.

## Modelado UML y Diagramas de Clase

El Lenguaje de Modelado Unificado (UML) es un estándar visual para especificar, construir y documentar artefactos de sistemas de software.

Para este proyecto, el foco principal es el Diagrama de Clase, que representa la estructura estática de un sistema. El software permite a los usuarios definir:

**Clases:** Representadas como cuadros que contienen el nombre de la clase, atributos (propiedades) y operaciones.

**Relaciones:** Conexiones entre clases que denotan interacciones o dependencias, tales como **asociación, herencia, composición** y **asociación bidireccional**. La herramienta permite visualizar estas relaciones de forma clara y estandarizada.

## Tecnologías de Software Moderno

**Frontend (React y Vite):** Se eligió React por su modelo de componentes declarativos, lo que facilita la gestión del estado complejo del diagrama y la interacción del usuario. La manipulación del lienzo y la UI se gestionan a través de componentes eficientes. **Vite** se utiliza como *build tool* para un entorno de desarrollo rápido y ligero.

**Backend (Express.js):** Se optó por Express.js por su flexibilidad y minimalismo. Actúa como un servidor de API RESTful para manejar las peticiones de guardado y generación de proyectos. Su capacidad para manejar WebSockets lo hace ideal para la funcionalidad de colaboración en tiempo real.

**Comunicación en Tiempo Real (WebSockets):** A diferencia del protocolo HTTP, que es transaccional y sin estado, los **WebSockets** establecen un canal de comunicación persistente y bidireccional entre el cliente y el servidor. Esto permite que los cambios realizados por un usuario se propaguen instantáneamente a todos los demás clientes conectados, logrando una verdadera experiencia colaborativa.

**Inteligencia Artificial y Generación de Código:** La integración de un modelo de lenguaje (como **Gemini-2.5-flash**) permite que la herramienta interprete descripciones en lenguaje natural para generar un objeto JSON estructurado que representa el diagrama. Este JSON luego se usa como base para:

* Dibujar el diagrama automáticamente en el lienzo.
* Generar código en un lenguaje de programación específico, como Java para un proyecto de Spring Boot. Esto demuestra la capacidad del software para convertir modelos abstractos en artefactos de software concretos.

## Ingeniería de software asistido por computadora

los agentes de IA y los entornos de desarrollo integrados (IDE) con capacidades de agente de IA se consideran una evolución de la **Ingeniería de Software Asistida por Computadora (CASE)**. Representan la próxima generación de herramientas que automatizan y mejoran el proceso de desarrollo de software.

## CASE y la Evolución de la Asistencia por Computadora

La **Ingeniería de Software Asistida por Computadora (CASE)** se define como el uso de herramientas de software para automatizar las actividades de la ingeniería de software a lo largo de su ciclo de vida, como el análisis de requisitos, el diseño, la codificación, las pruebas y el mantenimiento. Su objetivo principal es aumentar la productividad del desarrollador y mejorar la calidad del software.

Originalmente, las herramientas CASE eran relativamente simples:

**Herramientas de nivel inferior (Lower CASE):** Se centraban en la fase de implementación y mantenimiento, incluyendo compiladores, depuradores y herramientas de control de versiones.

**Herramientas de nivel superior (Upper CASE):** Se enfocaban en las fases iniciales del ciclo de vida, como el modelado de requisitos y el diseño, con herramientas para crear diagramas de flujo de datos o diagramas UML.

Los agentes de IA elevan este concepto a un nivel superior. En lugar de ser solo herramientas pasivas que el desarrollador utiliza, actúan como asistentes proactivos e inteligentes que pueden:

* **Generar código** a partir de una descripción en lenguaje natural.
* **Refactorizar código** de manera automática.
* **Sugerir correcciones** en tiempo real.
* **Escribir documentación** basada en el código.
* **Detectar errores** y vulnerabilidades de seguridad de forma predictiva.

# OBJETIVOS

## Objetivo general

Desarrollar un software para diseñar diagramas de entidad de base de datos de manera colaborativa y en tiempo real para generar un proyecto backend que use spring boot.

## Objetivo especifico

* Analizar los requerimientos del software.
* Diseñar una interface.
* Diseñar la arquitectura del software.
* Diseñar las entidades de la base de datos.
* Implementar las funcionalidades principales del software.
  + Crear proyecto.
  + Diseñar diagrama de secuencia.
  + Colaboración en tiempo real.
  + Compatibilidad con Enterprise Architect.
* Desplegar el software.

# Alcance

**Creación y edición de diagramas de secuencia**: Los usuarios podrán crear nuevos diagramas de secuencia y editar diagramas existentes. El software proporcionará herramientas para dibujar líneas de vida, objetos, mensajes y otros elementos comunes en los diagramas de secuencia.

**Colaboración en tiempo real:** Múltiples usuarios podrán trabajar en el mismo diagrama de secuencia de forma simultánea. Las modificaciones realizadas por un usuario se reflejarán instantáneamente en la vista de los demás usuarios.

**Exportación de diagramas**: Los diagramas de secuencia podrán exportarse en un formato compatible con Enterprise Architect.

# PROCESO DE DESARROLLO DE SOFTWARE

## Captura de requisitos

1. Lista de casos de usos  
   Crear cuenta / Registrarse
2. Iniciar sesión / Autenticarse
3. Crear nuevo proyecto/pizarra
4. Abrir pizarra (canvas) / Unirse a sala
5. Crear clase UML en canvas
6. Editar clase (nombre/atributos/posición)
7. Crear/editar relación UML
8. Eliminar objeto (clase o relación)
9. Guardar/auto-guardar diagrama
10. Sincronización en tiempo real (collaboration)
11. Compartir sala / Generar enlace de invitación
12. Generar diagrama con IA (asistente)
13. Exportar proyecto a Spring Boot (generador de código)

### Casos de Uso Detallados - Diagram To SpringBoot

### CU1: Crear cuenta / Registrarse

|  |
| --- |
|  |

|  |  |
| --- | --- |
| **Nombre del Caso de Uso** | **CU1: Crear cuenta / Registrarse** |
| **Propósito** | Permitir a un usuario nuevo crear una cuenta con credenciales para acceder a funcionalidades persistentes. |
| **Resumen** | El usuario completa un formulario de registro. El sistema valida los datos y crea una nueva cuenta en la base de datos, iniciando la sesión automáticamente. |
| **Actores** | Usuario (nuevo) |
| **Actor Iniciador** | Usuario |
| **Precondiciones** | El formulario de registro debe ser accesible y la base de datos de usuarios debe estar operativa. |
| **Post Condición** | El usuario tiene una cuenta creada y la sesión ha sido iniciada. |
| **Flujo Principal** | 1. El usuario accede a la página de registro. 2. El usuario completa los campos del formulario (ej. nombre, email, contraseña). 3. El usuario envía el formulario. 4. El sistema valida los datos de entrada. 5. Si la validación es exitosa, el sistema crea la cuenta y la guarda en la base de datos. 6. El sistema inicia la sesión del usuario y lo redirige a la página principal. |
| **Flujos Alternativos** | **A1. Validación Fallida:**<br>1a. Si el sistema encuentra un error de validación (ej. campos vacíos, email no válido, contraseña débil), muestra un mensaje de error.<br>2a. El usuario corrige los campos y reenvía el formulario.<br><br>**A2. Usuario ya Existe:**<br>1a. El sistema detecta que el email ya está registrado.<br>2a. Se muestra un mensaje de error indicando que la cuenta ya existe. |

### CU2: Iniciar sesión / Autenticarse

|  |  |
| --- | --- |
| **Nombre del Caso de Uso** | **CU2: Iniciar sesión / Autenticarse** |
| **Propósito** | Autenticar a un usuario registrado para permitirle el acceso a sus proyectos y funcionalidades de colaboración. |
| **Resumen** | El usuario ingresa sus credenciales. El sistema verifica la información y crea una sesión para el usuario, redirigiéndolo al panel de proyectos. |
| **Actores** | Usuario registrado |
| **Actor Iniciador** | Usuario |
| **Precondiciones** | El usuario ya debe tener una cuenta registrada en el sistema. |
| **Post Condición** | El usuario tiene una sesión activa y puede acceder a las funcionalidades del sistema. |
| **Flujo Principal** | 1. El usuario accede a la página de inicio de sesión.<br>2. El usuario ingresa su email y contraseña.<br>3. El usuario envía el formulario.<br>4. El sistema verifica las credenciales con la base de datos.<br>5. Si las credenciales son válidas, el sistema crea una sesión de usuario.<br>6. El sistema redirige al usuario a la página de su panel de proyectos. |
| **Flujos Alternativos** | **A1. Credenciales Inválidas:**<br>1a. El sistema no encuentra una coincidencia para las credenciales.<br>2a. Se muestra un mensaje de error genérico (ej. "Credenciales incorrectas"). |

### CU3: Crear nuevo proyecto/pizarra

|  |  |
| --- | --- |
| **Nombre del Caso de Uso** | **CU3: Crear nuevo proyecto/pizarra** |
| **Propósito** | Permitir a un usuario autenticado crear un nuevo proyecto para comenzar un diagrama. |
| **Resumen** | El usuario inicia el proceso de creación de un proyecto, le asigna un nombre y el sistema lo guarda, abriendo una pizarra de diagramación vacía. |
| **Actores** | Usuario autenticado |
| **Actor Iniciador** | Usuario |
| **Precondiciones** | El usuario debe haber iniciado sesión con éxito. |
| **Post Condición** | Un nuevo proyecto ha sido creado y está disponible para la edición. |
| **Flujo Principal** | 1. El usuario se encuentra en su panel de proyectos.<br>2. El usuario hace clic en el botón "Crear Nuevo Proyecto".<br>3. Se le solicita al usuario un nombre para el proyecto.<br>4. El sistema guarda la configuración inicial del nuevo proyecto.<br>5. El sistema redirige al usuario a la pizarra de diagramación para el nuevo proyecto. |
| **Flujos Alternativos** | **A1. Nombre Duplicado:**<br>1a. El usuario introduce un nombre de proyecto que ya existe.<br>2a. El sistema muestra un mensaje de error y le pide al usuario que elija otro nombre. |

### CU4: Unirse a sala / Abrir pizarra

|  |  |
| --- | --- |
| **Nombre del Caso de Uso** | **CU4: Unirse a sala / Abrir pizarra** |
| **Propósito** | Permitir a un usuario acceder a la pizarra de un proyecto para ver y editar un diagrama. |
| **Resumen** | El usuario accede a un proyecto existente. El sistema carga los datos del diagrama desde el servidor, establece una conexión en tiempo real y muestra el canvas listo para la interacción. |
| **Actores** | Usuario autenticado, Usuario con enlace de invitación |
| **Actor Iniciador** | Usuario |
| **Precondiciones** | El proyecto debe existir y el servidor debe estar operativo. |
| **Post Condición** | El usuario se encuentra en la sala de colaboración con el canvas sincronizado. |
| **Flujo Principal** | 1. El usuario selecciona un proyecto de su lista o accede a través de un enlace de invitación.<br>2. El sistema envía una petición para cargar el estado del diagrama.<br>3. El sistema establece una conexión WebSocket con el servidor.<br>4. El sistema recibe los datos del diagrama y los dibuja en el canvas.<br>5. El canvas se muestra listo para la edición colaborativa. |
| **Flujos Alternativos** | **A1. Proyecto no Encontrado:**<br>1a. El sistema no encuentra el proyecto solicitado.<br>2a. Se muestra una página de error indicando que el proyecto no existe. |

### CU5: Crear clase UML en canvas

|  |  |
| --- | --- |
| **Nombre del Caso de Uso** | **CU5: Crear clase UML en canvas** |
| **Propósito** | Permitir a un usuario añadir una nueva clase a un diagrama en el canvas. |
| **Resumen** | El usuario activa la herramienta para crear una clase. Al hacer clic en el lienzo, se añade una nueva clase con su nombre y atributos, y el cambio se sincroniza con otros colaboradores. |
| **Actores** | Usuario en la pizarra |
| **Actor Iniciador** | Usuario |
| **Precondiciones** | El usuario debe estar en un proyecto con el canvas abierto. |
| **Post Condición** | Una nueva clase es visible en el diagrama y el cambio ha sido notificado al servidor. |
| **Flujo Principal** | 1. El usuario selecciona la herramienta "Clase" en la barra de herramientas.<br>2. El usuario hace clic en un área del canvas.<br>3. Se abre un cuadro de diálogo para que el usuario ingrese el nombre y los atributos de la clase.<br>4. El usuario completa los datos y confirma.<br>5. El sistema dibuja la clase en el canvas y envía una actualización vía WebSocket a los demás usuarios. |
| **Flujos Alternativos** | **A1. Validación de Datos:**<br>1a. El usuario intenta crear una clase con un nombre o atributos no válidos.<br>2a. El sistema muestra un mensaje de error y no permite la creación hasta que se corrijan los datos. |

### CU6: Editar clase

|  |  |
| --- | --- |
| **Nombre del Caso de Uso** | **CU6: Editar clase** |
| **Propósito** | Permitir a un usuario modificar las propiedades o la posición de una clase existente. |
| **Resumen** | El usuario selecciona una clase para editarla. Los cambios se reflejan en el canvas y se envían en tiempo real al servidor para que se sincronicen con otros usuarios. |
| **Actores** | Usuario en la pizarra |
| **Actor Iniciador** | Usuario |
| **Precondiciones** | Debe haber al menos una clase existente en el canvas. |
| **Post Condición** | Los cambios en la clase se persisten en el servidor y se sincronizan con todos los colaboradores. |
| **Flujo Principal** | 1. El usuario hace clic en una clase existente.<br>2. El usuario arrastra la clase para cambiar su posición o hace doble clic para editar sus propiedades.<br>3. Si edita propiedades, se abre un cuadro de diálogo con los campos.<br>4. El usuario modifica los datos (nombre, atributos) y confirma.<br>5. El sistema actualiza la clase en el canvas local y envía el cambio vía WebSocket. |
| **Flujos Alternativos** | **A1. Conflicto de Edición:**<br>1a. Dos usuarios editan la misma clase simultáneamente.<br>2a. El sistema utiliza una estrategia (ej. "el último en guardar gana") para resolver el conflicto y notificar los cambios. |

### CU7: Crear/editar relación UML

|  |  |
| --- | --- |
| **Nombre del Caso de Uso** | **CU7: Crear/editar relación UML** |
| **Propósito** | Permitir a un usuario establecer relaciones entre clases en el diagrama. |
| **Resumen** | El usuario selecciona una herramienta de relación y conecta dos clases. El tipo de relación se define y el cambio se sincroniza en tiempo real. |
| **Actores** | Usuario en la pizarra |
| **Actor Iniciador** | Usuario |
| **Precondiciones** | Debe haber al menos dos clases en el canvas. |
| **Post Condición** | La relación se dibuja en el canvas y se persiste en el servidor. |
| **Flujo Principal** | 1. El usuario selecciona una herramienta de relación (ej. Herencia, Asociación).<br>2. El usuario hace clic en la clase de origen y luego en la clase de destino.<br>3. El sistema dibuja la línea de relación entre las dos clases.<br>4. El sistema envía una actualización vía WebSocket para sincronizar la relación con los demás usuarios. |
| **Flujos Alternativos** | **A1. Destino no Válido:**<br>1a. El usuario intenta conectar una relación a un elemento que no es una clase.<br>2a. El sistema impide la creación de la relación y muestra un mensaje de advertencia. |

### CU8: Eliminar objeto

|  |  |
| --- | --- |
| **Nombre del Caso de Uso** | **CU8: Eliminar objeto** |
| **Propósito** | Permitir a un usuario eliminar una clase o una relación del diagrama. |
| **Resumen** | El usuario selecciona un objeto y lo elimina. El sistema lo borra del canvas y propaga el cambio a todos los colaboradores. |
| **Actores** | Usuario en la pizarra |
| **Actor Iniciador** | Usuario |
| **Precondiciones** | El elemento (clase o relación) a eliminar debe existir y estar seleccionado. |
| **Post Condición** | El objeto se elimina del canvas y de la base de datos, y el cambio se sincroniza en tiempo real. |
| **Flujo Principal** | 1. El usuario selecciona una clase o una relación.<br>2. El usuario hace clic en el botón "Eliminar".<br>3. El sistema le pide confirmación.<br>4. Al confirmar, el sistema elimina el objeto del canvas local y notifica al servidor vía WebSocket para que se elimine en el servidor y en los demás clientes. |
| **Flujos Alternativos** | **A1. Cancelar Eliminación:**<br>1a. El usuario hace clic en "Cancelar" en el cuadro de confirmación.<br>2a. El objeto permanece sin cambios. |

### CU9: Guardar/auto-guardar diagrama

|  |  |
| --- | --- |
| **Nombre del Caso de Uso** | **CU9: Guardar/auto-guardar diagrama** |
| **Propósito** | Persistir el estado actual del diagrama en la base de datos para su recuperación futura. |
| **Resumen** | El sistema guarda automáticamente el diagrama cada cierto tiempo, o el usuario lo guarda manualmente. Los datos se serializan y se envían al servidor. |
| **Actores** | Sistema, Usuario |
| **Actor Iniciador** | Usuario o Sistema |
| **Precondiciones** | Debe existir una conexión activa con el servidor y la base de datos. |
| **Post Condición** | El estado completo del diagrama ha sido guardado de forma segura en la base de datos. |
| **Flujo Principal** | 1. El usuario hace clic en "Guardar" o se activa el temporizador de auto-guardado.<br>2. El sistema serializa el estado del diagrama a un formato JSON.<br>3. El sistema envía el JSON al servidor a través de una petición POST.<br>4. El servidor procesa el JSON y lo almacena en la base de datos.<br>5. El servidor responde con una confirmación de éxito. |
| **Flujos Alternativos** | **A1. Fallo de Conexión:**<br>1a. Si la petición falla debido a un problema de red, el sistema lo detecta.<br>2a. El sistema notifica al usuario del error y reintenta la operación después de un breve retraso. |

### CU10: Sincronización en tiempo real

|  |  |
| --- | --- |
| **Nombre del Caso de Uso** | **CU10: Sincronización en tiempo real** |
| **Propósito** | Mantener a todos los colaboradores sincronizados para que vean los mismos cambios en tiempo real. |
| **Resumen** | Las acciones de un usuario se emiten a través de WebSockets y el servidor las reenvía a todos los demás clientes de la sala. |
| **Actores** | Usuarios conectados |
| **Actor Iniciador** | Usuario(s) |
| **Precondiciones** | Todos los usuarios deben tener una conexión WebSocket estable y estar en la misma sala del proyecto. |
| **Post Condición** | Las acciones de un usuario se reflejan de forma instantánea en los canvas de todos los colaboradores. |
| **Flujo Principal** | 1. El usuario A realiza una acción (ej. mueve una clase).<br>2. La acción se emite como un evento a través del WebSocket.<br>3. El servidor recibe el evento y lo reenvía a todos los demás clientes conectados a la misma sala.<br>4. Los clientes B, C, etc. reciben el evento y actualizan sus canvas. |
| **Flujos Alternativos** | **A1. Desconexión del WebSocket:**<br>1a. Un usuario se desconecta de la sala.<br>2a. Al reconectar, el cliente solicita el estado completo del diagrama al servidor para "reconciliarse" y sincronizarse. |

### CU11: Compartir sala / Generar enlace de invitación

|  |  |
| --- | --- |
| **Nombre del Caso de Uso** | **CU11: Compartir sala / Generar enlace de invitación** |
| **Propósito** | Permitir al propietario o a los colaboradores de un proyecto invitar a otros usuarios a unirse. |
| **Resumen** | Un usuario genera un enlace único de invitación que, al ser compartido, permite a otros unirse al proyecto con permisos de visualización o edición. |
| **Actores** | Propietario/Usuario del proyecto |
| **Actor Iniciador** | Usuario |
| **Precondiciones** | El usuario debe tener los permisos necesarios para compartir el proyecto. |
| **Post Condición** | Un enlace de invitación válido ha sido generado y copiado. |
| **Flujo Principal** | 1. El usuario hace clic en el botón "Compartir".<br>2. El sistema genera un enlace URL único con un token de invitación.<br>3. El enlace se muestra en un cuadro de diálogo y se copia automáticamente al portapapeles.<br>4. El usuario puede compartir el enlace con otros colaboradores. |
| **Flujos Alternativos** | **A1. Acceso Revocado:**<br>1a. El propietario del proyecto revoca el acceso del enlace.<br>2a. El enlace se vuelve inválido y los usuarios que intenten usarlo recibirán un error. |

### CU12: Generar diagrama con IA

|  |  |
| --- | --- |
| **Nombre del Caso de Uso** | **CU12: Generar diagrama con IA** |
| **Propósito** | Asistir al usuario en la creación de diagramas utilizando un agente de IA que interpreta lenguaje natural. |
| **Resumen** | El usuario proporciona una descripción textual de un diagrama. El sistema la envía a la API de Gemini, que devuelve un JSON estructurado para dibujar el diagrama automáticamente en el canvas. |
| **Actores** | Usuario en la pizarra |
| **Actor Iniciador** | Usuario |
| **Precondiciones** | El servicio de la API de Gemini debe estar disponible y la clave de API debe estar configurada en el backend. |
| **Post Condición** | Nuevos elementos del diagrama (clases, relaciones) han sido añadidos al canvas. |
| **Flujo Principal** | 1. El usuario hace clic en el botón "Asistente de IA" o "Hablar para Generar".<br>2. Se abre un modal donde el usuario puede escribir o dictar su descripción.<br>3. El usuario envía la descripción al servidor.<br>4. El servidor realiza una llamada a la API de Gemini.<br>5. Gemini devuelve una respuesta en formato JSON.<br>6. El servidor valida y reenvía el JSON al cliente.<br>7. El cliente parsea el JSON y actualiza el estado del diagrama, dibujando los nuevos elementos. |
| **Flujos Alternativos** | **A1. Respuesta Inválida de la IA:**<br>1a. La respuesta de la IA no se adhiere al formato JSON esperado.<br>2a. El sistema muestra un mensaje de error y notifica al usuario. |

### CU13: Exportar a Spring Boot

|  |  |
| --- | --- |
| **Nombre del Caso de Uso** | **CU13: Exportar a Spring Boot** |
| **Propósito** | Permitir al usuario transformar su diagrama de clases en un proyecto backend de Spring Boot y descargarlo. |
| **Resumen** | El sistema serializa el diagrama y lo envía a un servicio generador de código, el cual crea un archivo ZIP con el proyecto completo y lo envía de vuelta para su descarga. |
| **Actores** | Usuario en la pizarra |
| **Actor Iniciador** | Usuario |
| **Precondiciones** | El servicio generador de Spring Boot debe estar disponible y operativo. |
| **Post Condición** | El usuario tiene un archivo ZIP descargado que contiene el proyecto de Spring Boot. |
| **Flujo Principal** | 1. El usuario hace clic en el botón "Generar Proyecto".<br>2. El sistema serializa el diagrama actual en un formato JSON o una estructura de datos compatible.<br>3. El sistema envía esta estructura a un servicio de backend dedicado a la generación.<br>4. El servicio genera un proyecto de Spring Boot a partir de los datos.<br>5. El servicio empaqueta el proyecto en un archivo ZIP y lo envía de vuelta al cliente.<br>6. El navegador fuerza la descarga del archivo ZIP para el usuario. |
| **Flujos Alternativos** | **A1. Error del Generador:**<br>1a. El servicio generador encuentra un error (ej. diagrama inválido, configuración incorrecta).<br>2a. El servicio retorna un error con detalles, que el cliente muestra al usuario. |

## Análisis

### Análisis de Paquetes

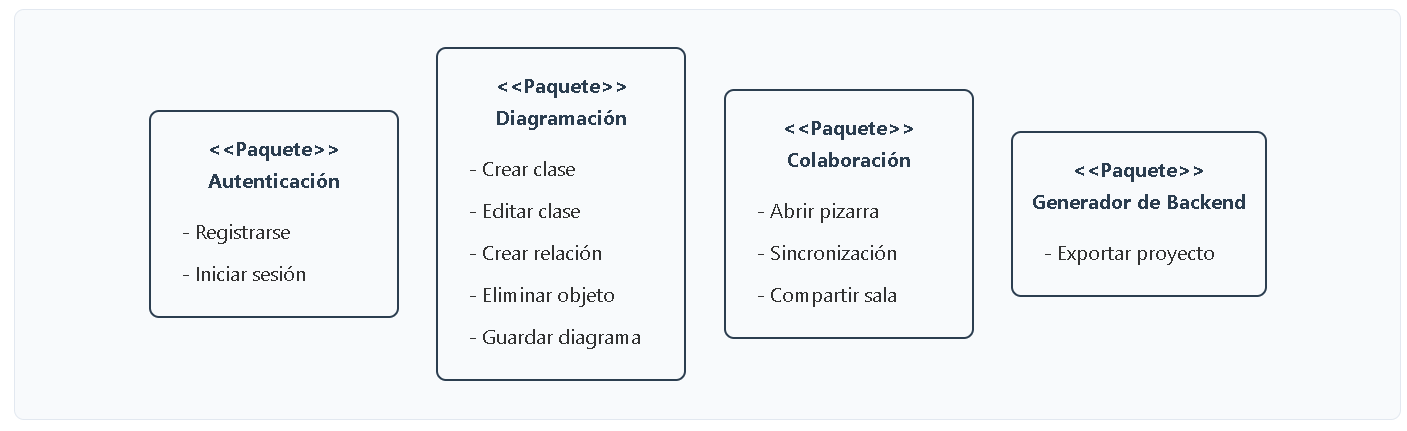


Diagrama de Paquetes que muestra la modularidad del sistema y las dependencias entre componentes

# Historias de Usuario - Diagram To SpringBoot

|  |  |  |
| --- | --- | --- |
| **Historia de Usuario** | **Descripción Detallada** | **Criterios de Aceptación** |
| **HU-1: Crear Cuenta / Registrarse** | Como usuario nuevo, quiero registrarme con mis credenciales para acceder a funcionalidades persistentes y personalizadas de la aplicación. | **GIVEN** que estoy en la página de registro<br>**WHEN** completo el formulario con mis datos y pulso "Registrarse"<br>**THEN** se crea mi cuenta, mis datos se guardan en la base de datos y se inicia mi sesión automáticamente. <br><br> **Criterio de Rechazo:** Si el usuario ya existe, se muestra un mensaje de error y no se crea una nueva cuenta. Si la validación falla (ej. contraseña débil), se notifica al usuario. |
| **HU-2: Iniciar Sesión / Autenticarse** | Como usuario registrado, quiero iniciar sesión para acceder a mis proyectos y participar en la colaboración en tiempo real. | **GIVEN** que estoy en la página de inicio de sesión<br>**WHEN** ingreso mis credenciales y pulso "Entrar"<br>**THEN** el sistema verifica mi información y me redirige a mi lista de proyectos. <br><br> **Criterio de Rechazo:** Si las credenciales son inválidas, se muestra un mensaje de error sin revelar detalles específicos. |
| **HU-3: Crear Nuevo Proyecto/Pizarra** | Como usuario autenticado, quiero crear un nuevo proyecto/pizarra para comenzar un nuevo diagrama. | **GIVEN** que he iniciado sesión y estoy en la página principal<br>**WHEN** pulso el botón "Crear Nuevo Proyecto"<br>**THEN** se me pide un nombre para el proyecto, se crea y se me redirige a la pizarra de diagramación. |
| **HU-4: Unirse a Sala / Abrir Pizarra** | Como usuario, quiero unirme a una sala de colaboración para ver y editar un diagrama existente. | **GIVEN** que he seleccionado un proyecto o he recibido un enlace de invitación<br>**WHEN** abro la pizarra<br>**THEN** la aplicación carga el estado actual del diagrama desde el servidor, establece una conexión WebSocket y me muestra el canvas para interactuar. <br><br> **Criterio de Rechazo:** Si el proyecto no existe o no tengo permisos, se me muestra un error. |
| **HU-5: Crear Clase UML** | Como colaborador, quiero añadir una clase a mi diagrama para modelar la estructura del software. | **GIVEN** que estoy en la pizarra<br>**WHEN** selecciono la herramienta "Clase", hago clic en el canvas y completo los datos<br>**THEN** se dibuja una nueva clase en el canvas, se emite un evento vía WebSocket y la clase se sincroniza con los demás colaboradores. |
| **HU-6: Editar Clase** | Como colaborador, quiero modificar las propiedades o la posición de una clase para refinar mi diagrama. | **GIVEN** que hay una clase en el canvas<br>**WHEN** edito su nombre, atributos o la arrastro a una nueva posición<br>**THEN** los cambios se actualizan en el canvas y se sincronizan en tiempo real con los demás usuarios a través del WebSocket. |
| **HU-7: Crear/Editar Relación** | Como colaborador, quiero crear relaciones entre clases para representar sus interacciones. | **GIVEN** que hay al menos dos clases<br>**WHEN** selecciono la herramienta de relación, hago clic en la clase de origen y luego en la de destino, y selecciono el tipo de relación<br>**THEN** se dibuja la relación en el canvas y los cambios se sincronizan en tiempo real. |
| **HU-8: Eliminar Objeto** | Como colaborador, quiero eliminar una clase o una relación del diagrama. | **GIVEN** que hay un objeto en el canvas<br>**WHEN** lo selecciono y pulso el botón de eliminar<br>**THEN** el objeto desaparece del canvas y la acción se sincroniza en tiempo real con los demás colaboradores. |
| **HU-9: Sincronización en Tiempo Real** | Como colaborador, quiero ver los cambios de otros usuarios en tiempo real, sin tener que recargar la página. | **GIVEN** que estoy conectado a la sala de colaboración<br>**WHEN** otro usuario realiza una acción (crear, editar, eliminar)<br>**THEN** mi canvas se actualiza automáticamente para reflejar los cambios. |
| **HU-10: Guardar y Cargar Diagrama** | Como usuario, quiero guardar el estado de mi diagrama para poder acceder a él en el futuro. | **GIVEN** que he modificado el diagrama<br>**WHEN** pulso "Guardar" o el sistema realiza un auto-guardado<br>**THEN** el estado completo del diagrama se envía al backend y se guarda en la base de datos. <br><br> **Criterio de Rechazo:** Si hay un error de conexión, se notifica al usuario y se permite reintentar. |
| **HU-11: Compartir Sala / Enlace** | Como propietario, quiero generar un enlace de invitación para que otros usuarios puedan unirse a mi proyecto. | **GIVEN** que tengo un proyecto en la pizarra<br>**WHEN** pulso el botón "Compartir"<br>**THEN** la aplicación genera un enlace único que puedo copiar y compartir para que otros se unan al canvas colaborativo. |
| **HU-12: Generar Diagrama con IA** | Como usuario, quiero describir un diagrama en lenguaje natural para que el asistente de IA lo genere por mí. | **GIVEN** que estoy en la pizarra y tengo acceso al asistente<br>**WHEN** escribo o dicto una descripción del diagrama que deseo<br>**THEN** el sistema envía mi descripción a la API de Gemini, recibe un JSON estructurado y automáticamente crea las clases y relaciones en el canvas. |
| **HU-13: Exportar Proyecto a Spring Boot** | Como usuario, quiero convertir mi diagrama de clases en un proyecto backend de Spring Boot y poder descargarlo. | **GIVEN** que mi diagrama está completo<br>**WHEN** pulso el botón "Generar Proyecto"<br>**THEN** el sistema envía la estructura del diagrama al servidor generador, crea un archivo .zip con el proyecto de Spring Boot y me lo ofrece para descargar. |

## Diseño

### diseño de la arquitectura de software

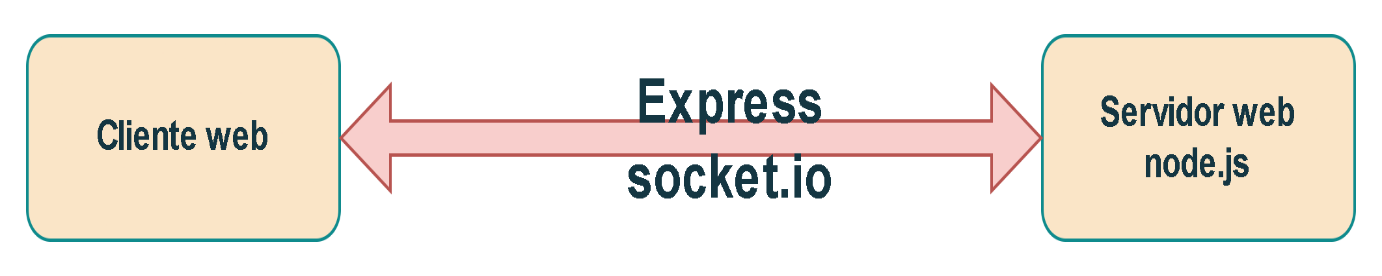
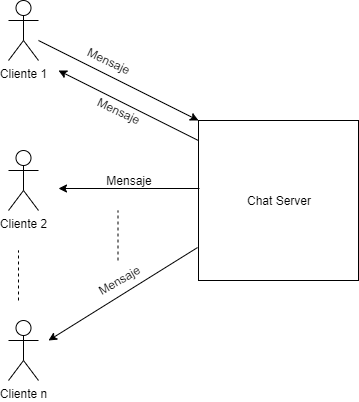


Figura 2 Arquitectura cliente servidor



### Diagrama de paquetes

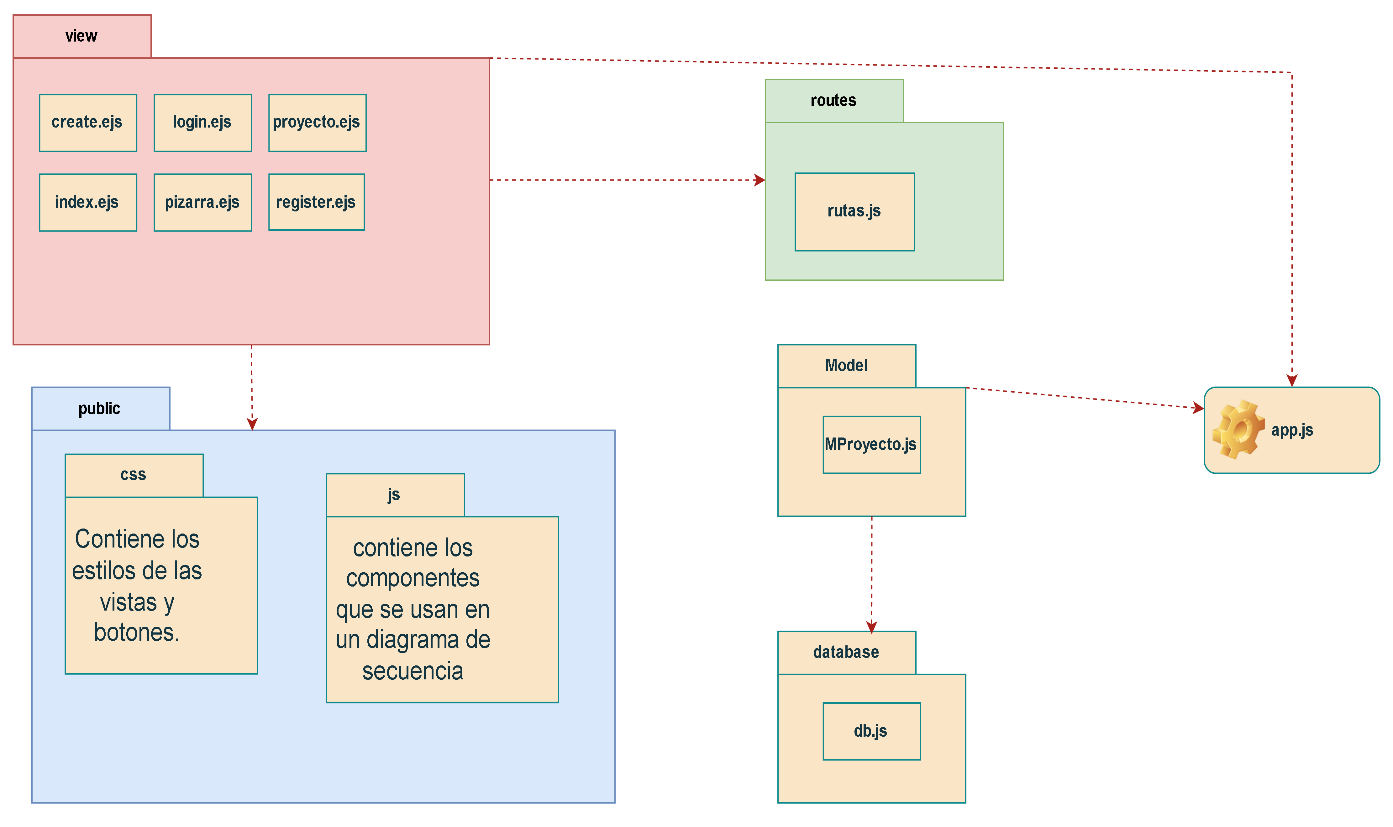
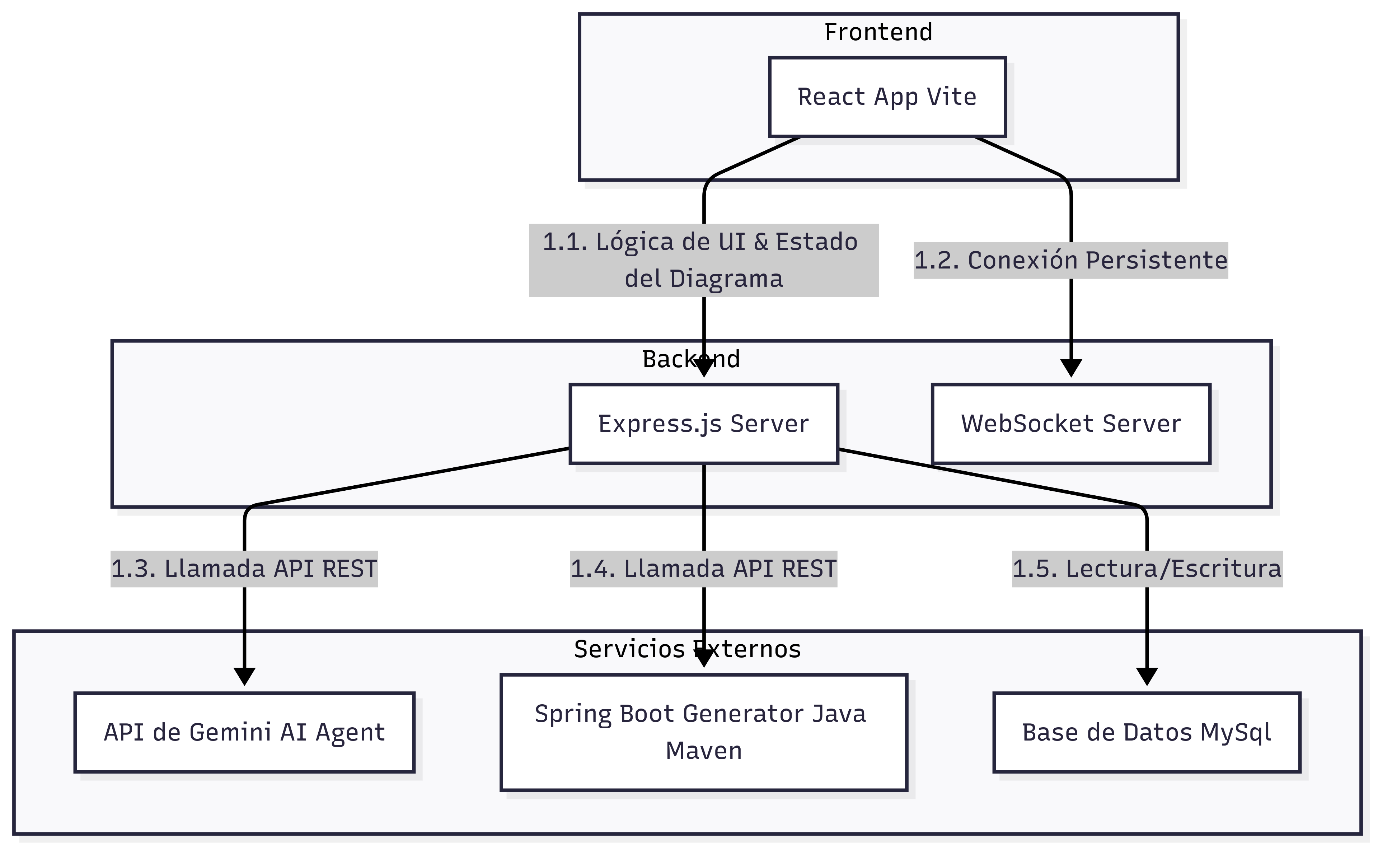


Figura 3 Diagrama de paquetes

### Diagrama de Componentes



### Diagrama de despliegue

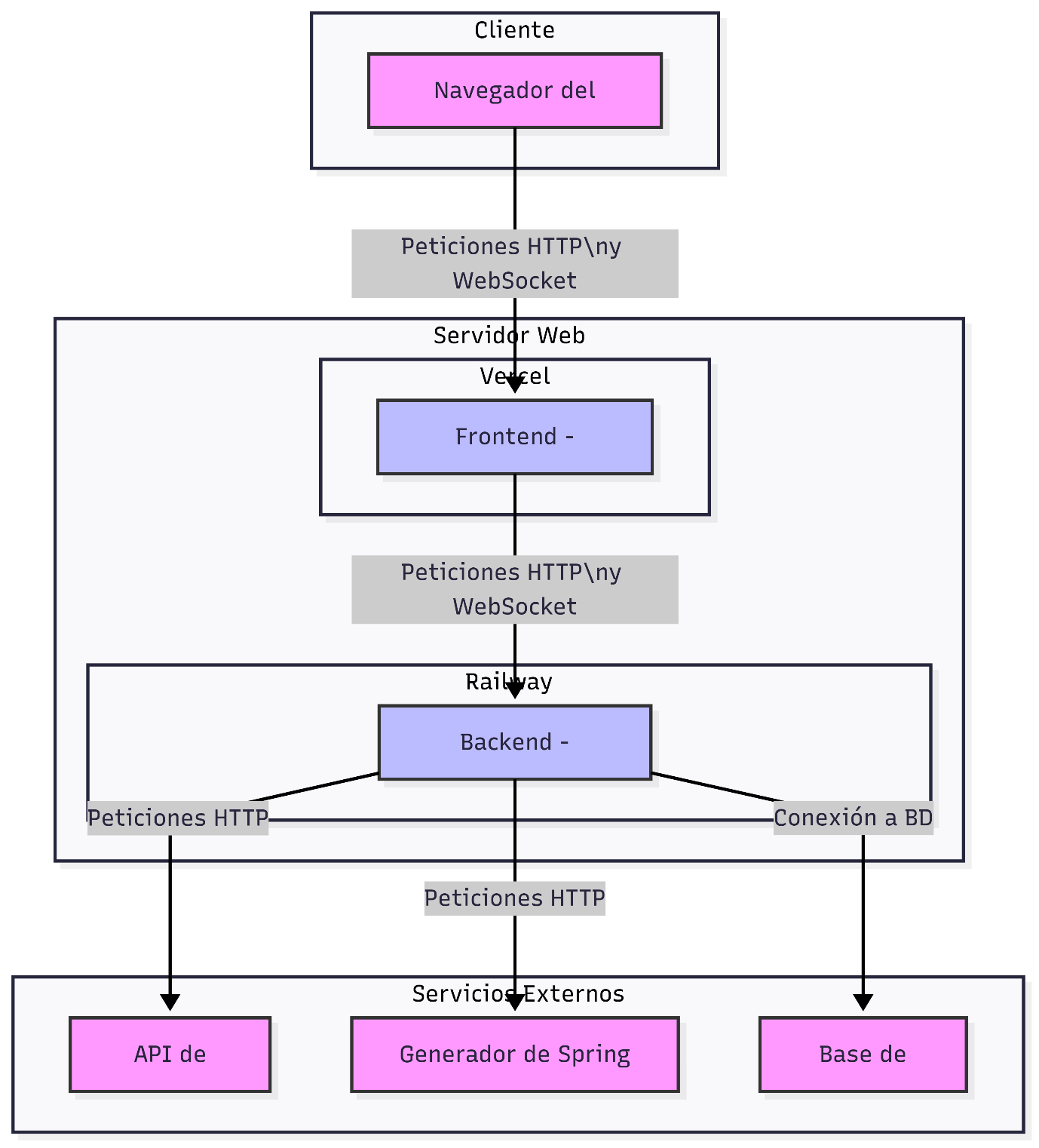


Figura 4 Diagrama de despliegue

## Implementación

🛠 COMPONENTES IMPLEMENTADOS

Backend:

✅ Servidor Express con APIs REST

✅ Servidor WebSocket con autenticación JWT

✅ Modelos de base de datos para:

✅ Users (usuarios)

✅ Projects (proyectos)

✅ Diagrams (diagramas)

✅ Collaborators (colaboradores)

✅ Invitations (invitaciones)

Frontend:

✅ Sistema de autenticación (login/registro)

✅ Gestión de proyectos (crear, listar, editar, eliminar)

✅ Editor de diagramas UML con canvas

✅ Panel de colaboración con chat

✅ Servicio WebSocket para comunicación en tiempo real

✅ Componentes modales (proyectos, código, etc.)

Servicios:

✅ api.js - Cliente HTTP para APIs REST

✅ websocketService.js - Cliente WebSocket

✅ codeGenerationService.js - Generación de código Java

✅ exportService.js - Exportación de imágenes y proyectos

# MANUAL DE USUARIO

ANEXOS