

TALLER FASE DE DISEÑO

DOCENTE: RUBÉN DARÍO GONZÁLEZ BARRERA

ASIGNATURA:

ARQUITECTURA DE COMPUTADORES

ESTUDIANTES:

DEIBID BENAVIDES YAYA

MATEO RAMIREZ ORTIZ

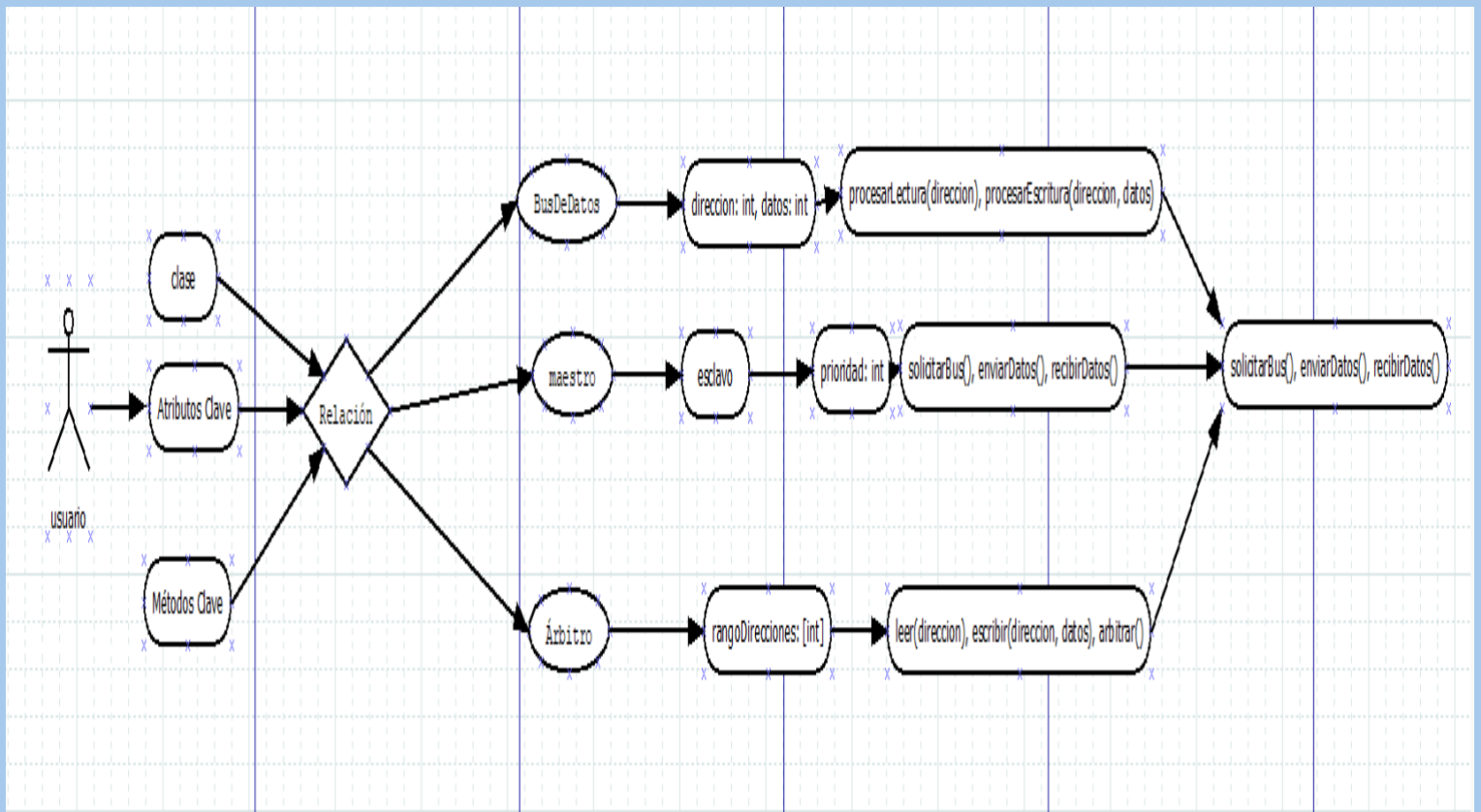
MATEO ALEJANDRO VARGAS VALERO

NRC: 80983

INSTITUCIÓN: UNIVERSIDAD MINUTO DE DIOS

FECHA DE ENTREGA: 15/11/2025

DIAGRAMA UML



BOCETO DE DIAGRAMA DE SECUENCIA (CREAR CONTACTO)

Actor/Componente	Descripción
Cliente/UI	La aplicación o interfaz del usuario (móvil, web) que inicia la acción.
API Gateway/Controlador	El punto de entrada inicial, valida la solicitud y la pasa al servicio.
Servicio Contactos	El microservicio responsable de la lógica de negocio de los contactos.
BUS DE EVENTOS	El componente central de mensajería (Ej: Kafka, RabbitMQ) que transporta los eventos.
Servicio Persistencia	El microservicio responsable de almacenar los datos en la base de datos.
Servicio Notificaciones	Un servicio secundario que reacciona al evento (Ej: envía un email de bienvenida).

Diagrama

participan UI as Cliente/UI

participan API as API Gateway/Controlador

participan SC as Servicio Contactos

participan BUS as BUS DE EVENTOS

participan SP as Servicio Persistencia

participan SN as Servicio Notificaciones

UI->>API: 1. POST /contactos (Datos de Contacto)

actíivate API

API->>SC: 2. Crear Contacto (Datos)

actíivate SC

Note light of SC: Validación de Datos

SC->>SC: 3. Generar ID y Objeto Contacto

Note light of SC: El servicio prepara el evento.

SC->>BUS: 4. Publicar Evento: "Contacto Creado"

de actíivate SC

API->>UI: 5. 202 Aceptad (Respuesta Rápida)

de actíivate API

Note light of BUS: El BUS enruta el evento a todos los suscriptores.

BUS->>SP: 6. Consumir Evento: "Contacto Creado"

actíivate SP

SP->>SP: 7. Almacenar Contacto en DB

de actíivate SP

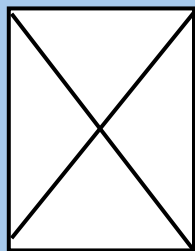
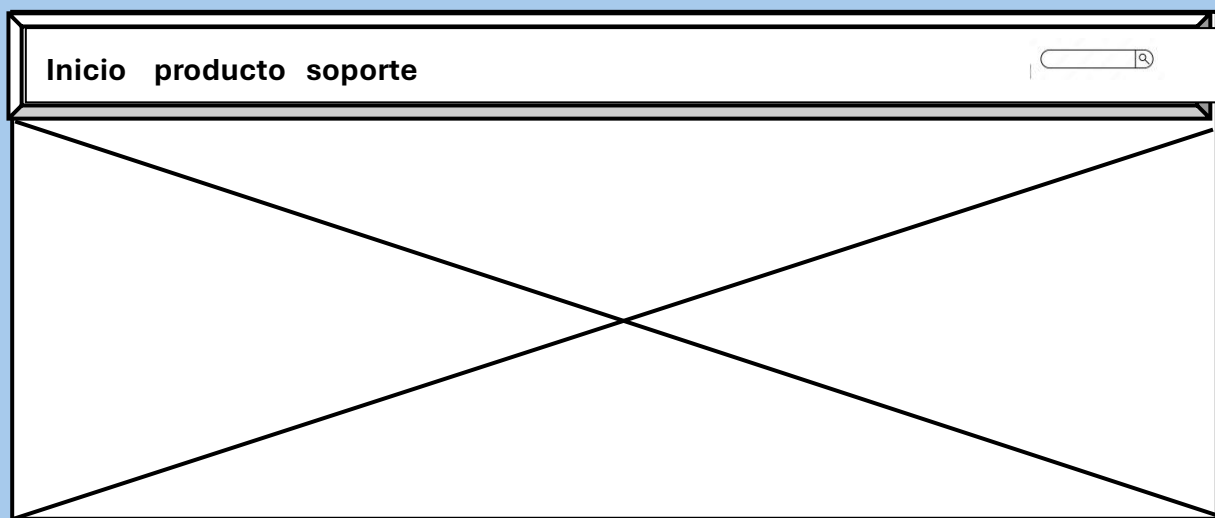
BUS->>SN: 8. Consumir Evento: "Contacto Creado"

actíivate SN

SN->>SN: 9. Enviar Email de Bienvenida

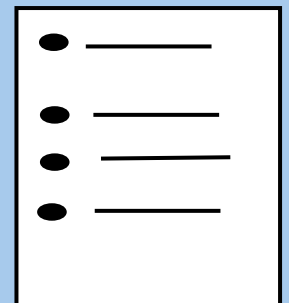
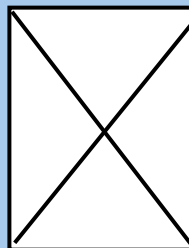
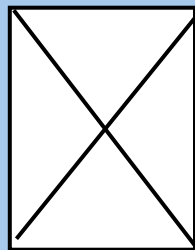
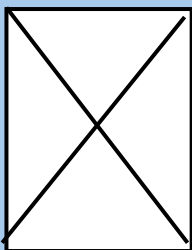
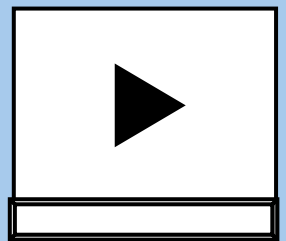
de actíivate SN

BOCETO O WIREFRAMES DE LAS INTERFACES



Sesión

salir



DEFINICIÓN DE LA ARQUITECTURA DEL SISTEMA

La arquitectura más adecuada para un Sistema de Bus de Datos Simple es la Arquitectura Orientada a Eventos (EDA) o, más específicamente, una arquitectura de Microservicios con Patrón Publicador/Suscriptor.

El objetivo central de esta arquitectura es desacoplar los diferentes componentes o servicios, permitiendo que operen y evolucionen de manera independiente mientras se comunican a través de un Bus de Eventos centralizado.

Componente	Función Principal	Rol en el Bus
Microservicios (Endpoints)	Lógica de negocio específica (Ej: Servicio Contactos, Servicio Pedidos).	Actúan como Publicadores (cuando generan un evento) y Suscriptores (cuando reaccionan a un evento).
Bus de Eventos (Broker)	Componente central para recibir, almacenar y distribuir mensajes de forma confiable.	El Intermediario que garantiza la entrega del evento a todos los suscriptores interesados. (Ejemplos tecnológicos: Apache Kafka, RabbitMQ).
Tópicos (Temas/Canales)	Los canales lógicos dentro del Bus a donde se envían los eventos.	La Clasificación de los mensajes (Ej: contactos.creado, pedidos.actualizado).

Componente	Función Principal	Rol en el Bus
Base de Datos (DB)	Almacena el estado persistente de cada servicio.	Puede ser Políglota, permitiendo a cada servicio elegir la DB más adecuada para su tarea.

Principios Arquitectónicos

Desacoplamiento (Decoupling): Los servicios no se conocen directamente entre sí. El Servicio A no sabe *quién* necesita el evento que publica, solo lo publica. El Servicio B no sabe *quién* lo generó, solo lo consume.

Asincronía: La comunicación principal se realiza mediante eventos que se procesan sin esperar una respuesta inmediata. Esto mejora la latencia y la escalabilidad.

Resiliencia: Si un servicio consumidor falla (ej. Servicio Notificaciones), el evento permanece en el Bus, y el servicio puede procesarlo una vez que se recupere, evitando la pérdida de datos.

Escalabilidad: Cada microservicio puede escalarse de forma independiente en función de su carga de trabajo. Si el Servicio Persistencia está saturado, se escala solo ese componente.

Flujo de Datos (Ejemplo: Crear Contacto)

Comando (Síncrono): El API Gateway recibe una solicitud HTTP para crear un contacto.

Conversión a Evento: El Servicio Contactos ejecuta su lógica y, en lugar de llamar directamente a la DB, publica un Evento (Contacto Creado) en el Bus de Eventos.

Procesamiento (Asíncrono): El Servicio Persistencia está suscrito al tópico contactos. Creado, consume el evento y guarda los datos en su DB.

El Servicio Notificaciones también está suscrito, consume el evento y envía el correo de bienvenida.