Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

TC3005B: Desarrollo e Implementación de Sistemas de Software

Documento de diseño (SDD)

Equipo 4:

Eduardo Zentella Castillo	| A00835387
Cesar Ivan Hernandez Melendez	| A00829868
Jose David de la Garza Salas	| A00834760
Pablo Andrés Martínez Sánchez	| A01252489
Javier Eduardo Corrales Cardoza	| A01742328

Lugar y Fecha:

Monterrey, Nuevo León. 13 de junio de 2024.

1. Índice

1.1 Tabla de contenidos

2. Introducción

2.1 Propósito

El propósito de este documento es proporcionar una descripción detallada del diseño del sistema de OracleBot, un chatbot desarrollado para interactuar con los usuarios de manera eficiente y automatizada. Este documento servirá como una guía para los desarrolladores y otros interesados, asegurando una comprensión clara y unificada del diseño y arquitectura del sistema.

2.2 Alcance

Este documento abarca todos los aspectos del diseño del OracleBot, incluyendo su arquitectura general, diseño de componentes, interfaces de usuario y datos, y consideraciones de rendimiento y seguridad. No se abordarán aspectos relacionados con la implementación detallada o el código fuente específico.

2.3 Definiciones, Acrónimos y Abreviaturas

- **API:** Application Programming Interface
- **UI:** User Interface
- **SDD:** Software Design Document
- **IEEE:** Institute of Electrical and Electronics Engineers

2.4 Referencias

Laboratorio de Telegram:
https://apexapps.oracle.com/pls/apex/r/dbpm/livelabs/run-workshop?p210_wid=3701&p210_wec=&session=1183742264263
https://github.com/AdanRuiz/oci-react-samples

2.4 Visión General del Documento

Este documento está estructurado en varias secciones que detallan la arquitectura del sistema, el diseño de sus componentes principales, las interfaces, los requisitos no funcionales y las pruebas y validación del sistema. Cada sección proporciona la información necesaria para entender y desarrollar el sistema conforme a los estándares establecidos.

3. Descripción General del Sistema

3.1 Perspectiva del Sistema

OracleBot es un sistema de chatbot diseñado para interactuar con los usuarios proporcionando respuestas automatizadas a los comandos del sistema. El sistema se integra con la plataforma de comunicación de Telegram y la base de datos de Oracle, permitiendo la gestión eficiente de las interacciones con los usuarios. La arquitectura del sistema está basada en componentes modulares para facilitar su mantenimiento y escalabilidad.

3.2 Funcionalidad del Sistema

OracleBot ofrece las siguientes funcionalidades principales:

- **Interacción con el Usuario:** Responde a los comandos de los usuarios en lenguaje natural.
- **Gestión de Consultas:** Procesa y sabe diferenciar los comandos del usuario, derivando al módulo correspondiente.
- **Acceso a la Base de Datos:** Consulta y actualiza información en la base de datos según las interacciones del usuario.
- **Integración con Telegram:** Se comunica con la plataforma de Telegram.

3.3 Características del Usuario

El sistema está diseñado para ser utilizado por:

- **Desarrolladores:** Desarrolladores que interactúan con el chatbot para obtener, crear, y eliminar sus tareas.
- **Managers:** Manager que interactúan con el chatbot para obtener las tareas de todo su equipo de trabajo.
- **Administradores del Sistema:** Usuarios que gestionan y supervisan el funcionamiento del OracleBot, incluyendo la configuración y mantenimiento del sistema.

3.4 Restricciones

El diseño del sistema debe considerar las siguientes restricciones:

- **Compatibilidad:** Debe ser compatible con la plataforma de Telegram.
- **Seguridad:** Debe garantizar la privacidad y seguridad de los datos del usuario.
- **Rendimiento:** Debe ser capaz de manejar múltiples comandos simultáneamente sin degradar el rendimiento.

4. Arquitectura del Sistema

4.1 Descripción de la Arquitectura

La arquitectura del sistema OracleBot está diseñada para ser modular y escalable, aprovechando el framework Spring Boot para la gestión del backend y una estructura de comandos para la interacción con el usuario. A continuación, se presenta una vista general de la arquitectura del sistema, incluyendo los principales componentes y su interacción.

El sistema está compuesto por los siguientes módulos principales:

- **Interfaz de Usuario (UI):** Maneja la interacción con el usuario, permitiendo el ingreso de comandos para gestionar las tareas.
- **Controlador (Controller):** Procesa los comandos recibidos desde la UI y delega las solicitudes a los servicios correspondientes.
- **Servicio de Negocio (Business Service):** Contiene la lógica de negocio, gestionando las operaciones sobre las tareas.
- **Repositorio (Repository):** Interactúa con la base de datos para realizar operaciones CRUD sobre las tareas.
- **Base de Datos (Database):** Almacena la información de las tareas.

4.2 Componentes Principales

Interfaz de Usuario (UI):

- **Descripción:** Este componente es responsable de recibir los comandos del usuario y mostrar las respuestas correspondientes.
- **Tecnologías Utilizadas:** Java, JavaScript, CSS, React, integración con Telegram (API de Telegram).

Controlador (Controller):

- **Descripción:** Actúa como intermediario entre la UI y los servicios de negocio. Recibe los comandos de la UI, los interpreta, y llama a los servicios correspondientes.
- **Tecnologías Utilizadas:** Spring Boot (Controladores REST).

Servicio de Negocio (Business Service):

- **Descripción:** Implementa la lógica de negocio del sistema. Gestiona las operaciones sobre las tareas, como crear, actualizar, listar y eliminar.
- **Tecnologías Utilizadas:** Spring Boot (Servicios).

Repositorio (Repository):

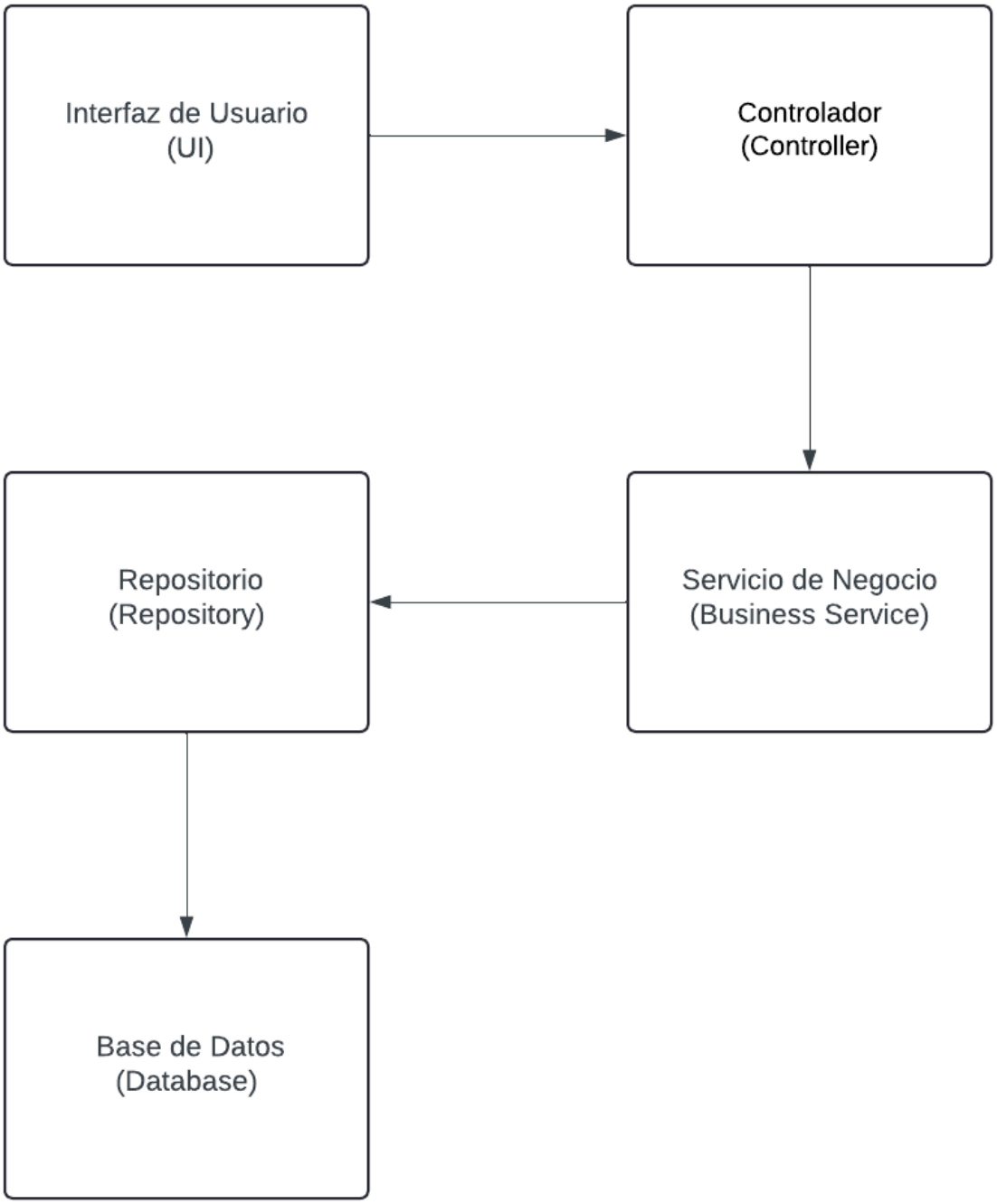
- **Descripción:** Gestiona la persistencia de datos. Interactúa con la base de datos para realizar operaciones CRUD.
- **Tecnologías Utilizadas:** Spring Data JPA, Hibernate.

Base de Datos (Database):

- **Descripción:** Almacena la información persistente del sistema, incluyendo las tareas y sus estados.
- **Tecnologías Utilizadas:** MySQL, PostgreSQL, o cualquier otro sistema de gestión de bases de datos relacionales.

4.3 Diagrama de Descomposición del Sistema

A continuación se presenta un diagrama de descomposición que muestra la estructura de los componentes del sistema y sus interacciones.


Interfaz de Usuario
(UI)
Controlador
(Controller)
Repositorio
(Repository)
Servicio de Negocio
(Business Service)
Base de Datos
(Database)

5. Diseño de los Componentes

5.1 Componente de Interfaz de Usuario

La interfaz de usuario (UI) es el punto de interacción entre el usuario y OracleBot. Este componente debe ser intuitivo y fácil de usar, permitiendo a los usuarios ingresar comandos y recibir respuestas de manera eficiente.

Diseño:

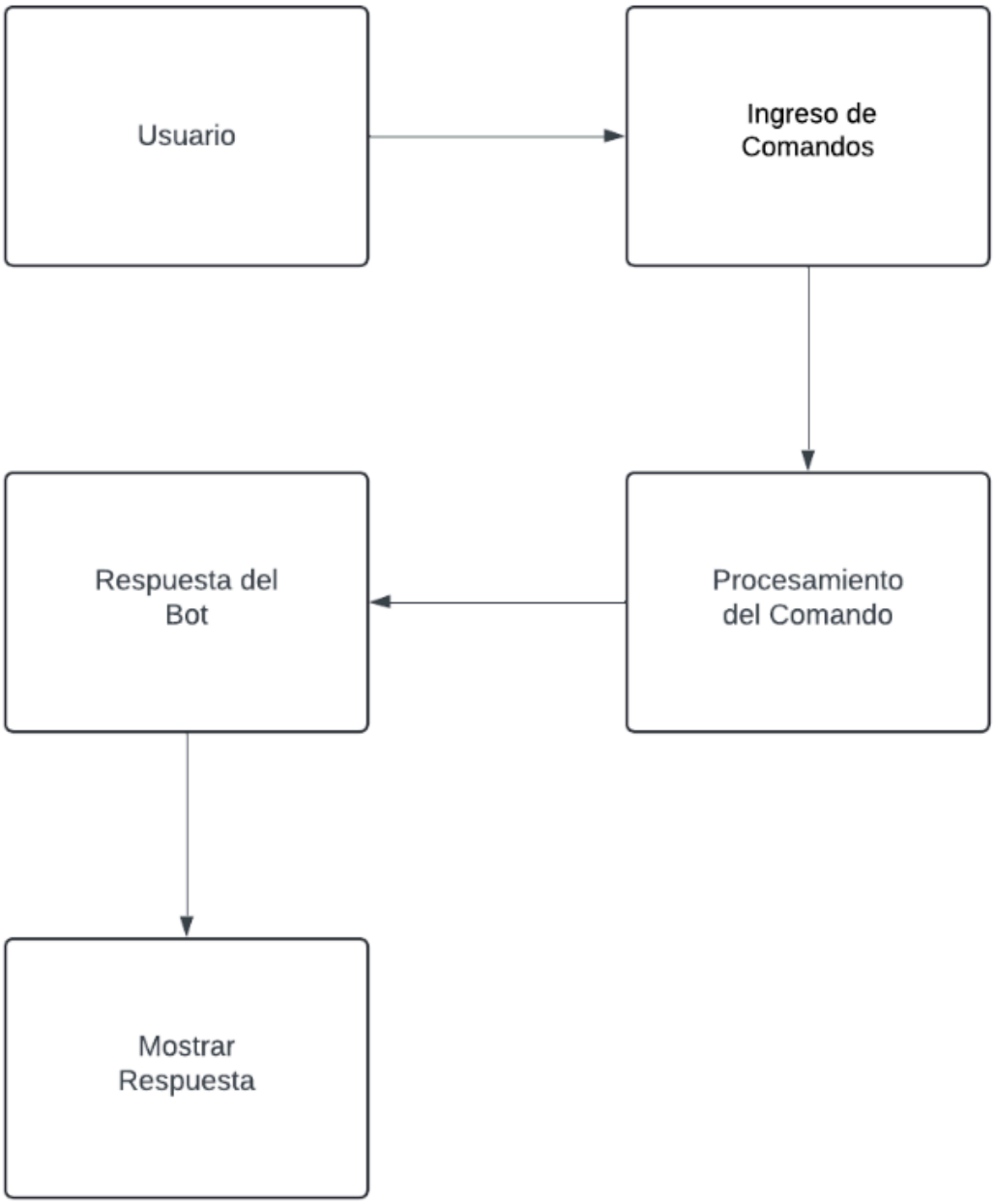
Comandos de Usuario: Los usuarios interactúan con el bot ingresando comandos específicos, estos comandos sirven entre ellos para agregar tareas, listar tareas pendientes, eliminar tareas, etc.

Respuesta del Bot: El bot procesa los comandos y devuelve respuestas claras y concisas.

Tecnologías Utilizadas:

- **Para la aplicación web:** HTML, CSS, JavaScript, y React.
- **Para integración con mensajería:** API de Telegram.

Diagrama de Flujo de UI:



5.2 Componente de Lógica de Negocio

La lógica de negocio se encarga de procesar los comandos de los usuarios y realizar las operaciones necesarias sobre las tareas.

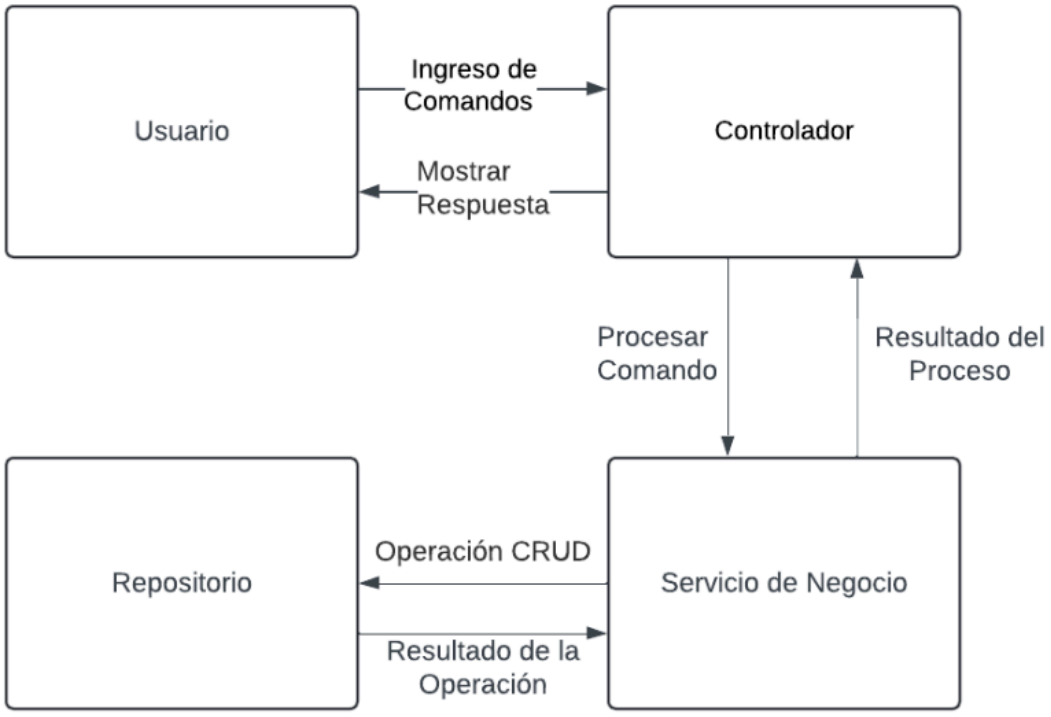
Diseño:

Operaciones Soportadas:

- **Agregar Tarea:** Crear una nueva tarea y almacenarla en la base de datos.

- **Listar Tareas Pendientes:** Recuperar y mostrar todas las tareas que no han sido completadas.
- **Eliminar Tarea:** Marcar una tarea como eliminada.

Diagrama de Secuencia de la Lógica de Negocio:



Tecnologías Utilizadas:

- **Framework:** Spring Boot
- **Patrones de Diseño:** MVC (Modelo-Vista-Controlador)

5.3 Componente de Persistencia de Datos

Este componente maneja la interacción con la base de datos, realizando operaciones CRUD (Create, Read, Update, Delete) sobre las tareas.

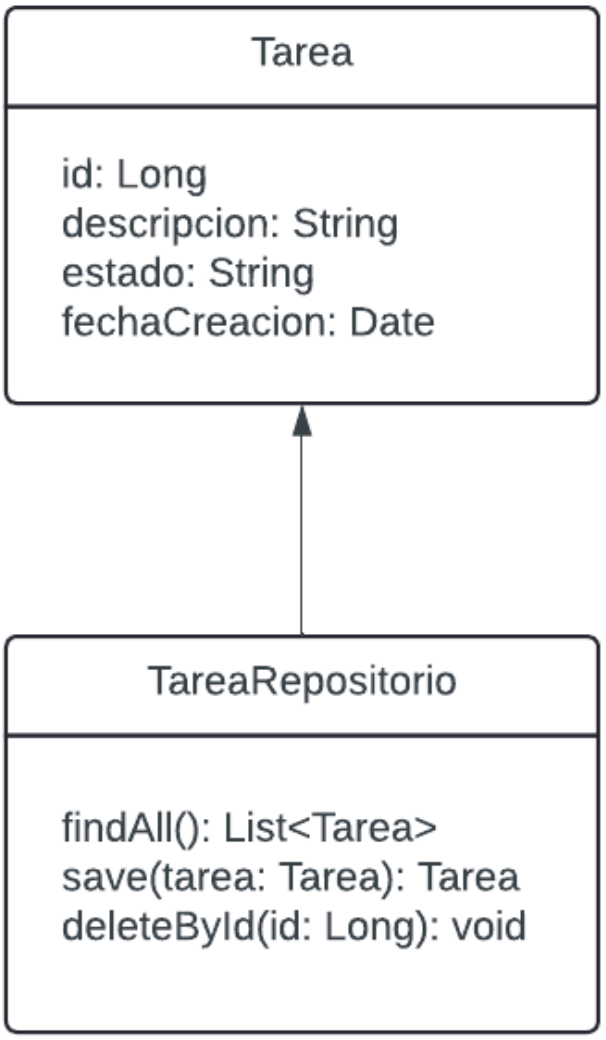
Diseño:

Entidades Principales:

- **Tarea:** Representa una tarea con atributos como id, descripcion, estado, fechaCreacion.

Repositorio: Define métodos para interactuar con la base de datos, utilizando Spring Data JPA.

Diagrama de Clases del Componente de Persistencia:



Tecnologías Utilizadas:

- **ORM:** Hibernate
- **JPA:** Spring Data JPA
- **Base de Datos:** Oracle Autonomous Database

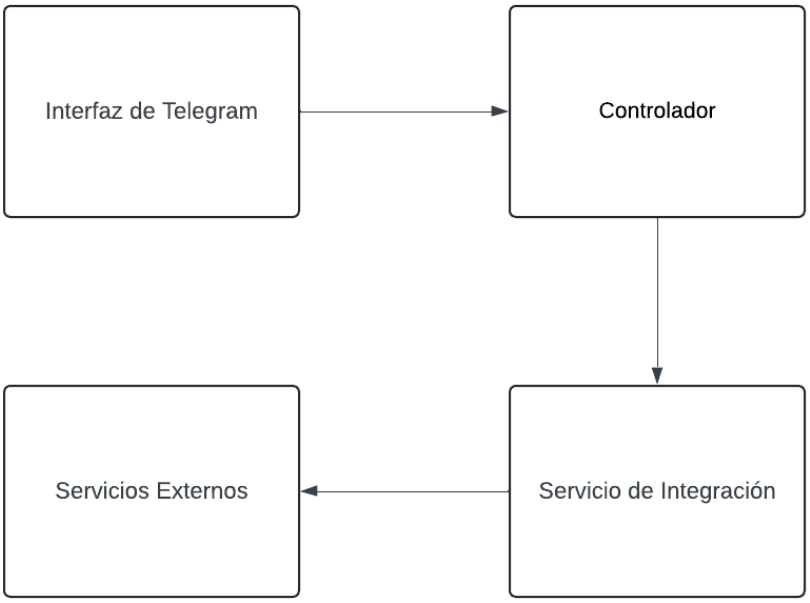
5.4 Componente de Integración

Este componente maneja la integración entre los diferentes módulos y servicios externos que OracleBot podría necesitar.

Diseño:

Integración con API de Telegram: Para recibir y enviar los comandos y mensajes de tareas a través de Telegram.

Diagrama de Interacción del Componente de Integración:



Tecnologías Utilizadas:

- Spring Boot (REST Controllers)
- APIs de Mensajería (Telegram API)

6. Diseño de la Interfaz

6.1 Interfaz de Usuario

La interfaz de usuario (UI) es el punto de interacción entre el usuario y OracleBot. El diseño de esta interfaz debe ser intuitivo, fácil de usar y permitir a los usuarios interactuar con el bot a través de comandos sencillos.

Elementos de la Interfaz de Usuario:

- **Entrada de Comandos:** Un campo de texto donde los usuarios pueden escribir sus comandos.
- **Área de Respuesta:** Una sección donde se muestran las respuestas del bot a los comandos ingresados.
- **Lista de Tareas:** Un área donde se listan las tareas pendientes.

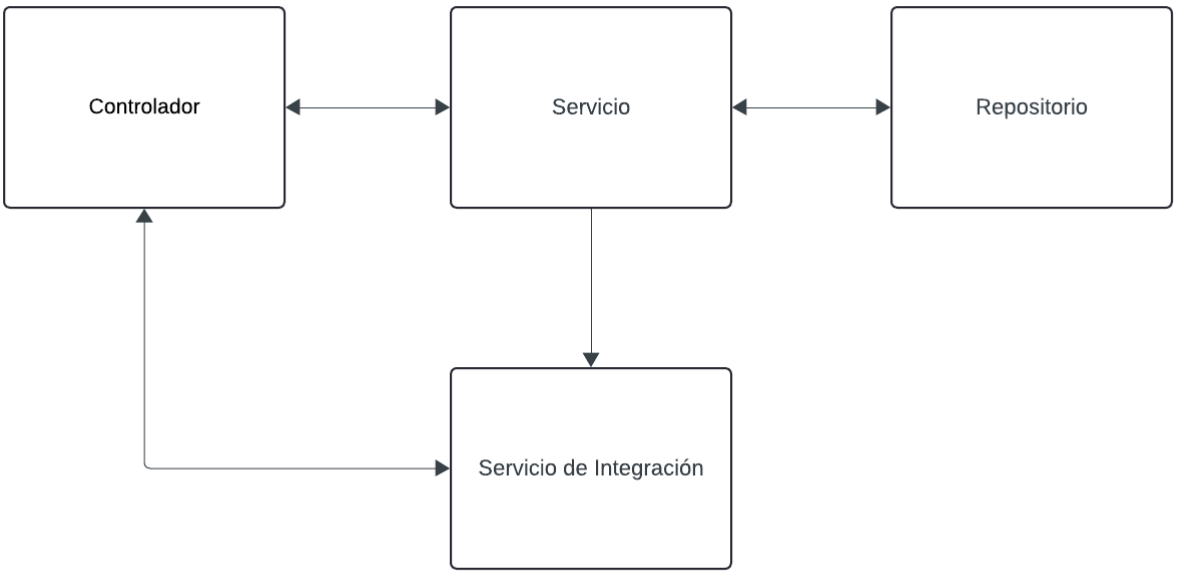
6.2 Interfaz de Componentes Internos

La interfaz de componentes internos define cómo se comunican los diferentes módulos del sistema OracleBot entre sí.

Componentes y su Interacción:

- **Controlador:** Recibe los comandos del usuario y los envía al servicio correspondiente.
- **Servicio de Negocio:** Procesa los comandos y realiza las operaciones necesarias.
- **Repositorio:** Interactúa con la base de datos para realizar operaciones CRUD.
- **Servicio de Integración:** Maneja la comunicación con servicios externos, como la API de Telegram.

Diagrama de Interacción de Componentes Internos:



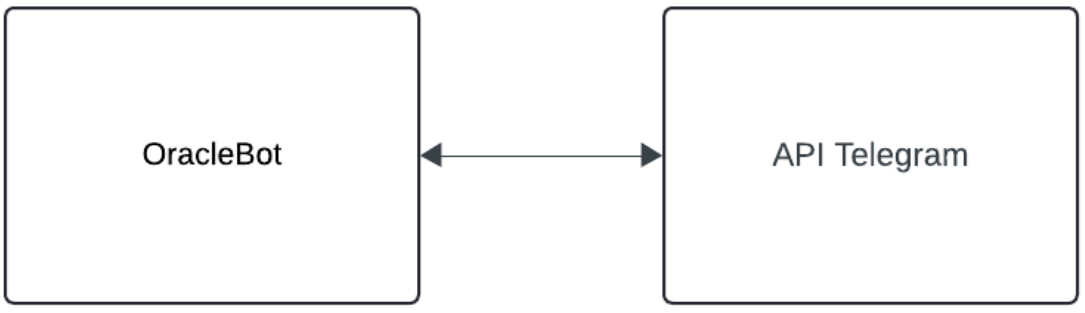
6.3 Interfaz Externa

La interfaz externa define cómo se comunica OracleBot con sistemas externos, como servicios web y APIs.

Servicios Externos:

- **API de Telegram:** Para recibir y enviar mensajes a través de Telegram.

Diagrama de Interacción con Servicios Externos:



7. Consideraciones de Diseño y Decisiones

7.1 Decisiones de Diseño

Decisión 1: Uso de Spring Boot

- **Justificación:** Spring Boot es un marco robusto y bien soportado para crear aplicaciones Java. Ofrece una configuración mínima, permite una rápida puesta en marcha, y se integra bien con otros componentes como Spring Data JPA y Spring Security.
- **Impacto:** La elección de Spring Boot facilita el desarrollo, la prueba y el despliegue de la aplicación OracleBot, asegurando una arquitectura modular y escalable.

Decisión 2: Utilización de React para la Interfaz de Usuario

- **Justificación:** React es una biblioteca popular para la construcción de interfaces de usuario interactivas. Permite la creación de componentes reutilizables y facilita la gestión del estado de la aplicación.
- **Impacto:** La elección de React mejora la experiencia del usuario con una interfaz dinámica y reactiva, además de hacer que el código sea más mantenible y escalable.

Decisión 3: Integración con la API de Telegram

- **Justificación:** Telegram ofrece una API robusta y bien documentada para crear bots. Es segura, gratuita y ampliamente utilizada.
- **Impacto:** La integración con la API de Telegram permite a OracleBot comunicarse eficientemente con los usuarios, ofreciendo un canal de mensajería confiable y seguro.

7.2 Patrones de Diseño Utilizados

Patrón 1: MVC (Modelo-Vista-Controlador)

- **Descripción:** Este patrón organiza el código en tres componentes principales: el Modelo (gestión de datos), la Vista (interfaz de usuario) y el Controlador (lógica de negocio).

- **Aplicación en OracleBot:** Spring Boot sigue el patrón MVC, con Controladores manejando las solicitudes HTTP, Servicios gestionando la lógica de negocio y Repositorios interactuando con la base de datos.

Patrón 2: Singleton

- **Descripción:** El patrón Singleton asegura que una clase tenga una única instancia y proporciona un punto global de acceso a ella.
- **Aplicación en OracleBot:** Utilizado en componentes como las configuraciones de la API de Telegram, donde se requiere una instancia única para manejar la comunicación con el servicio de mensajería.

Patrón 3: Factory

- **Descripción:** El patrón Factory crea objetos sin exponer la lógica de creación al cliente y se refiere a la interfaz común.
- **Aplicación en OracleBot:** Utilizado para crear instancias de servicios externos y componentes que requieren configuraciones complejas.

8. Pruebas y Validación

8.1 Estrategia de Pruebas

La estrategia de pruebas para OracleBot se diseñó para asegurar que todas las funcionalidades del sistema se comporten de acuerdo con los requisitos especificados. La estrategia incluye los siguientes tipos de pruebas:

Pruebas Unitarias

- **Descripción:** Se prueban individualmente las unidades de código, como métodos y funciones, para garantizar que cada una funcione correctamente.
- **Herramientas:** JUnit para pruebas en el lado del servidor (Spring Boot) y Jest para pruebas del lado del cliente (React).

Pruebas de Integración

- **Descripción:** Se prueba la integración entre diferentes componentes del sistema, como la interacción entre el backend de Spring Boot y la API de Telegram.
- **Herramientas:** Spring Boot Test y MockMVC para simular peticiones HTTP y validar la correcta integración de servicios y controladores.

Pruebas Funcionales

- **Descripción:** Se validan las funcionalidades completas del sistema en un entorno que simula el entorno de producción, asegurando que todas las funciones trabajen juntas como se espera.
- **Herramientas:** Selenium para pruebas automatizadas de la interfaz de usuario y Postman para pruebas de los endpoints del API.

Pruebas de Regresión

- **Descripción:** Se realizan para asegurar que nuevas modificaciones en el código no introduzcan errores en funcionalidades previamente probadas.

- **Herramientas:** Jenkins para la integración continua y ejecución automática de pruebas unitarias e integrales cada vez que se realiza un cambio en el código.

Pruebas de Usuario Final

- **Descripción:** Se llevan a cabo con usuarios finales para obtener retroalimentación sobre la usabilidad y funcionalidad del sistema.
- **Herramientas:** Sesiones de prueba manuales con usuarios y herramientas de seguimiento de feedback como Google Forms o UserTesting.

8.2 Plan de Validación

El plan de validación está diseñado para asegurar que OracleBot cumpla con las necesidades del usuario y los requisitos del proyecto. Incluye los siguientes pasos:

Validación de Requisitos

- **Descripción:** Verificar que todos los requisitos funcionales y no funcionales están claramente definidos y documentados.
- **Métodos:** Revisiones formales y validación con stakeholders y usuarios clave.

Pruebas de Aceptación

- **Descripción:** Realizar pruebas con escenarios del mundo real basados en los casos de uso definidos en los requisitos del sistema.
- **Métodos:** Scripts de pruebas de aceptación que cubren todas las funcionalidades críticas del sistema.

Validación de la Interfaz de Usuario

- **Descripción:** Asegurar que la interfaz de usuario es intuitiva y cumple con los estándares de usabilidad.
- **Métodos:** Pruebas de usabilidad y sesiones de feedback con usuarios reales.

Pruebas de Desempeño

- **Descripción:** Validar que el sistema puede manejar la carga esperada y operar eficientemente bajo diferentes condiciones.
- **Métodos:** Herramientas como JMeter para simular cargas y medir el rendimiento del sistema.

Revisión de Seguridad

- **Descripción:** Verificar que el sistema es seguro y protege adecuadamente los datos del usuario.
- **Métodos:** Análisis de vulnerabilidades, pruebas de penetración y revisiones de código de seguridad.

Revisión de Código

- **Descripción:** Asegurar que el código cumple con los estándares de calidad y mejores prácticas de desarrollo.
- **Métodos:** Revisiones de código entre pares y uso de herramientas como SonarQube para análisis de calidad de código.

Documentación y Feedback

- **Descripción:** Documentar todas las pruebas realizadas y obtener feedback continuo de los usuarios para mejoras continuas.
- **Métodos:** Mantener registros detallados de todas las pruebas, resultados y acciones correctivas tomadas.