Tecnológico de Monterrey

Desarrollo e implantación de sistemas de software

Grupo 106

Documento de Diseño

Profesores:

Adan Octavio Ruiz Martinez

Jorge Alvarez Bujanos

Leonardo S. Gamez Peña

Lorena Guadalupe Gómez Martínez

Alberto Emmanuel Benavides Contreras

Equipo 6:

Eugenia Uresti Arriaga A01284839

Mónica Parra Franco A01251941

Gerardo Manzur Morales A01177622

Jorge Nuñez Gurrola A00833455

Javier Banegas A00827812

13 de junio de 2024

1. Introducción

1.1 Propósito

El propósito de este documento de diseño de software (SDD) describe la arquitectura y el diseño del sistema de “Oracle Java ChatBot”. Su objetivo es proporcionar una guía detallada sobre los componentes del sistema, la estructura de datos, la interfaz de usuario y los requisitos del sistema. Está destinado a desarrolladores de software, arquitectos de sistemas, ingenieros de calidad y cualquier otro personal técnico involucrado en el desarrollo y mantenimiento del sistema.

1.2 Alcance

El producto de software que se desarrollará es el “Oracle Java ChatBot”, un servicio de chatbot diseñado para automatizar la gestión de tareas de los desarrolladores y ofrecer a los managers visibilidad sobre las tareas del equipo. Este bot funcionará en la plataforma de Telegram y se integrará con Oracle Cloud Infrastructure y Oracle Autonomous Database. Permitirá a los desarrolladores agregar, revisar y eliminar sus tareas, y brindará a los managers la capacidad de realizar estas acciones mientras visualizan las tareas de todo el equipo a su cargo.

1.3 Material de Referencia

- IEEE Std 1016-2009. IEEE Recommended Practice for Software Design Descriptions.
- Documento de Requisitos de Software SRS, basado en IEEE Std 830-1998

1.4 Definiciones y Acrónimos

- SDD: Software Design Document
- SRS: Software Requirements Specification
- OCI: Oracle Cloud Infrastructure
- JDBC: Java Database Connectivity
- UCP: Universal Connection Pool

2. Visión general del sistema

El Oracle Java ChatBot es una solución que optimiza la eficiencia y productividad de los desarrolladores al automatizar la gestión de tareas y ofrecer a los managers una

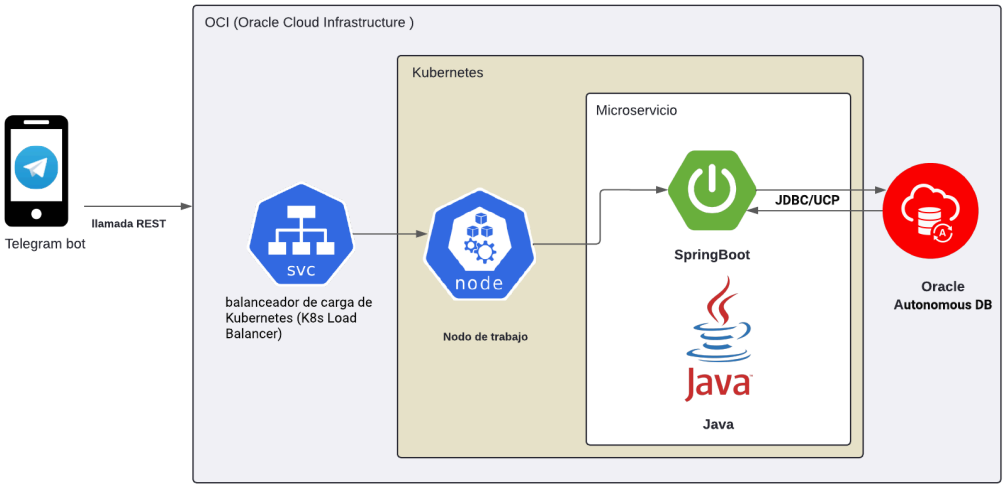
herramienta de visibilidad. Utilizando la plataforma de Telegram, el chatbot permite a los desarrolladores agregar, revisar y eliminar tareas de manera rápida y sencilla. A su vez, los managers pueden visualizar y gestionar las tareas del equipo, mejorando la coordinación y seguimiento de los proyectos. Este sistema se integra con Oracle Cloud Infrastructure para alojar sus microservicios, asegurando la escalabilidad y fiabilidad, y almacena datos en Oracle Autonomous Database. El backend, desarrollado en Spring Boot, maneja la lógica de negocio y las interacciones con la base de datos.

3. Arquitectura del Sistema

3.1 Diseño General de la Arquitectura

La arquitectura del Oracle Java ChatBot se basa en Oracle Cloud Infrastructure (OCI) para el despliegue de una solución robusta y escalable. El diseño del sistema incluye varios componentes interconectados que aseguran una comunicación eficiente y una gestión óptima de los recursos.

3.2 Diagrama de Arquitectura del Sistema



Oracle Java Chatbot

3.3 Descripción de la Descomposición

La arquitectura del sistema se descompone en los siguientes componentes.

- **Telegram Bot:** Actúa como la interfaz de usuario inicial, permitiendo la interacción con los usuarios a través de dispositivos móviles. Este componente se comunica con el backend mediante llamadas API REST.
- **K8s Load Balancer:** Asegura la distribución equitativa del tráfico entre los diferentes nodos de trabajo, garantizando la alta disponibilidad y escalabilidad del sistema.
- **Kubernetes:** Aloja los contenedores que ejecutan los microservicios del backend, proporcionando un entorno administrado que facilita la escalabilidad y la gestión de recursos.
- **Microservicio:** Desarrollado en Spring Boot, este componente maneja la lógica de negocio de la aplicación. Se conecta a la base de datos utilizando JDBC/UCP, garantizando una comunicación eficiente y segura con el sistema de gestión de base de datos.
- **Oracle Autonomous Database:** Es el sistema de gestión de datos centralizado que almacena toda la información relacionada con usuarios, tareas y roles.

3.4 Integración DevOps

Creamos una conexión externa a nuestro repositorio de Github que incluye nuestro proyecto del ChatBot, el cual hará un 'mirroring' de este en OCI.

Creamos nuestro build pipeline que contiene los siguientes pasos:

- Un 'undeploy_spec.yaml' para borrar los pods y services de kubernetes de versiones pasadas de nuestro bot.
- Un 'build_spec.yaml' que crea compila y empaqueta nuestro proyecto en un archivo JAR que después lo construye en una imagen de Docker con las especificaciones de nuestro Dockerfile, y guardamos una variable que contenga el 'tag' de la imagen.
- El siguiente paso sería guardar la imagen de Docker en un artifact, el cual es uno de los recursos que ofrece OCI DevOps.

Después creamos un deploy pipeline que contiene un solo paso:

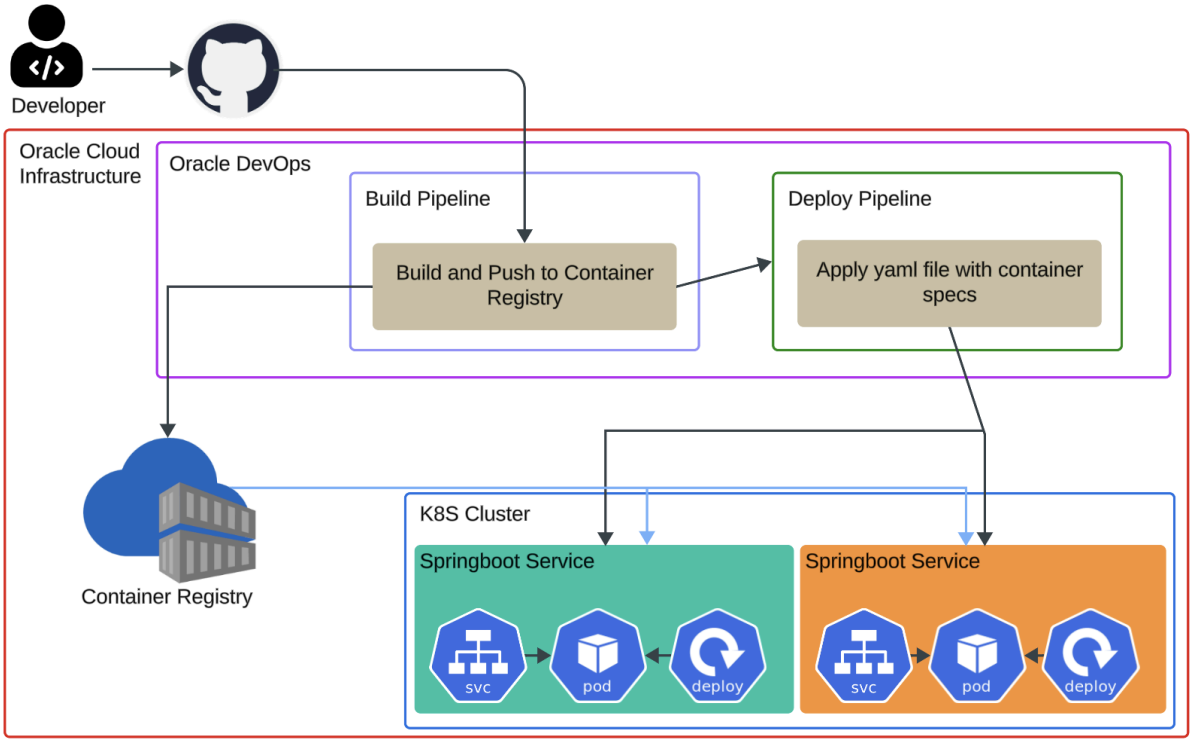
- Desplegar la imagen de docker por medio de kubernetes; para ello, creamos un artifact que contiene un kubernetes manifest por Inline; este artifact contiene, toda la información que necesita kubernetes para poder conectar el proyecto con la base de datos por JDBC, la creacion del Load Balancer en services, los pods que ejecutan el

programa, así como el path de la imagen de docker con la variable que funciona como tag.

- En el pipeline, incluimos el ambiente de OKE en el que se desplegará el proyecto y el artifact con el kubernetes manifest.

Por último, creamos un trigger que cada vez que se identifique un cambio en cualquiera de los branches del repositorio clonado en OCI, este ejecutara el build pipeline.

3.5 Diagrama de Integración DevOps



3.6 Razonamiento del Diseño

El diseño del Oracle Java Chat Bot se ha realizado considerando varios factores clave que garantizan su eficacia, escalabilidad y facilidad de mantenimiento. A continuación, se detallan los principales razonamientos detrás del diseño elegido:

- **Modularidad y Escalabilidad**: La separación clara entre el frontend (Telegram Bot), el backend (Spring Boot) y la base de datos (Oracle Autonomous Database) permite que cada componente sea desarrollado, desplegado y escalado de manera

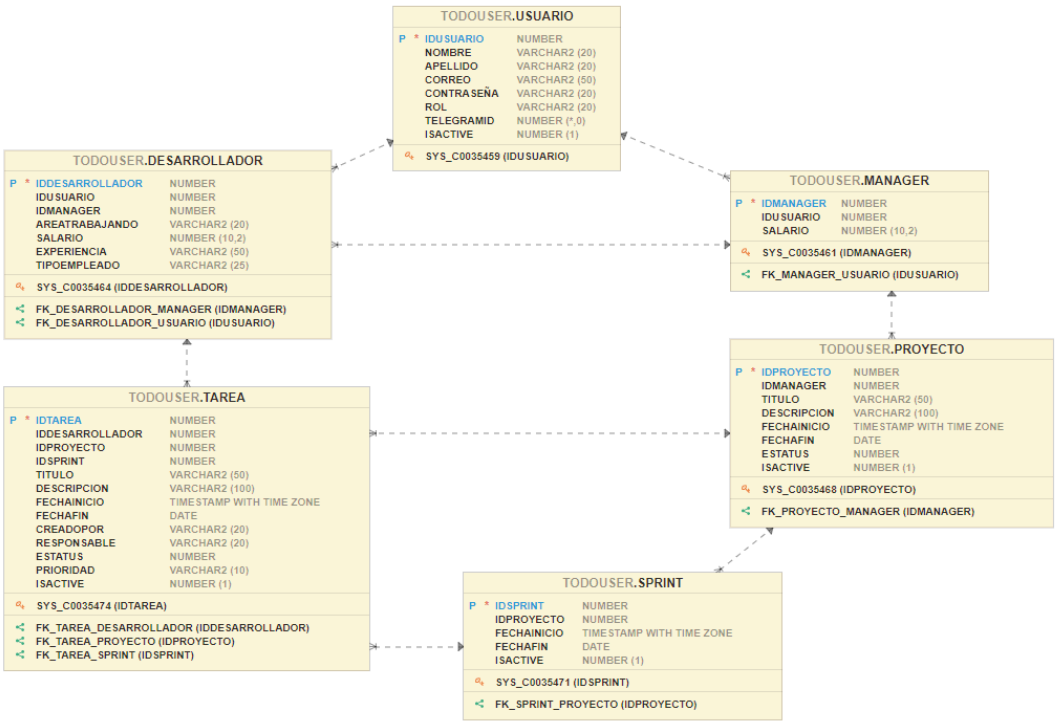
independiente. Esto asegura que el sistema pueda crecer y adaptarse a las necesidades cambiantes sin afectar negativamente otros componentes.

- **Uso de Tecnologías Modernas**: La elección de Spring Boot para el backend proporciona un marco robusto y flexible para desarrollar aplicaciones Java de manera rápida y eficiente. El uso de Kubernetes para la orquestación de contenedores garantiza que el sistema pueda manejar una carga variable de manera eficiente, escalando automáticamente según sea necesario.
- **Alta Disponibilidad y Resiliencia**: Implementar la infraestructura en Oracle Cloud Infrastructure (OCI) proporciona un entorno seguro y altamente disponible para la ejecución de la aplicación. El uso de Kubernetes asegura que los contenedores se mantengan operativos y se reinicien automáticamente en caso de fallos, mejorando la resiliencia del sistema.
- **Facilidad de Mantenimiento y Actualización**: La integración de prácticas DevOps, como la integración continua (CI) y el despliegue continuo (CD), permite que los desarrolladores realicen cambios y despliegues de manera ágil y con menos riesgo de introducir errores.
- **Seguridad:** La arquitectura asegura que los datos se mantengan seguros mediante el uso de conexiones seguras y prácticas recomendadas de seguridad en la nube. Se emplean certificados SSL y políticas de seguridad en la configuración de Kubernetes y los servicios de OCI.

4. Diseño de Datos

4.1 Descripción de Datos

La base de datos del Oracle Java ChatBot está diseñada para manejar y almacenar eficientemente la información crítica relacionada con usuarios, tareas, proyectos, sprints y roles. La estructura de la base de datos está organizada en varias tablas interrelacionadas mediante llaves primarias y foráneas, garantizando la integridad referencial de los datos.



La base de datos sigue un modelo relacional, donde cada entidad se representa con una tabla con sus respectivos atributos.

- **Usuarios:** Contiene información básica de todos los usuarios del sistema, independientemente de su rol.
- **Desarrolladores**: Contiene detalles específicos de los desarrolladores, incluyendo su experiencia, área de trabajo y salario.
- **Managers:** Contiene detalles específicos de los managers, como su salario y proyectos que gestionan.
- **Proyectos:** Contiene información sobre los proyectos, incluyendo el manager responsable, el título y las fechas de inicio y fin.
- **Tareas:** Contiene información de las tareas asignadas a los desarrollados, incluyendo el proyecto y sprint al que pertenecen
- **Sprints:** Contiene información sobre los sprints dentro de los proyectos

4.2 Diccionario de Datos

- Usuarios (TODOUSER.USUARIO)
    - **IdUsuario:** Identificador único del usuario (Primary Key)
    - **Nombre**: Nombre del usuario
    - **Apellido:** Apellido del usuario
    - **Correo:** Correo electrónico del usuario

    - **Contraseña:** Contraseña del usuario
    - **Rol**: Rol del usuario (desarrollador o manager)
    - **TelegramId**: ID del usuario en Telegram
    - **IsActive**: Indicador de si el usuario está activo (1 para activo, 0 para inactivo)

- Desarrolladores (TODOUSER.DESARROLLADOR)
    - **IdDesarrollador**: Identificador único del desarrollador (Primary Key)
    - **IdUsuario**: Identificador del usuario asociado (Foreign Key)
    - **IdManager**: Identificador del manager al que reporta (Foreign Key)
    - **AreaTrabajando**: Área en la que está trabajando
    - **Salario**: Salario del desarrollador
    - **Experiencia**: Años de experiencia del desarrollador
    - **TipoEmpleado**: Tipo de empleado (tiempo completo o medio tiempo)

- Mánagers (TODOUSER.MANAGER)
    - **IdManager**: Identificador único del mánager (Primary Key)
    - **IdUsuario**: Identificador del usuario asociado (Foreign Key)
    - **Salario**: Salario del manager.

- Proyectos (TODOUSER.PROYECTO)
    - **IdProyecto**: Identificador único del proyecto (Primary Key)
    - **IdManager**: Identificador del manager a cargo del proyecto (Foreign Key)
    - **Titulo**: Título del proyecto
    - **Descripcion**: Descripción del proyecto
    - **FechaInicio**: Fecha de inicio del proyecto
    - **FechaFin**: Fecha de finalización del proyecto
    - **Estatus**: Estado del proyecto (0-100)
    - **IsActive**: Indicador de si el proyecto está activo (1 para activo, 0 para inactivo)

- Tareas (TODOUSER.TAREA)
    - **IdTarea**: Identificador único de la tarea (Primary Key)
    - **IdDesarrollador**: Identificador del desarrollador asignado a la tarea (Foreign Key)
    - **IdProyecto**: Identificador del proyecto al que pertenece la tarea (Foreign Key)
    - **IdSprint**: Identificador del sprint al que pertenece la tarea (Foreign Key)

    - **Titulo**: Título de la tarea
    - **Descripcion**: Descripción de la tarea
    - **FechaInicio**: Fecha de inicio de la tarea
    - **FechaFin**: Fecha de finalización de la tarea
    - **Creador**: Usuario que creó la tarea
    - **Responsable**: Usuario responsable de la tarea
    - **Estatus**: Estado de la tarea (0-100)
    - **Prioridad**: Prioridad de la tarea (alta, media, baja)
    - **IsActive**: Indicador de si la tarea está activa (1 para activa, 0 para inactiva)

- Sprints (TODOUSER.SPRINT)
    - **IdSprint**: Identificador único del sprint (Primary Key)
    - **IdProyecto**: Identificador del proyecto al que pertenece el sprint (Foreign Key)
    - **FechaInicio**: Fecha de inicio del sprint
    - **FechaFin**: Fecha de finalización del sprint
    - **IsActive**: Indicador de si el sprint está activo (1 para activo, 0 para inactivo)

5. Diseño de Componentes

El diseño de componentes del Oracle Java ChatBot detalla cómo cada parte del sistema interactúa y contribuye al funcionamiento general. El Telegram Bot actúa como la interfaz principal de usuario, permitiendo la interacción a través de dispositivos móviles mediante el envío y recepción de mensajes de texto. Este bot se comunica con el backend a través de llamadas API REST, permitiendo a los usuarios agregar, modificar y eliminar tareas.

El Kubernetes Load Balancer garantiza la distribución equitativa del tráfico, asegurando la alta disponibilidad y escalabilidad del sistema al distribuir el tráfico de red de manera uniforme entre los nodos de trabajo. Kubernetes, por su parte, aloja los contenedores de los microservicios, proporcionando un entorno gestionado que facilita la escalabilidad y la gestión de recursos, administrando el ciclo de vida de los contenedores y garantizando que los servicios estén siempre disponibles.

El microservicio Spring Boot maneja la lógica de negocio del sistema. Desarrollado en Spring Boot, este componente procesa las solicitudes del frontend y realiza operaciones en la base de datos. Se comunica con la base de datos utilizando JDBC/UCP, asegurando una comunicación eficiente y segura. Finalmente, Oracle Autonomous Database almacena los

datos del sistema. Este sistema de gestión de base de datos centralizado almacena toda la información relacionada con usuarios, tareas y roles, proporcionando funcionalidades de autoservicio, rendimiento optimizado y seguridad avanzada.

El flujo de trabajo del sistema comienza con la interacción del usuario, donde los usuarios interactúan con el Telegram Bot ingresando comandos y solicitudes. El bot envía estas solicitudes al backend a través de API REST, y el Kubernetes Load Balancer distribuye el tráfico a los nodos disponibles para garantizar que el sistema maneje la carga eficientemente. Los microservicios Spring Boot procesan las solicitudes, ejecutan la lógica de negocio y se comunican con la base de datos. Oracle Autonomous Database maneja el almacenamiento y la recuperación de datos, asegurando que la información esté siempre disponible y segura. Finalmente, el backend envía la respuesta de vuelta al Telegram Bot, que luego presenta la información al usuario.

Esta arquitectura modular y escalable permite que el Oracle Java ChatBot funcione de manera eficiente, manejando múltiples solicitudes simultáneamente y garantizando la disponibilidad del sistema en todo momento.

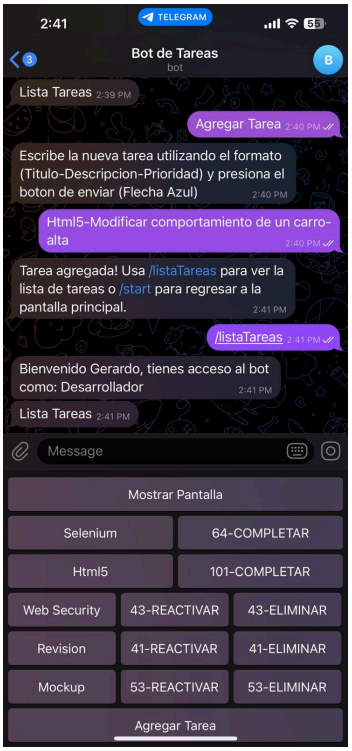
6. Diseño de Interfaz

6.1 Visión General de la Interfaz de Usuario

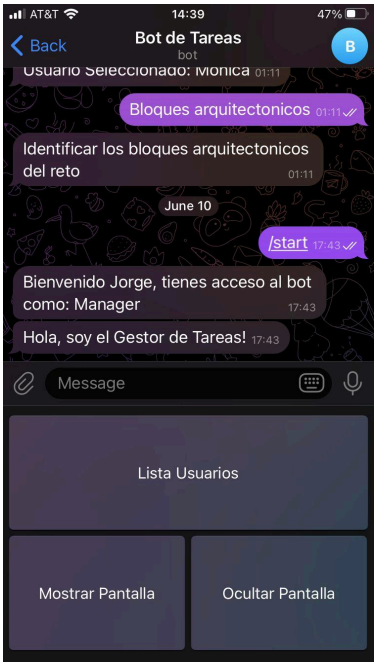
Al entrar a la aplicación, los usuarios verán la interfaz según el rol asignado, que puede ser manager o desarrollador. Ambas interfaces contienen funcionalidades similares, pero varían en ciertas capacidades. Los desarrolladores pueden agregar, modificar y eliminar tareas asignadas o creadas por ellos mismos. Los managers, en cambio, pueden ver, editar, agregar y eliminar tareas de todos los miembros de su equipo. La interfaz se inicia mediante el comando **/start** en el chat de Telegram.

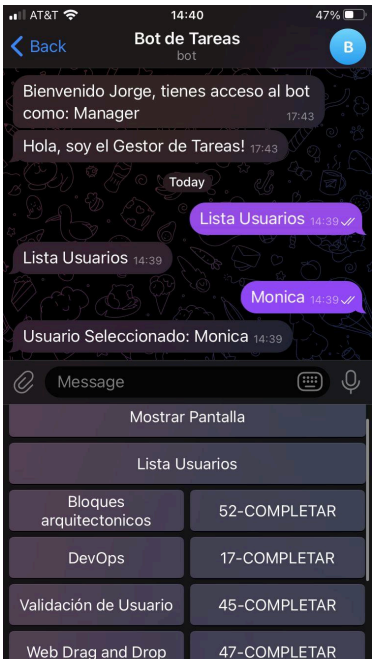
6.2 Imágenes de Pantallas

- Vista de Desarrollador



- Vista Manager





6.3 Objetos y Acciones de Pantallas

La interfaz del Oracle Java ChatBot incluye varios objetos y acciones que permiten a los usuarios interactuar de manera eficiente con el sistema.

Objetos

1. Campo de Texto: Permite al usuario ingresar comandos y solicitudes.
2. Botones de Acción: Incluyen botones para agregar, eliminar, completar, reactivar y listar tareas.
3. Listas de Tareas: Muestran las tareas pendientes y completadas.
4. Campos de Detalle de Tareas: Muestran información detallada sobre cada tarea, incluyendo título, descripción y prioridad.

Acciones

1. Agregar Tarea: Los usuarios pueden ingresar detalles de una nueva tarea, como título, descripción y prioridad.
2. Eliminar Tarea: Permite a los usuarios eliminar tareas específicas.
3. Completar Tarea: Marca una tarea como completada.
4. Reactivar Tarea: Permite volver a activar una tarea previamente completada.
5. Visualizar Tareas: Los usuarios pueden ver una lista de tareas pendientes y completadas.
6. Ver Lista de Usuarios (Manager): Los managers pueden ver y gestionar la lista de usuarios de su equipo.

7. Matriz de Requerimientos

7.1 Requerimientos funcionales

ID	Descripción	Componentes Relacionados	Estado
RF-01	El sistema debe permitir a los usuarios iniciar sesión.	Telegram Bot, Spring Boot Backend, Oracle Autonomous Database	Implementado

RF-02	Se muestran mensajes claros sobre el estado de inicio de sesión.	Telegram Bot, Spring Boot Backend	Implementado
RF-03	Incluir el campo de texto en la pantalla principal para ingresar comandos.	Telegram Bot, Spring Boot Backend	Implementado
RF-04	Incluir un botón para enviar el comando escrito en el campo de texto.	Telegram Bot, Spring Boot Backend	Implementado
RF-05	Al mostrar el listado de tareas, se muestra en la parte superior las tareas sin completar y en la parte inferior se muestran las tareas ya completadas.	Telegram Bot, Spring Boot Backend, Oracle Autonomous Database	Implementado
RF-06	Permitir agregar nuevas tareas.	Telegram Bot, Spring Boot Backend, Oracle Autonomous Database	Implementado
RF-07	Permitir eliminar tareas del listado de tareas.	Telegram Bot, Spring Boot Backend, Oracle Autonomous Database	Implementado
RF-08	Permitir marcar como completada una tarea.	Telegram Bot, Spring Boot Backend, Oracle Autonomous Database	Implementado
RF-09	Permitir reactivar una tarea ya completada, es decir regresar	Telegram Bot, Spring Boot	Implementado

	tarea a no completada.	Backend, Oracle Autonomous Database	
RF-10	Botón que te permite modificar alguno de los campos de una tarea.	Telegram Bot, Spring Boot Backend, Oracle Autonomous Database	Pendiente
RF-11	Después de cada comando se muestran mensajes con atajos/sugerencias para el siguiente comando.	Telegram Bot, Spring Boot Backend	Implementado
RF-12	Botón que te permite regresar a la pantalla principal	Telegram Bot	Implementado
RF-13	Si el usuario llega a insertar comandos incorrectos, el bot responde con ejemplos de comandos disponibles.	Telegram Bot	Pendiente
RF-14	Opciones de filtro que ayudan a los usuarios a ver tareas específicas.	Telegram Bot, Spring Boot Backend, Oracle Autonomous Database	Pendiente
RF-15	Botón que enlista todas las tareas de todo el equipo bajo su responsabilidad. (Manager)	Telegram Bot, Spring Boot Backend, Oracle Autonomous Database	Implementado
RF-16	Al agregar una nueva tarea el bot pide diferentes campos a llenar, como título, descripción y	Telegram Bot, Spring Boot Backend	Implementado

	prioridad.		
RF-17	Botón que enlista todas las tareas asignadas al usuario. (Desarrollador)	Telegram Bot, Spring Boot Backend, Oracle Autonomous Database	Implementado

7.2 Requerimientos no funcionales

ID	Descripción	Componentes Relacionados	Estado
RNF-01	El sistema debe de ser intuitivo y fácil de usar	Telegram Bot, Spring Boot Backend, Frontend UI	Implementado
RNF-02	Debe de existir una conexión con Oracle Autonomous Database donde se guardará la información de los usuarios	Spring Boot Backend, Oracle Autonomous Database	Implementado
RNF-03	Toda la información guardada en la base de datos debe ser encriptada	Oracle Autonomous Database	Pendiente
RNF-04	El sistema debe de ser capaz de manejar simultáneamente al menos a 60 usuarios	Kubernetes, K8s Load Balancer	Implementado
RNF-05	El sistema debe de estar disponible las 24 horas del dia	Kubernetes, K8s Load Balancer	Implementado
RNF-06	Toda la comunicación entre los usuarios y el servicio deben de ser a través de Telegram	Telegram Bot	Implementado

RNF-07	El sistema debe de ser una aplicación nativa de la nube, utilizando Oracle Cloud Infrastructure, Oracle Autonomous Database y Java.	Oracle Cloud Infrastructure, Oracle Autonomous Database, Spring Boot Backend, Java	Implementado