Documentación Agenda de Contactos

Mónica Martín Rojas

Introducción

Descripción del Proyecto

El proyecto consiste en el desarrollo de una aplicación de agenda de contactos que permite a los usuarios gestionar sus contactos de manera eficiente. Lo que ofrece la aplicación son funciones para crear, actualizar, eliminar y visualizar contactos.

Propósito

El propósito de la aplicación de agenda de contactos es proporcionar a los usuarios una herramienta intuitiva y fácil de usar para gestionar sus contactos de forma centralizada. La aplicación busca simplificar la gestión de la información de contacto, y permite realizar acciones como añadir nuevos contactos, actualizar información existente o eliminar contactos.

Objetivo

Implementar funciones de creación, actualización, eliminación y visualización de contactos para cubrir las necesidades básicas de gestión de contactos.

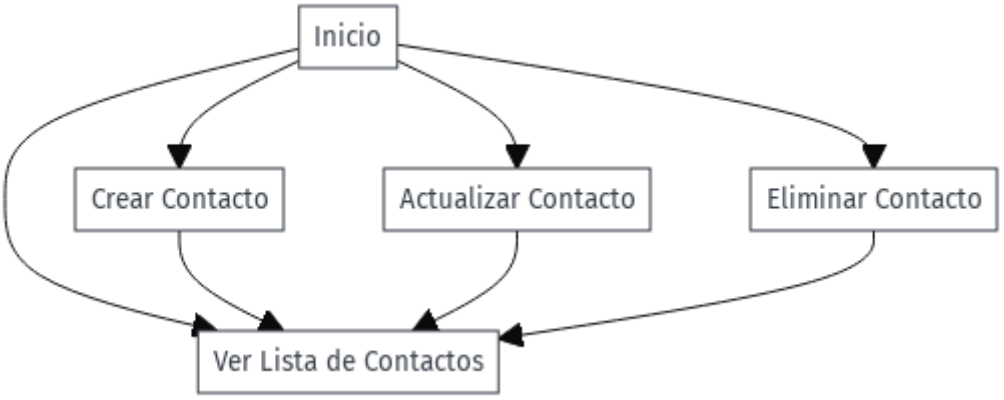
Alcance

El alcance del proyecto incluye el desarrollo de la aplicación de agenda de contactos, así como la implementación de las funciones básicas de gestión de contactos. Esto abarca la implementación de la lógica de negocio y la realización de pruebas para garantizar la calidad y fiabilidad del software.

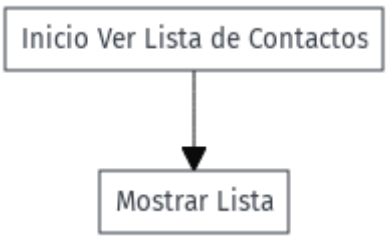
Flujo de la aplicación

Diagramas de Flujo del Proceso de Gestión de Contactos

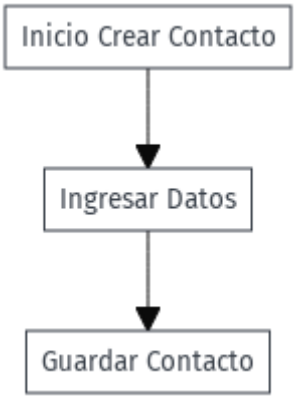
El siguiente diagrama de flujo representa el proceso general de gestión de contactos dentro de la aplicación:



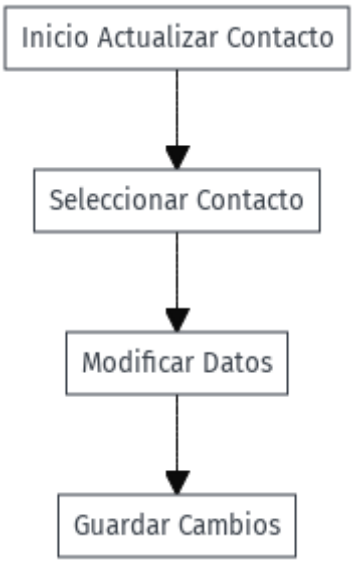
- **Inicio:** El usuario inicia sesión en la aplicación de agenda de contactos.
- **Ver Lista de Contactos:** El usuario accede a la lista de contactos almacenados en la base de datos.



- **Crear Contacto:** El usuario tiene la opción de crear un nuevo contacto proporcionando la información requerida, como nombre, apellidos y correo electrónico.



- **Actualizar Contacto:** Si el usuario desea actualizar la información de un contacto existente, puede seleccionar el contacto deseado y modificar sus detalles.



- **Eliminar Contacto:** En caso de que un contacto ya no sea relevante, el usuario puede optar por eliminarlo de la lista de contactos.

Estos diagramas de flujos proporcionan una visión general del proceso de gestión de contactos dentro de la aplicación, mostrando cómo los usuarios interactúan con las funciones de creación, actualización y eliminación de contactos, así como la muestra de la lista de contactos.

Arquitectura del sistema

La aplicación de Agenda de Contactos sigue una arquitectura de tres capas, que incluye la capa de presentación, la capa de lógica de negocio y la capa de acceso a datos. A continuación, se describe cada una de estas capas:

Capa de Presentación:

- Esta capa es responsable de interactuar directamente con los usuarios finales.
- Incluye el controlador REST API implementado en Spring Boot, que gestiona las solicitudes HTTP y las respuestas.

Capa de Lógica de Negocio:

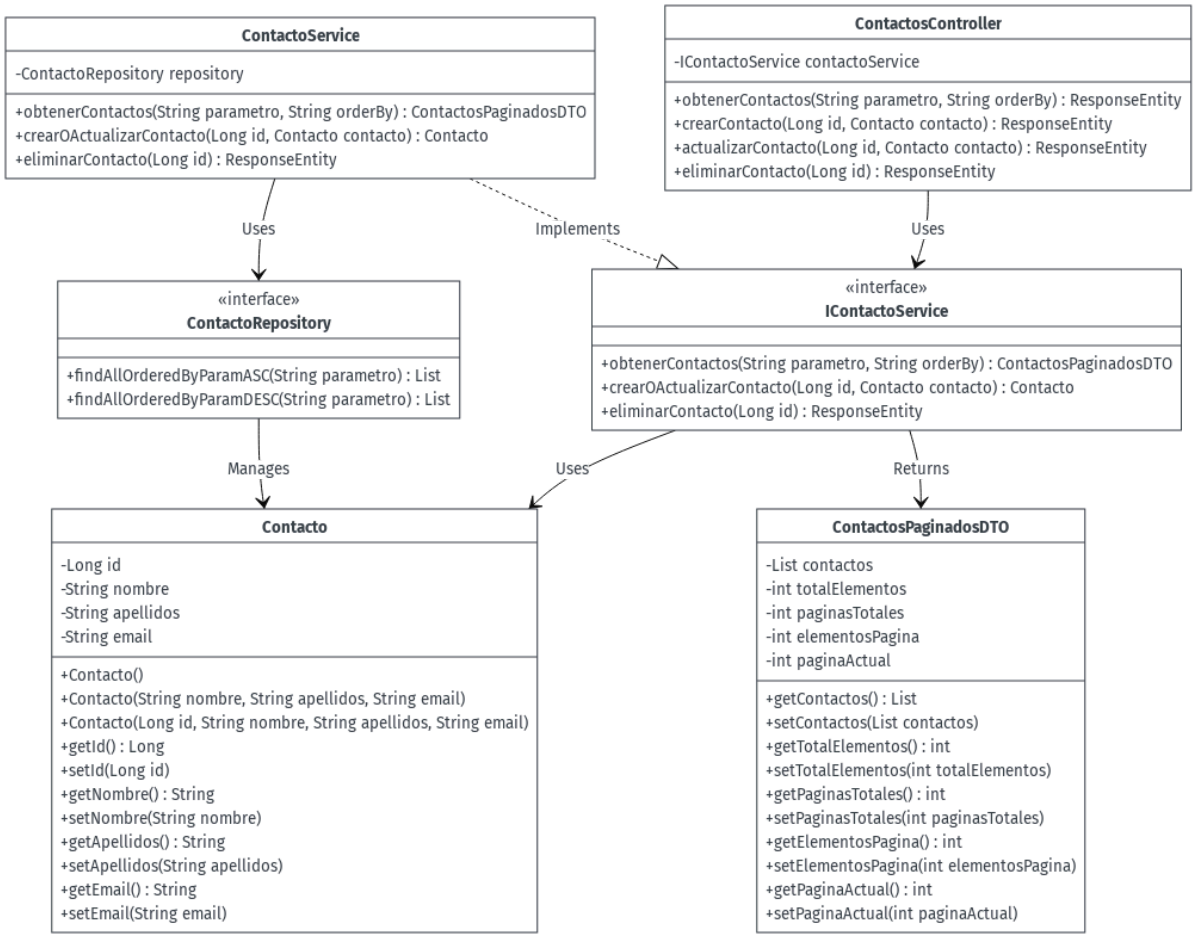
- La capa de lógica de negocio contiene la funcionalidad central de la aplicación, que coordina las operaciones y procesos relacionados con la agenda de contactos.
- Incluye servicios como la creación, actualización, eliminación y recuperación de contactos.
- Estos servicios encapsulan la lógica funcional y se encargan de validar los datos, realizar operaciones en la base de datos y devolver los resultados apropiados.

Capa de Acceso a Datos:

- La capa de acceso a datos se encarga de interactuar con la base de datos subyacente para almacenar y recuperar la información de los contactos.
- Incluye el repositorio de datos que proporciona métodos para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en la base de datos.
- El repositorio utiliza JPA para simplificar las operaciones de persistencia y consultar la base de datos.

La arquitectura de tres capas garantiza una separación de funcionalidades y responsabilidades. Además, el uso de Spring Boot y JPA facilita el desarrollo, la implementación y el mantenimiento de la aplicación de Agenda de contactos.

Diagrama de clases



Procedimientos de Pruebas

Pruebas de Funcionalidad:

- Comprobar que los contactos se muestran correctamente en la lista, ordenados según el criterio seleccionado por el usuario. Se probará cada opción de ordenamiento (ascendente y descendente) para todos los campos de los contactos (id, nombre, apellidos, email).
- Probar que se puedan crear nuevos contactos con la información correcta, asegurándose de que se guarden correctamente en la base de datos y se muestren en la lista de contactos.
- Verificar que los contactos que ya existen puedan ser actualizados correctamente con la nueva información.
- Comprobar que los contactos puedan ser eliminados de la lista sin errores.

Pruebas de Integración:

- Probar la integración entre el controlador, el servicio y el repositorio para garantizar que la lógica de negocio funcione correctamente en todas las capas de la aplicación.
- Verificar que los métodos de acceso a datos funcionen correctamente al interactuar con la base de datos, incluyendo la lectura, escritura, actualización y eliminación de contactos.
- Asegurarse de que los servicios se comuniquen de manera válida con el controlador y devuelvan respuestas correctas a las solicitudes HTTP.

Pruebas Unitarias (JUnit):

- Verificar que cada método en las clases de servicio y controlador funcione correctamente y produzca los resultados esperados.
- Probar las funcionalidades de ordenamiento para garantizar que devuelvan los resultados correctos según los parámetros proporcionados.
- Validar que los métodos de creación, actualización y eliminación de contactos se comporten como se espera.

Estos procedimientos de pruebas aseguran que la aplicación funcione correctamente, cumpla con los requisitos del cliente y proporciona una experiencia de usuario correcta. Además, las pruebas unitarias garantizan la estabilidad y la calidad del código, y el correcto funcionamiento del sistema en su conjunto.

Requisitos Funcionales

Gestión de Contactos:

- La API debe proporcionar una lista de contactos.
- La API debe permitir la creación de nuevos contactos.
- La API debe permitir la actualización de información de contactos existentes.
- La API debe permitir la eliminación de contactos existentes.

Casos de Prueba:

Obtención de Contactos:

- **Requisito Funcional**: La API debe proporcionar una lista de contactos.
- **Caso de Prueba**: Verificar que se devuelva una lista de contactos con el tamaño de página correcto y la información adecuada.
- **Caso de Prueba**: Comprobar que los contactos se ordenen correctamente según el criterio especificado por el usuario.

Creación de Contactos:

- **Requisito Funcional:** La API debe permitir la creación de nuevos contactos.
- **Caso de Prueba:** Verificar que se pueda crear un nuevo contacto proporcionando la información necesaria.
- **Caso de Prueba:** Comprobar que se devuelva el contacto creado con su id y el contacto creado después de una solicitud de creación válida.
- **Caso de Prueba**: Validar que no se pueda crear un contacto sin proporcionar información obligatoria y que se reciba el código HTTP de error adecuado.

Actualización de Contactos:

- **Requisito Funcional:** La API debe permitir la actualización de información de contactos existentes.
- **Caso de prueba**: Verificar que se pueda actualizar la información de un contacto existente proporcionando los datos actualizados.
- **Caso de prueba**: Comprobar que se devuelva el contacto actualizado después de una solicitud de actualización válida.
- **Caso de prueba**: Validar que no se pueda actualizar un contacto inexistente y que se reciba el código HTTP de error adecuado.

Eliminación de Contactos:

- **Requisito Funcional**: La API debe permitir la eliminación de contactos existentes.
- **Caso de prueba**: Verificar que se pueda eliminar un contacto existente proporcionando su ID.
- **Caso de prueba**: Comprobar que se devuelva el código HTTP de éxito después de una solicitud de eliminación válida.
- **Caso de prueba**: Validar que no se pueda eliminar un contacto inexistente y que se reciba el código HTTP de error adecuado.