Informe de Solución de Arquitectura para Generación de Pólizas de Faltantes y Gestión de Inventarios

FRONTEND
Para el front-end se siguió el patrón MVVM (Model-View-ViewModel) para la interacción de la interfaz del usuario con los modelos asignados a cada vista del sistema empleados por los ViewModels para su despliegue en la lógica de presentación. Como auxiliar se emplea *Angular Material* para el uso de componentes predefinidos.

Componentes
Listado de pólizas
Descripción: Esta pantalla muestra un elemento tabular paginado por 2 registros donde se encuentra la información de las pólizas que han sido generadas que a su vez para mayor comodidad están ordenadas de forma descendente mediante su fecha de creación.
También cuenta con un buscador que permite explorar elementos que coincidan con el dato de “Nombre de empleado” y “SKU”.

Caracteristicas
Para su construcción se empleó HTML para la construcción de las vistas y SCSS para el manejo de los estilos de la lista. Para el renderizado se emplearon directivas de la biblioteca de *Angular Material* para la generación de elementos visuales como:
data-source - para definir un conjunto de resultados y generar una tabla dinámica.
matColumnDef - para definir el un contenedor para una columna en particular
matHeaderCellDef - para definir el contenido que tendrá la cabecera de la columna objetivo.
matCellDef - para definir el contenido de todas las celdas que tendrá la columna objetivo.
matHeaderRowDef - para definir los identificadores de las columnas así como la cantidad de las mismas
matRowDef - para definir el número de renglones que tendrá la tabla

Acciones
Buscar elementos
Descripción: Realiza una búsqueda de la información en base al criterio proporcionado. Útil para buscar pólizas por el nombre del empleado que generó o por el SKU del inventario afectado.
Características: Se ejecuta una función que rescata el valor proporcionado en el buscador y al presionar la tecla “Enter” o el botón de búsqueda utiliza el paginado actual para ajustar los resultados.

Navegar entre resultados
Descripción: Realiza un cambio de “página” para obtener el siguiente bloque de información disponible (si existe más de un bloque).
Características: Se ejecuta una función incrementa la página actual y efectúa una petición que intenta recuperar el siguiente bloque de información, una vez terminado renderiza la información obtenida.

Generar póliza
Descripción: Realiza una redirección hacia la vista que permite dar de alta polizas .
Características: Se ejecuta una función emplea el componente de Angular “Router” para redirigir a la ruta o página que se encarga de registrar las pólizas.

Desactivar póliza

Descripción: Desactiva una póliza en particular, tras su desactivar la cantidad definida es restaurar al inventario correspondiente, cuando termina consulta de nuevo el listado para obtener las más actualizadas.
Características: A cada registro de la lista se le proporcionó un botón con un icono de basura que al darle click utilizar con controlador de Angular llamado "AlertController" para generar un cuadro de diálogo para dar una advertencia de eliminar con el objetivo de que el usuario decline o acepte. Tras confirmarse la pantalla se refresca con la información actualizada.

Actualizar Empleado
Descripción: Genera un modal que da la oportunidad al usuario de modificar los datos del empleado asignado.
Características: A cada registro de la lista se le proporcionó un botón con un icono de persona que al darle click utilizar con controlador de Angular llamado "ModalController" para generar un cuadro de diálogo que pueda modificar los datos del empleado, tras la modificación se recarga nuevamente la interfaz para obtener los datos actualizados.

Registro de pólizas
Descripción: Esta pantalla presenta un formulario donde se deben ingresar datos del empleado a asignar, así como el sku del inventario a afectar y posteriormente la cantidad que ha de restarse con el objetivo de permitir dar de alta pólizas.

Caracteristicas
Para su construcción se empleó HTML para la construcción de las vistas y SCSS para el manejo de los estilos de la lista. Para el renderizado se emplearon directivas propias de Angular, como
formGroup- Define una estructura de formulario para su flexible tratamiento.
ngSubmit- Evento para detectar cuando se le ha efectuado un "Submit" al formulario
formControlName- Permite asignar de forma textual el control del formulario que se encargara ese elemento.
ngFor- Permite iterar una lista y generar de forma dinámica elemento con muy poca complejidad
ngIf- Permite asignar condicionantes para determinar si un elemento será visible o no

Así como otras directivas y elementos visuales de la biblioteca de *Angular Material* como:
matInput - Directiva que permite que un elemento "Input" obtenga las propiedades de uno de los del la biblioteca.
mat-form-field- Contendor que estiliza un elemento "Input" y que le brinda propiedades flexibles.
mat-label - Elemento de apoyo para anidar etiquetas asignadas a "inputs".
mat-error- Contedor asignador para definir mensajes de error según cumpla con los criterios definidos.

Acciones
Generar póliza
Descripción: Utiliza el id del Empleado seleccionado, el sku del inventario seleccionado y una cantidad para poder generar la póliza, con el objetivo de restar al inventario objetivo.
Características: Utiliza un combo para poder elegir el empleado deseado y obtener su id, aplica lo mismo para el inventario a restar y al final se ingresa la cantidad a sustraer, como restricción no se puede tomar más de lo que hay restante, en caso de ocurrir un error reporta el mensaje en la pantalla para que el usuario esté

enterado. Después genera una transacción para poder restar de forma segura la cantidad objetivo al inventario deseado para después generar la póliza. Una vez terminado regresa al listado anterior.

BACKEND
Para el back-end se siguió el patrón Modelo Vista Controlador seccionado por capas, de las cuales se centran en una tarea en especifico, que van desde la recepción de la petición, el procesamiento de la información y su requisición desde la Base de Datos y todo el flujo de regreso hacia la aplicación visual.

Se emplearon funciones y métodos generales para estandarizar el mismo tipo de respuesta y hacer el mantenimiento más sencillo.

Descripción:

La API ha sido diseñada para gestionar los diferentes procesos relacionados con el manejo de empleados, puestos, inventarios y pólizas dentro del sistema. Se ha estructurado en cuatro controladores principales, cada uno encargado de una entidad específica, lo que permite una organización clara y modular del código.

Cada controlador sigue los principios de las buenas prácticas RESTful, utilizando métodos HTTP apropiados para cada operación, asegurando así una comunicación eficiente y predecible entre el cliente y el servidor. Además, se han implementado mecanismos para el manejo de excepciones y validaciones a fin de garantizar la integridad y seguridad de los datos procesados.

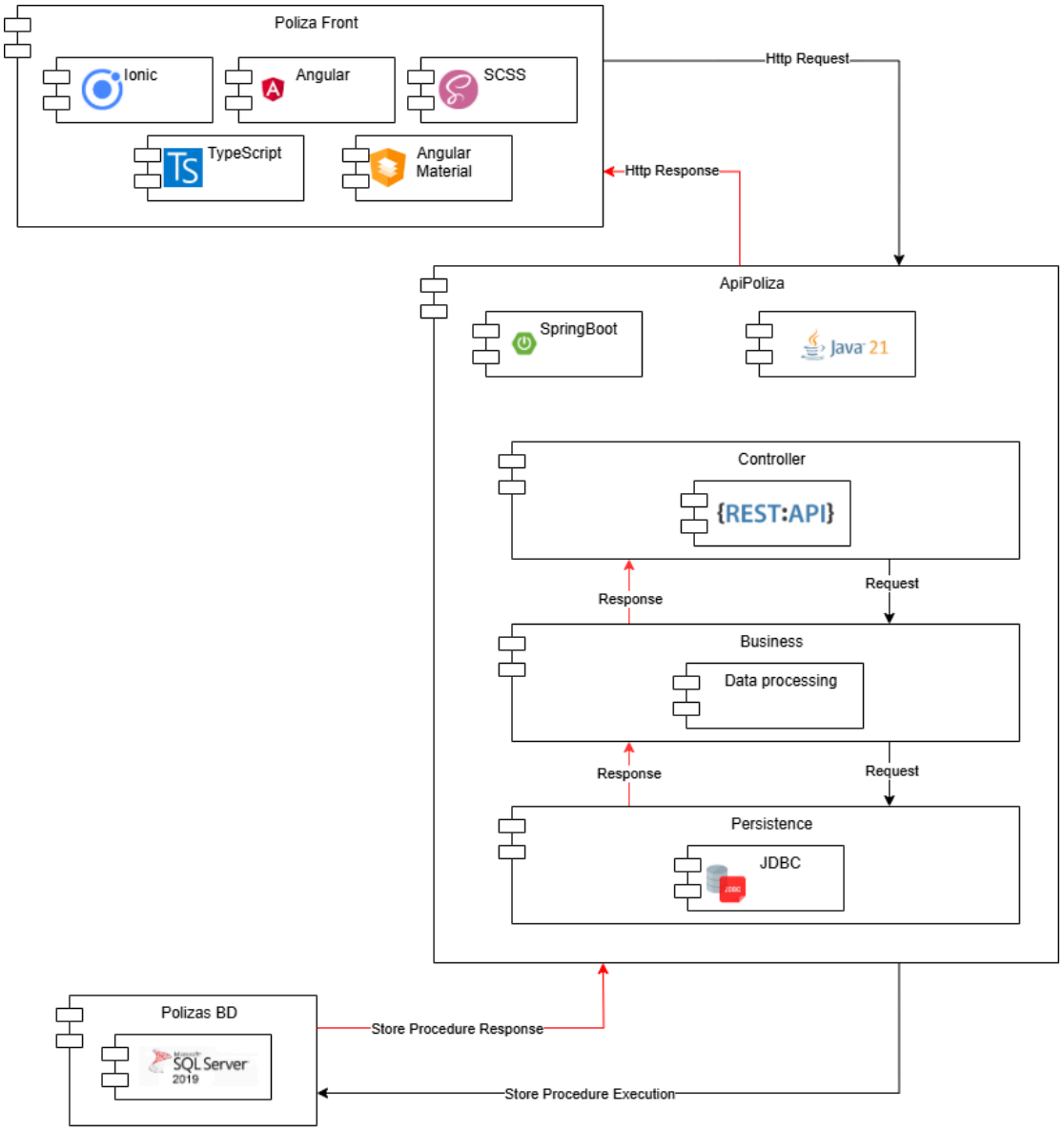
A continuación, se detallan los controladores y sus funcionalidades::

- **EmpleadoController**
  - Gestiona la información relacionada con los empleados, permitiendo la consulta y actualización de sus datos.
  - Endpoints
    - *[GET] /empleado*
      - Retorna una lista de empleados que actualmente se encuentran activos en el sistema.
    - *[GET] /empleado/{id}*
      - Obtiene los detalles de un empleado específico, identificado por su ID.
    - *[PUT] /empleado*
      - Permite actualizar los datos de un empleado registrado.
- **PuestoController**
  - Facilita la administración de los puestos dentro de la organización.
  - Endpoints
    - [*GET] /puesto*
      - Recupera la lista de puestos activos disponibles en el sistema.
- **InventarioController**
  - Encargado de gestionar los elementos dentro del inventario de la empresa.
  - Endpoints
    - *[GET] /inventario*
      - Obtiene todos los elementos del inventario que están actualmente disponibles.
    - *[PUT] /inventario/ajusteinventario*
      - Ajusta el inventario al sustraer la cantidad deseada del inventario, si se excede esta cantidad marca la excepción correspondiente
    - *[PUT] /inventario/restaurarinventario*

        - Ajusta el inventario al agregar la cantidad asignada a la póliza objetivo.
- **PolizaController**
    - Administra las operaciones relacionadas con las pólizas, permitiendo su creación, consulta y eliminación.
    - Endpoints
        - *[POST] /poliza/paginado*
            - Recupera un conjunto de pólizas de manera paginada, optimizando el rendimiento en consultas de grandes volúmenes de datos.
        - *[POST] /poliza*
            - Permite registrar una nueva póliza en el sistema.
        - *[PUT] /poliza/cambiarempleado*
            - *Permited cambiar el empleado responsable de la póliza*
        - *[DELETE] /poliza*
            - Elimina una póliza específica, garantizando la validación de permisos y restricciones antes de su eliminación.

Caracteristicas:
Esta API fue implementada con el framework de SpringBoot que cuenta con distintos endpoints que cubren diferentes peticiones dentro del aplicativo de pólizas. Como normativa se emplearon distintas anotaciones para sobrellevar los patrones de trabajo utilizados en la arquitectura de servicios web RESTful, ejemplo de estos son *@RestController*, *@RequestMapping* y *@GetMapping*. Para su mayor modularización se emplearon interfaces que nos apoyan a mantener el código mejor estructurado y ordenado para ello se usa la anotación @Autowired útil para la inyección de estas dependencias; también se resalta el uso de @CrossOrigin en los controladores para ayudar a aceptar solicitudes de cualquier origen (para este caso los que provienen del front) sin restricciones.

Diagrama arquitectonico



Ficha Tecnica

FrontEnd
Software
- Framework
  - Angular v14.2.14
  - Nodejs v16.19.1
  - Ionic v6.20.8
- HTML
- SCSS
- TypeScript v4.7.3

Hardware
Se puede emplear en computadoras, teléfonos o tablets. Para este caso se empleó un equipo personal. Como referencia: Laptop Acer Nitro 5 2020.

Backend
Software
- SprintBoot v3.4.2 en Java 21 usando Maven
  - Dependencias:
    - Spring Boot DevTools
    - Spring Boot Starter for Web
    - Spring Boot Starter for Test
    - Spring Boot Starter for Data JDBC
    - Microsoft SQL Server JDBC Driver
    - Project Lombok
    - Jackson DataBind
    - Logback Classic
- Base de datos Microsoft SQL
  - SQL server 2019
  - Gestor de SSMS

Hardware
Se empleó equipo personal
Caracteristicas
- Procesador: Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz 2.40 GHz
- RAM: 24GB
- Almacenamiento
  - 1000GB SSD (principal)
  - 1000GB HDD (secundario)

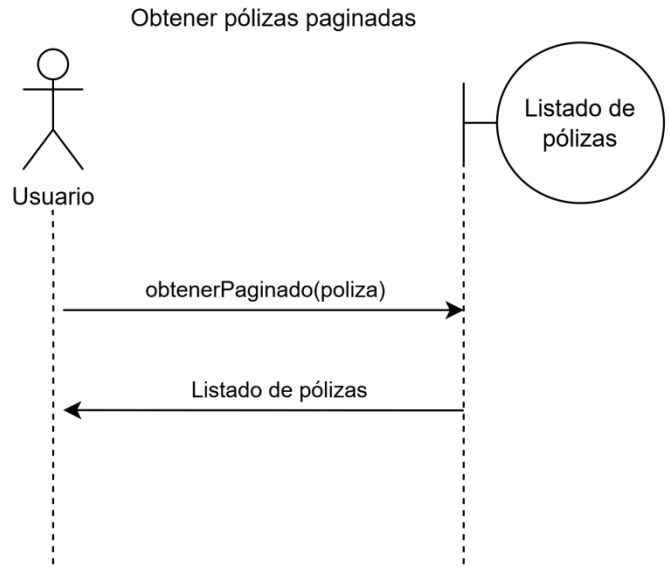
Bases de datos
Tablas
- CatEmpleado
  - IdEmplado **int**
  - Nombre **varchar(250)**
  - Apellido **varchar(250)**
  - IdPuesto **int**
  - Activo **bit**
  - FechaRegistro **datetime**
- CatPuesto

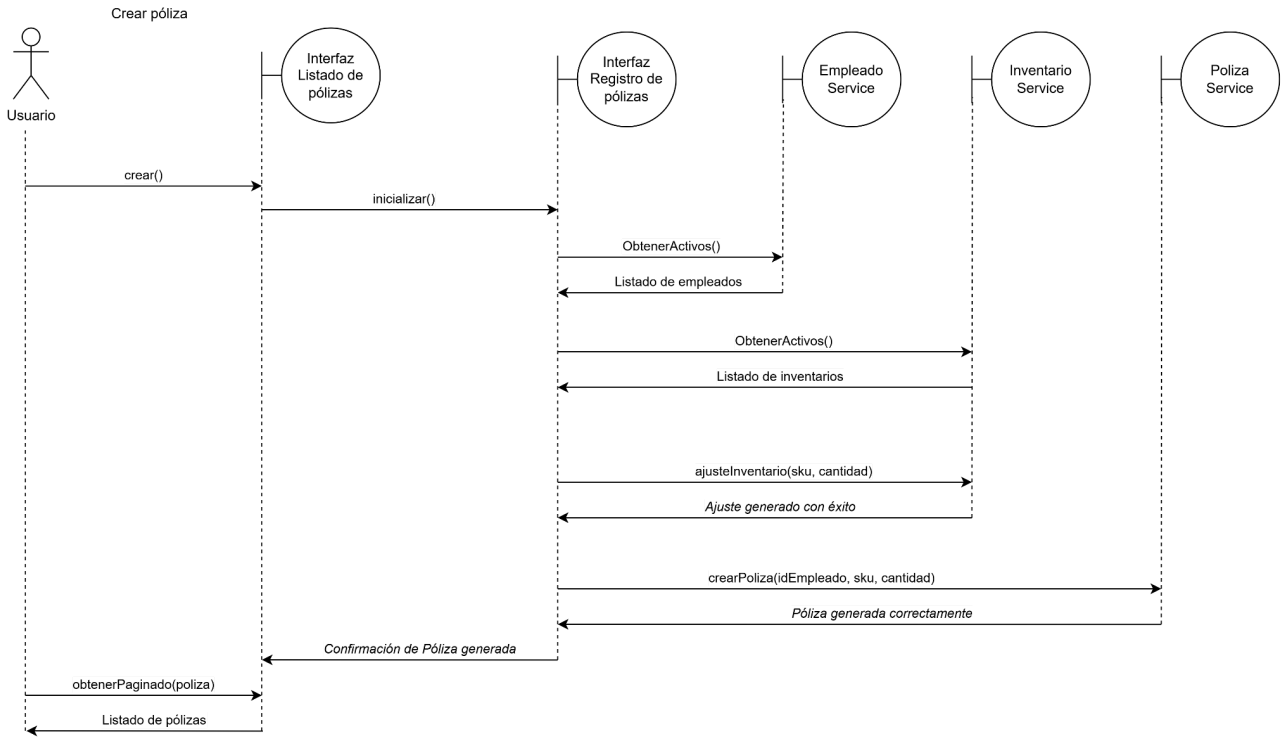
    - IdPuesto **int**
    - Nombre **varchar(250)**
    - Activo **bit**
    - FechaRegistro **datetime**
- MovInventario
    - SKU **varchar(5)**
    - Nombre **varchar(250)**
    - Cantidad **int**
    - Activo **bit**
    - FechaRegistro **datetime**
- MovPolizas
    - IdPoliza **int**
    - IdEmpleado **int**
    - SKU **varchar(5)**
    - Cantidad **int**
    - Activo **bit**
    - FechaRegistro **datetime**

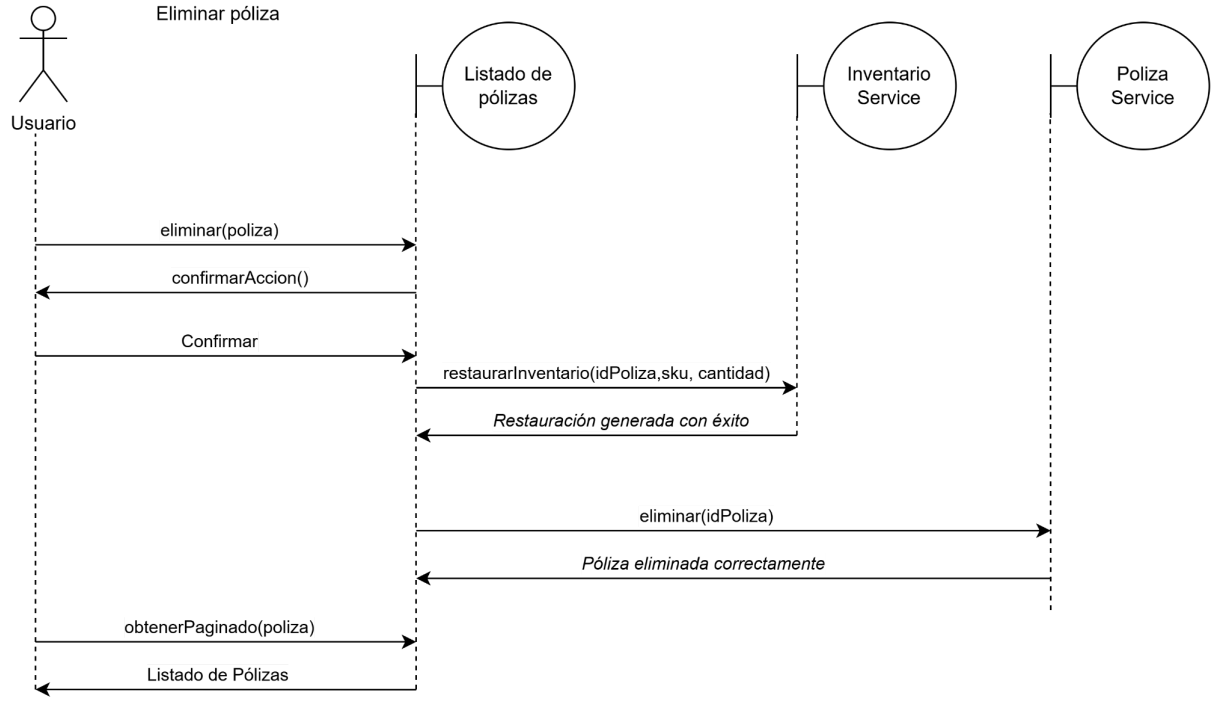
StoreProcedures

- obtenerPuestos
- obtenerPuestoPorId
- obtenerEmpleadosActivos
- obtenerEmpleado
- actualizarEmpleado
- obtenerInvetarios
- obtenerInvetarioPorSKU
- ajusteInventario
- restaurarInventario
- crearPoliza
- obtenerPolizas
- obtenerPolizasPaginadas
- actualizarEmpleadoPoliza
- eliminarPoliza

Diagrama de secuencia




Crear póliza
Usuario
Interfaz Listado de pólizas
Interfaz Registro de pólizas
Empleado Service
Inventario Service
Poliza Service
crear()
inicializar()
ObtenerActivos()
Listado de empleados
ObtenerActivos()
Listado de inventarios
ajusteInventario(sku, cantidad)
Ajuste generado con éxito
crearPoliza(idEmpleado, sku, cantidad)
Póliza generada correctamente
Confirmación de Póliza generada
obtenerPaginado(poliza)
Listado de pólizas


Eliminar póliza
Usuario
Listado de pólizas
Inventario Service
Poliza Service
eliminar(poliza)
confirmarAccion()
Confirmar
restaurarInventario(idPoliza,sku, cantidad)
Restauración generada con éxito
eliminar(idPoliza)
Póliza eliminada correctamente
obtenerPaginado(poliza)
Listado de Pólizas

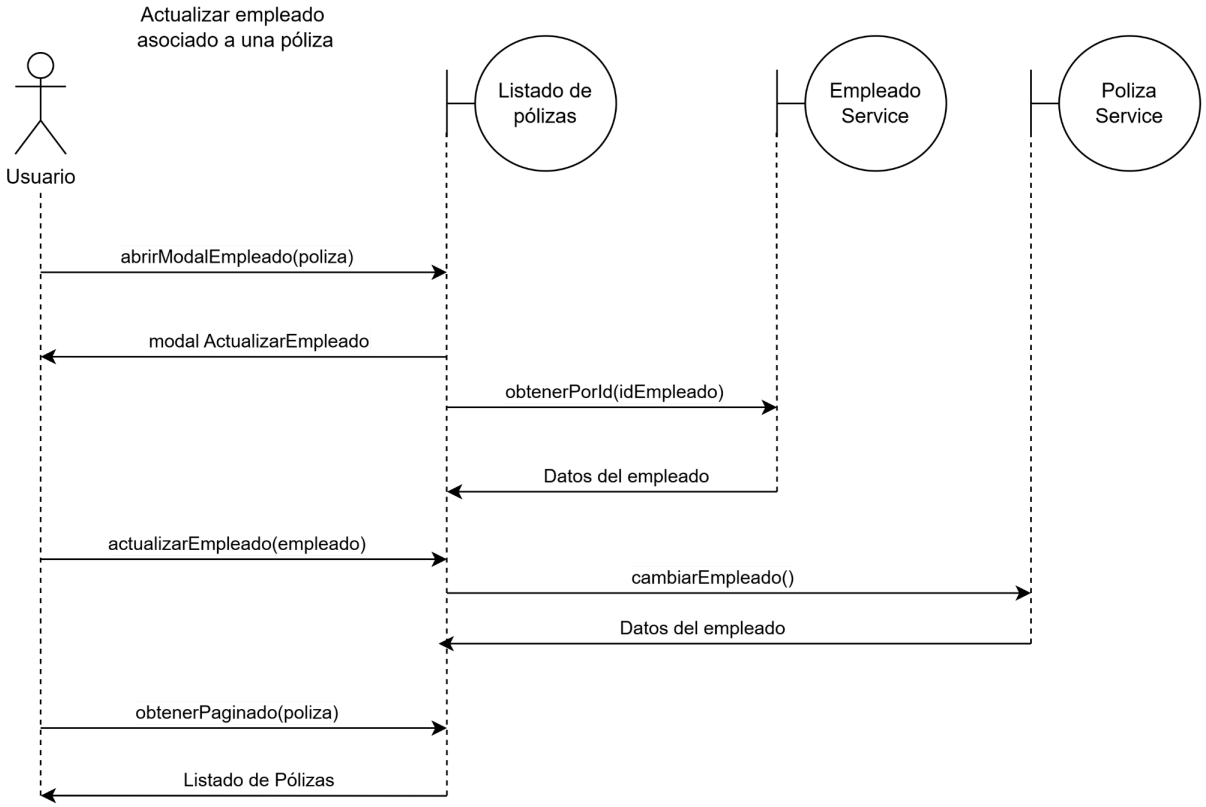

Actualizar empleado
asociado a una póliza
Usuario
Listado de
pólizas
Empleado
Service
Poliza
Service
abrirModalEmpleado(poliza)
modal ActualizarEmpleado
obtenerPorId(idEmpleado)
Datos del empleado
actualizarEmpleado(empleado)
cambiarEmpleado()
Datos del empleado
obtenerPaginado(poliza)
Listado de Pólizas

Diagrama de datos