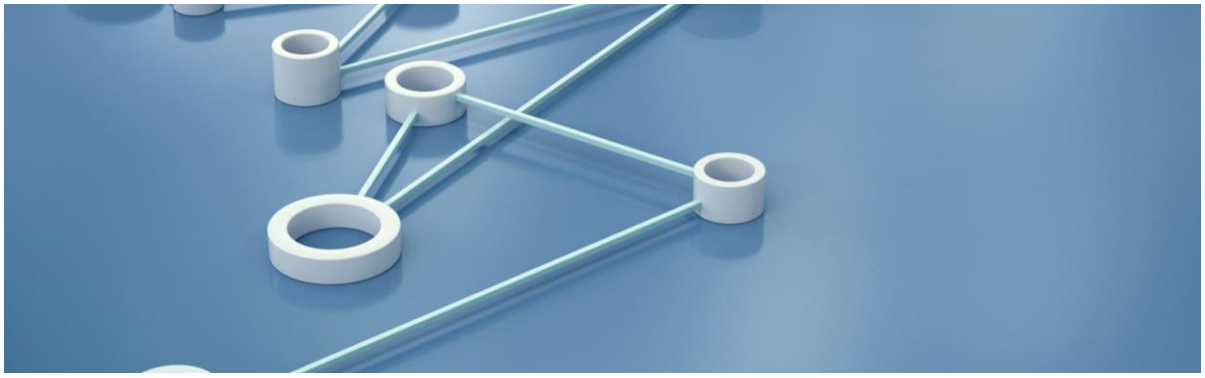

# SISTEMA DE GESTIÓN DE EMPLEADOS

Descripción sobre cada función creada, detallando cuál es su función y su implementación.

**‘createEmployee’**:

Esta función crea un nuevo empleado con el nombre del empleado y el ID del empleador proporcionado en el cuerpo de la solicitud.

- Utiliza el modelo ‘Employee’ de la base de datos para almacenar la información del empleado.
- Incrementa automáticamente su versión cuando es editado.

**‘createEmployer’:**

Crea un nuevo empleador, utilizando el nombre del empleador en el cuerpo de la solicitud.

- Utiliza el modelo ‘Employer’ que almacena la información de cada empleador.
- Este no posee cambios en la versión ya que no tiene una versión asociada.

**‘getAllEmployees’:**

Con esta función, podemos obtener una lista completa de los empleados, incluyendo la información sobre quien es su empleador y la versión asociada a cada empleado.

- Utiliza el modelo ‘Employee’ y ‘Employer’ sumando la función ‘findAll’ de sequelize para mostrar toda la información.

**‘updateEmployer’:**

Actualiza el empleador de un empleado, incrementando la version del empleado actualizado.

- Utiliza el modelo ‘Employee’ y ‘Employer’ de la base de datos para recibir la información que debe actualizar.
- Incrementa la versión del empleado cada vez que se hace un cambio en el empleador.

**‘handleNullEmployer’:**

Obtiene una lista de empleados que no tienen un empleador.

- Utiliza la función ‘findAll’ de sequelize para filtrar los empleados sin empleador.

**‘getFullhierarchy’:**

Obtiene la jerarquía completa de la empresa, considerando empleadores, empleados y su historial.

- Utiliza los modelos ‘Employer’, ‘Employee’, y ‘HierarchyHistory’ para construir la jerarquía completa.

# Descripción detallada sobre el proyecto:

Comenzando con la estructura de datos que he elegido utilizar, he tomado la decision de utilizar modelos de datos tales como “Employee”, “Employer”, y “HierarchyHistory”, que me permiten almacenar la

información de manera eficiente, también he decidido almacenar la versión de cada empleado teniendo un seguimiento en los cambios jerárquicos. Esta información es almacenada en el modelo de "Employee".

El incremento de versiones lo he atado a cada vez que a un empleado se le asigna un nuevo empleador, por lo tanto, cada vez que un empleado tenga a su nuevo empleador a este se le incrementara +1 la versión. Incrementando esta versión, proporciona un método más claro y eficiente para saber la cantidad de cambios que ha tenido el empleado.

Para el caso en el que se quiera ver la información completa de la jerarquía de la empresa, he decidido mostrar, por cada empleador, la información de su ID, el empleado al que está atado y la ID de ese empleado, incorporando la versión del empleado para dar a entender que antes de ese empleador, habría tenido otro. Para esto también he utilizado sequelize, para hacer una consulta de datos rápida y estructurada.

Hablando del manejo de nulos, la decision fue bastante sencilla de tomar, simplemente el empleado todavía no ha sido asignado con un empleador, hasta esto tendrá de valor NULL en el campo de empleador, hasta que, en la actualización de datos, se le asigne el empleador correspondiente, en este caso el id de su empleador se guardará en la base de datos y su versión aumentará por el cambio de la información.

También he decidido que lo más inteligente y eficiente es implementar un modelo de 'Hierarchy History' para guardar el historial de cambios, proporcionado gracias a la implementación de un método claro y eficiente que realiza un seguimiento de las modificaciones en la estructura jerárquica.

Para finalizar, quiero destacar que he decidido implementar un amplio manejo de errores, siendo un paso esencial para garantizar un código claro y fácil de entender.