Turnos Rotativos

Empleado

Consideraciones

Se utiliza EmpleadoDTO para transferir información entre capas

Repositorio contiene métodos query **exists** para hacer validaciones más rápidas

removeEmpleado se agregó una validación para impedir se intente borrar un empleado relacionado a una Jornada

Reglas de negocio

Si una regla de negocio no es cumplida, los métodos usarán **BussinessException**. Que recibe dos parámetros: Mensaje y Código con los que forma la excepción a mostrar.
El handler se encuentra en **CustomExceptionHandler**.

Alta empleado

Para cumplir con el requerimiento de dar de alta un empleado, se creó una entidad **Empleado**. En la misma se hacen las validaciones a través de anotaciones

- @NotNull para campos requeridos
- @Pattern para string de solo letras,
- @DateFormat para darle el formato requerido a las fechas
- @PastOrPresent para validar fecha

El resto de validaciones requeridas para el alta de un empleado son definidos e implementadas dentro **EmpleadoServiceImplement** y llamados por **addEmpleado**.

Para verificar Email y NroDocumento se usan métodos definidos al final de **EmpleadoServiceImplement**.

- existsEmpleadoWithEmail(String email)
- existsEmpleadoWithNroDocumento(Integer NroDocumento)

Para validar si el empleado es mayor de edad se usó el método:

- boolean esMayorDeEdad (LocalDate fechaNacimiento).

Actualizar Empleado

Se usan mismas validaciones que **Alta empleado** y al final se valida con un **exists** (método query) si existe el empleado al que se busca actualizar.

Conceptos

Consideraciones

Se utiliza ConceptoDTO para transferir información entre capas

Jornadas

Consideraciones

Entidad Jornada está relacionada 1 a 1 con Empleado y Concepto. Mediante sus respectivos Ids.

Se utiliza **JornadaResponse**, para mostrar solo la información necesaria y **JornadaRequest**, para recibir información necesaria. Me pareció una buena oportunidad de experimentar con esta forma de mostrar y recibir datos. Que según entendí reemplaza al DTO.

Reglas De Negocio todas las validaciones están realizadas en clase JornadaValidation, que podría haber sido Interface pero priorice la legibilidad y que cada validación sea independiente. Además me propuse hacer la mayor cantidad de validaciones con lambdas (espero no haber exagerado).

Alta Jornada Laboral

Para dar de alta una jornada laboral se utiliza el método **AddJornada** que recibe como parámetro **JornadaRequest**. Dentro del método se utiliza una entidad **Jornada**, para poder hacer todas las validaciones necesarias y al finalizar se devuelve **JornadaResponse**.

Validaciones

A continuación se hará referencia a cada punto detallado en **REGLAS DE NEGOCIO** - **HU006**. Junto con el método que realiza la validación(en carpeta validations, JornadaValidation).

1. public void **validarRangoDeHoras**(Integer hsTrabajadas, Integer hsMinimo, Integer hsMaximo, Boolean laborable)
2. public void **validarDiaLibre**(List<Jornada> todasLasJornadas, Integer idEmpleado, LocalDate fecha)

3. public void **validarMismoConcepto**(List<Jornada> todasLasJornadas, Integer idEmpleado, Integer idConcepto, LocalDate fecha)

4. public void **validarHorasDia**(List<Jornada> todasLasJornadas, Integer idEmpleado,LocalDate fecha, Integer hsTrabajadas)

5. public void **validarTurnosYDiaLibre**(List<Jornada> todasLasJornadas, Integer idEmpleado, Integer idConcepto, LocalDate fecha)

6. public void **validarHorasSemana**(List<Jornada> todasLasJornadas, Integer idEmpleado,LocalDate fecha, Integer hsTrabajadas)

7. public void **validarTurnosExtra**(List<Jornada> todasLasJornadas, Integer idEmpleado, Integer idConcepto, LocalDate fecha)

8. public void **validarTurnosNormales**(List<Jornada> todasLasJornadas, Integer idEmpleado, Integer idConcepto, LocalDate fecha)

9. public void **validarDiasLibres**(List<Jornada> todasLasJornadas, Integer idEmpleado, Integer idConcepto, LocalDate fecha)

Dentro de JornadaValidation se crearon constantes para no hardcodear límites de horas o de conceptos semanales/diarios

(al momento de escribir el readme noto podría haber mejorado la forma de pasar parametros, para no repetirme tanto).

Obtener Jornadas(Query Params)

Para realizar obtener jornadas segui estos pasos:

1. En el repositorio cree los siguientes métodos, findByEmpleadoNroDocumento, findByFecha y uno que combine ambos findByEmpleadoNroDocumentoAndFecha.

2. En Servicio el siguiente: getJornadasByFechaAndDocumento quien se encarga de validar los parámetros.

3. El controlador recibe @RequestParam o no y luego llama el método del servicio que se encarga de filtrar las jornadas.

Github

Repositorio

https://github.com/Ignacio-albornoz/tp2_api

(el repositorio es público si necesitan deje de serlo por lo comentado en el primer trabajo práctico, contener la palabra neoris, me avisan e inmediatamente lo paso a privado).