Ejercicio - Funcionalidad - Enunciado

Vamos a desarrollar una funcionalidad para generar préstamos.

Tenemos todo montado excepto los casos de generación de préstamos (precondiciones y condiciones)

Definición OpenApi

Extracto que tenemos ya implementado en OpenAPI

- endpoint
- schemas

```
      /prestamos:
             post:
           summary: Crear Prestamo
           operationId: crearPrestamo
           tags:
           - prestamos
           requestBody:
             description: Crear una nueva Prestamo en la Aplicación
             content:
               application/json:
                 schema:
                   $ref: "#/components/schemas/PrestamoCreate"
           responses:
             "201":
               description: Consejería creada correctamente
               content:
                 application/json:
                   schema:
                     $ref: "#/components/schemas/PrestamoDetail"
     [...]
     PrestamoCreate:
    type: object
    required:
      - socio_id
      - libro_id
    properties:
      socio_id:
        type: integer
        format: int64
      libro_id:
        type: integer
        format: int64
     PrestamoDetail:
    type: object
    properties:
      id:
        type: integer
        format: int64
      socio:
        $ref: "#/components/schemas/SocioDetail"
      libro:
        $ref: "#/components/schemas/LibroDetail"
      devolucion:
        type: date
```

tenemos el servicio que guarda prestamo tras calcular la fecha concreta

```
public class SavePrestamoService {

        @Autowired
        PestamosRepository prestamosRepository;
        @Autowired
        CalcularPrestamoService calcularPrestamoService;

        public Prestamo execute(Socio socio, Libro libro, LocalDate localDate){
         int prestamoDias = 7;

         return Prestamo.builder()
                 .libro(libro)
                 .socio(socio)
                 .expiraEn(localDate.plusDays(prestamoDias))
                 .build();
    }

}
```

En los modelos tenemos funciones que nos indican acciones que vamos a necesitar

Socio

- `public boolean haSuperadoElLimiteDePrestamo()`
- `public boolean tienePrestamoVencido()`

Libro

- `public boolean estaEnPrestamo()`

Sólo es necesario modificar el Servicio

Puedes implementar todo el ejercicio cambiando únicamente la lógica del método `execute` de `CalculaPrestamoService`

1. `CalculaPrestamoService` Casos de Uso

En la clase ir implementando los casos de uso

1. El tipo de usuario (estudiante, profesor, visitante) afecta el tiempo del préstamo:
   - Visitantes: 7 días.
   - Estudiantes: 15 días.
   - Profesores: 30 días.
2. El elemento solicitado no debe estar prestado a otro usuario.
3. El usuario no debe tener más de 3 libros prestados.
4. El usuario no debe tener ningún libro retrasado.

5. Los profesores pueden pedir libros en cualquier momento, mientras que los estudiantes y visitantes solo pueden hacerlo de lunes a viernes.

2. Comprobar funcionamiento.

Comprobar que la funcionalidad es correcta.

3. Refactorizar

Una vez comprobado cambia el código para intentar dejarlo más claro para los futuros mantenimiento.

Refactorizar es el proceso de cambiar el código pero que siga haciendo lo mismo.