INSTITUTO DE EDUCACIÓN SUPERIOR CIBERTEC
DIRECCIÓN ACADÉMICA
CARRERA PROFESIONALES

CURSO	:	**Desarrollo de Aplicaciones Web I**
PROFESOR	:	**César Enrique Santos Torres**
CICLO	:	**Quinto**
SECCIÓN	:	
GRUPO	:	
FECHA	:	**03/12/2024**
DURACIÓN	:	**50 minutos**

NOTA

ALUMNO (A) : Jesus Medina Lazaro

CASO DE LABORATORIO 2 (CL2)

Consideraciones generales:

- El laboratorio consta de 1 Crud implementado con Spring MVC + Spring Data JPA, cada operación del CRUD deberá ir acompañada (De forma obligatoria) de capturas de pantalla de lo implementado.

- Sólo debe subir este documento, con sus evidencias y respuestas en él. El código fuente del proyecto debe ser subido a Github (Adjuntar link del repositorio). No se aceptará código zipeado.

- El nombre del presente archivo deberá tener la siguiente estructura: "DAWI-APELLIDOPATERNO-APELLIDOMATERNO-NOMBRES.pdf".

LOGRO DE LA EVALUACION:
Al término de la evaluación, el alumno deberá implementar un CRUD con Spring MVC, dicho CRUD deberá incluir las siguientes operaciones:

- Consulta de películas
- Detalle de una película
- Actualización de una película
- Eliminación de una película
- Registro de una película

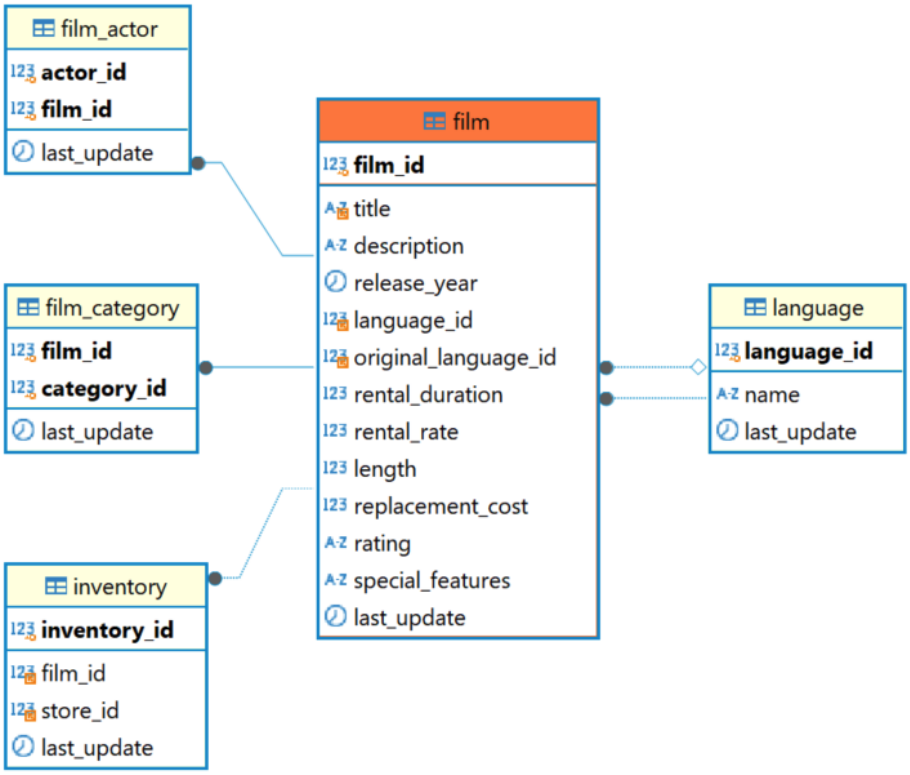
CONSOLIDADO

Pregunta	Puntaje		Llenar solo en caso de Recalificación justificada	
	Máximo	Obtenido	Sustento	Puntaje
1	5			
2	5			
3	5			
4	5			
Total	20			

Nota Recalificada	

Alcance de la prueba

Implementar un CRUD del siguiente modelo de BD, utilizando Spring MVC y Spring Data JPA. El CRUD deberá realizarse de la tabla “film”, considerando su impacto en las tablas con las que se relaciona.




```
public interface MaintenanceService {  4 usages  1 implementation

    List<FilmDto> findAllFilms();  1 usage  1 implementation

    FilmDetailDto findDetailById(Integer id);  2 usages  1 implementation

    Boolean updateFilm(FilmDetailDto filmDetailDto);  1 usage  1 implementation

    Boolean deleteFilm(Integer id);  1 usage  1 implementation
}
```

Consulta de films

- Consulte el listado total de films (Sin paginado) y muestrelo en un “template”. Considere no traer todos los campos de la tabla, sólo lo necesario haciendo uso de un DTO.

```java
@Override  1 usage
public List<FilmDto> findAllFilms() {

    List<FilmDto> films = new ArrayList<FilmDto>();
    Iterable<Film> iterable = filmRepository.findAll();
    iterable.forEach(film -> {
        FilmDto filmDto = new FilmDto(film.getFilmId(),
                film.getTitle(),
                film.getLanguage().getName(),
                film.getRentalDuration(),
                film.getRentalRate());
        films.add(filmDto);
    });
    return films;

}
```

Detalle del film

- Consulte los datos de un film seleccionado y muestrelo en un “template”. Considere el uso de un DTO para mostrar los campos en un formulario de solo lectura.

```java
@Override  2 usages
public FilmDetailDto findDetailById(Integer id) {

    Optional<Film> optional = filmRepository.findById(id);
    return optional.map(
            film -> new FilmDetailDto(film.getFilmId(),
                    film.getTitle(),
                    film.getDescription(),
                    film.getReleaseYear(),
                    film.getRentalDuration(),
                    film.getRentalRate(),
                    film.getLength(),
                    film.getReplacementCost(),
                    film.getRating(),
                    film.getSpecialFeatures(),
                    film.getLastUpdate())
    ).orElse( other: null);

}
```

Actualización del film

- Actualice los datos de un film seleccionado, a través de un formulario de edición. No considere la actualización del campo "language". Considere implementar una alerta (Venta de confirmación), antes de proceder con la actualización.

```
@Override  1 usage
public Boolean updateFilm(FilmDetailDto filmDetailDto) {

    Optional<Film> optional = filmRepository.findById(filmDetailDto.filmId());
    return optional.map(
            film -> {
                film.setTitle(filmDetailDto.title());
                film.setDescription(filmDetailDto.description());
                film.setReleaseYear(filmDetailDto.releaseYear());
                film.setRentalDuration(filmDetailDto.rentalDuration());
                film.setRentalRate(filmDetailDto.rentalRate());
                film.setLength(filmDetailDto.length());
                film.setReplacementCost(filmDetailDto.replacementCost());
                film.setRating(filmDetailDto.rating());
                film.setSpecialFeatures(filmDetailDto.specialFeatures());
                film.setLastUpdate(new Date());
                filmRepository.save(film);
                return true;
            }
    ).orElse( other: false);

}
```

Eliminación del film

- Elimine un film seleccionado (Aplique delete), a través del enlace de "Eliminar" de la grilla de consulta de films. Considere que la eliminación deberá eliminar en cascada las tablas "film_actor", "film_category" y "inventory". Considere también implementar una alerta (Venta de confirmación), antes de proceder con la eliminación

```
@Override  1 usage
public Boolean deleteFilm(Integer id) {
    return filmRepository.findById(id).map(film -> {
        filmRepository.deleteById(film.getFilmId());
        return true;
    }).orElse( other: false);
}
```

```
<td>
    <a th:href="@{/maintenance/delete/{id}(id=${film.filmId()})}" class="btn btn-danger" onclick="return confirmUpdate()">Eliminar</a>
</td>
```

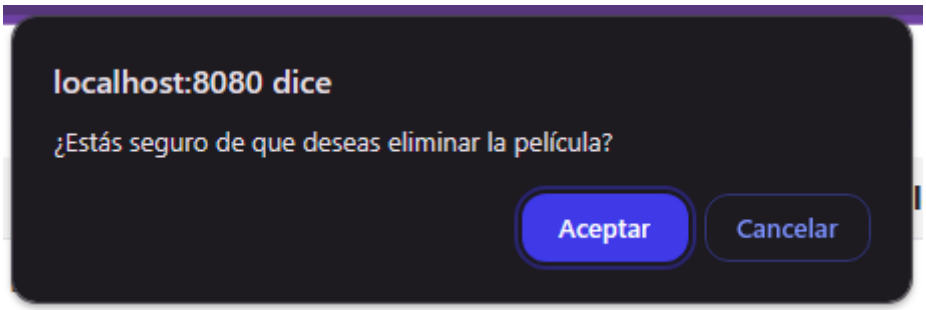
Prueba:

1)

Mantenimiento de películas

Código	Título	Idioma	Tiempo de alquiler (Días)	Precio de alquiler (USD)	Acciones	
3	ADAPTATION HOLES	English	7	2.99	Editar	Eliminar
4	AFFAIR PREJUDICE	English	5	2.99	Editar	Eliminar
6	AGENT TRUMAN	English	3	2.99	Editar	Eliminar
7	AIRPLANE SIERRA	English	6	4.99	Editar	Eliminar
8	AIRPORT POLLOCK	English	6	4.99	Editar	Eliminar
9	ALABAMA DEVIL	English	3	2.99	Editar	Eliminar
10	ALADDIN CALENDAR	English	6	4.99	Editar	Eliminar
11	ALAMO VIDEOTAPE	English	6	0.99	Editar	Eliminar
12	ALASKA PHANTOM	English	6	0.99	Editar	Eliminar
13	ALI FOREVER	English	4	4.99	Editar	Eliminar
14	ALICE FANTASIA	English	6	0.99	Editar	Eliminar
15	ALIEN CENTER	English	5	2.99	Editar	Eliminar

2)



3)

Mantenimiento de películas

Código	Título	Idioma	Tiempo de alquiler (Días)	Precio de alquiler (USD)	Acciones	
3	ADAPTATION HOLES	English	7	2.99	Editar	Eliminar
4	AFFAIR PREJUDICE	English	5	2.99	Editar	Eliminar
7	AIRPLANE SIERRA	English	6	4.99	Editar	Eliminar
8	AIRPORT POLLOCK	English	6	4.99	Editar	Eliminar
9	ALABAMA DEVIL	English	3	2.99	Editar	Eliminar
10	ALADDIN CALENDAR	English	6	4.99	Editar	Eliminar
11	ALAMO VIDEOTAPE	English	6	0.99	Editar	Eliminar
12	ALASKA PHANTOM	English	6	0.99	Editar	Eliminar
13	ALI FOREVER	English	4	4.99	Editar	Eliminar

Como ve ya no hay el registro 6 **--- Se logró ---**

Registro del film

- Registre un film al hacer clic en “Nuevo film”, la operación a gatillar será el registro de un nuevo “film” y la selección de su lenguaje correspondiente (En el formulario de registro, puede considerar una lista desplegable para mostrar las categorías o simplemente una caja de texto para ingresar el código de la categoría).

```java
@CacheEvict(value = "films", allEntries = true)  1 usage
@Override
public void createFilm(FilmData filmData) {
    Optional<Language> language = languageRepository.findById(filmData.languageId());
    if (language.isEmpty()) {
        throw new IllegalArgumentException("El lenguage con ID " + filmData.languageId() + " no existe");
    }

    Film film = new Film();
    film.setTitle(filmData.title());
    film.setDescription(filmData.description());
    film.setReleaseYear(filmData.releaseYear());
    film.setRentalDuration(filmData.rentalDuration());
    film.setRentalRate(filmData.rentalRate());
    film.setLength(filmData.length());
    film.setReplacementCost(filmData.replacementCost());
    film.setRating(filmData.rating());
    film.setSpecialFeatures(filmData.specialFeatures());
    film.setLanguage(language.get());
    film.setLastUpdate(new Timestamp(System.currentTimeMillis()));

    filmRepository.save(film);
}
```

Gestión de caching

- El archivo “application.properties” deberá tener configurado el pintado de SQL en la terminal: **spring.jpa.show-sql=true**
- La primera vez que ingrese a la grilla de consulta de films, deberá ir a BD y traer los datos. Pero en las posteriores invocaciones deberá traer los datos del caché. A menos que, haya realizado una operación de “actualización”, “registro” o “eliminación”.

```java
@Configuration  no usages
@EnableCaching
public class CacheConfig {
}
```

Diseño de arquitectura

- Debe respetar las pautas de la arquitectura (Modelo de capas), notación CamelCase y buenas prácticas.

Link de mi repo: **https://github.com/LGsus113/CL2-JesusMedina.git**