Instrucciones

1. Descargue este repositorio y abra los proyectos en Netbeans.
2. En Netbeans vaya a Services > Databases > JavaDB y cree una base de datos que se llame `series` (los demás campos déjelos en blanco).
3. En Netbeans vaya a Tools > Options > Java, seleccione la pestaña Maven y marque la opción `Skip Tests for any build executions no directly related to testing`.
4. Cada vez que complete un paso genere un commit.
5. Al finalizar ingrese a github y genere un release con el nombre `Entrega_Parcial_S4`.

Contexto

Esta aplicación permite gestionar series de televisión. El recurso presente en la aplicación es `Serie`, la cual tiene un nombre (`name: String`), una descripción (`description: String`) y un identificador (`id: Long`) que es la llave primaria.

En la aplicación usted encontrará que las funcionalidades de crear (`POST`) y solicitar (`GET`) ya están implementadas.

Se nos ha solicitado completar los siguientes requisitos:

Punto 1 (60%): Personajes

Se desea que el sistema permita gestionar ahora el listado de personajes que aparcen en cada serie.

De cada personaje se conoce su nombre (`name: String`), el nombre del actor que lo representa (`portrayedBy: String`) y se tiene un campo de identificación (`id: Long`) que representa la llave primaria del personaje.

Ud. debe extender su programa para que cuando ejecute

```
POST localhost:8080/s4_series-api/api/series
```

con el json:

```
{
    "name": "Breaking bad",
    "description": "Set and filmed in Albuquerque, New Mexico, t
    "personajes": [{
        "name": "Walter Hartwell White Sr",
        "portrayedBy": "Bryan Cranston"
    }]
}
```

se cree la serie con la información de uno o varios personajes.

Para esto Ud. debe:

1. (10%) Crear las clases `PersonajeDTO` y `PersonajeEntity` que modelan al personaje. En la clase `PersonajeDTO`, además de tener un constructor sin parámetros, defina un constructor para convertir un `PersonajeEntity` en un `PersonajeDTO`:

```
public PersonajeDTO(PersonajeEntity personaje) {
    this.id = personaje.getId();
    this.name = personaje.getName();
    ...
}
```

Para convertir un `PersonajeDTO` en un `PersonajeEntity` defina el siguiente método:

```
public PersonajeEntity toEntity() {
    PersonajeEntity entity = new PersonajeEntity();
    entity.setId(this.id);
    entity.setName(this.name);
    ...
    return entity;
}
```

1. (10%) Defina en `SerieEntity` la relación con `Personaje` (unidireccional) e implemente sus `set/get`. Esta es una relación de **composición** de uno de muchos (`OneToMany`).

2. (15%) Defina un atributo nuevo en `SerieDetailDTO` que representa el listado de personajes que participan en la serie. Defina `set/get` y

actualice el método constructor que recibe un `SerieEntity` al igual que el método `toEntity`, el cual retorna un objeto de tipo `SerieEntity`, para que también hagan la conversión del listado de personajes.

3. (25%) Modifique el método `createSerie` de la clase `SerieLogic` para que tenga en cuenta las siguientes reglas de negocio.
4. No deben existir dos series con la el mismo nombre.
5. La longitud de la descripción debe ser superior a 30 caracteres.

Si las reglas de negocio se cumplen, se debe llamar la persistencia para que el objeto sea persistido, de lo contrario debe lanzar una excepción `BusinessLogicException` con un mensaje donde se especifique cuál regla no se cumplió.

1. Ejecute su prueba unitaria.
2. Ejecute la siguiente prueba la cual debe arrojar el código 200.

`POST localhost:8080/s4_series-api/api/series/`

- json body

```
{
    "name": "Orange Is the New Black",
    "description": "The series begins revolving around Piper Cha
    "personajes": [{
        "name": "Alex Vause",
        "portrayedBy": "Laura Prepon"
    }]
}
```

- Fijese en el `id` que retornó el `POST` y ejecute

`GET localhost:8080/s4_series-api/api/series/:id`

1. Ejecute la siguiente prueba que debe arrojar un código 412, ya que la descripción de la serie no es superior a 30 caracteres.

```
{
    "name": "Game of Thrones",
    "description": "American fantasy drama",
    "personajes": [{
        "name": "Catelyn Tully",
```

```
        "portrayedBy": "Michelle Fairley"
    }]
}
```

Punto 2 (40%): Eliminar

Se desea que el sistema permita eliminar una serie y todos sus personajes. Para ello usted de cumplir con los siguientes pasos.

1. (10%) Cree un método con el nombre `delete` en la clase `SeriePersistent` el cual recibe el `id` de la serie y la elimina.
2. (10%) Cree un método de prueba en la clase `SeriePersistentTest` el cual valida que efectivamente se esté borrando la serie.
3. (10%) Cree un método con el nombre `deleteSerie` en la clase `SerieLogic` en el cual se debe validar la siguiente regla de negocio:
4. Solo se puede eliminar una serie que tenga personajes. En el caso que no tenga, el método debe lanzar una excepción `BusinessLogicException` con la información del error, de lo contrario se llama a la persistencia y se procede a eliminar la serie.
5. (10%) Cree el método `deleteSerie` en la clase `SerieResource` el cual verifica la existencia del recurso; en el caso que no exista lanza una excepción `WebApplicationException` con el mensaje correspondiente, de lo contrario llama a la lógica para validar reglas de negocio.
6. Verifique el funcionamiento ejecutando:

`DELETE localhost:8080/s4_series-api/api/series/:id`