PRÁCTICA BACKEND

MEMORIA

Tecnologías de Desarrollo para la Web

Realizado por:

Olga Arenas Rosario

Índice

Operaciones y relaciones para Entidades y Personas

Esta fase de la práctica se corresponde con el cumplimiento de los tres primeros objetivos propuestos:

1. Operaciones básicas para Entidades y Personas (GET, CGET, DELETE)
2. Operaciones POST y PUT para Entidades y Personas
3. Relaciones (métodos GET y PUT) para Entidades y Personas

Con el fin de lograrlos se han realizado modificaciones en diferentes clases del código, las cuáles son:

- *EntityCommandController*
- *EntityQueryController*
- *EntityRelationsController*
- *PersonCommandController*
- *PersonQueryController*
- *PersonCommandController*

Para llevar a cabo dichas modificaciones ha sido necesario completar los métodos marcados como *TO DO* en las clases pertinentes, tomando como referencia los controladores de *Product*.

Seguidamente se han realizado los tests de los controladores modificados referentes a personas y entidades para comprobar el correcto funcionamiento de los mismos. En este caso todas las pruebas se han ejecutado correctamente y no ha habido complicaciones.

Por último, se han realizado los comandos especificados en la práctica para la instalación del proyecto, incluyendo la conexión con la base de datos y la modificación de los archivos *./.env.local* y *./phpunit.xml* (aunque este último está orientado a las pruebas finales, se ha modificado en este punto).

Para que la conexión con la base de datos se realizase correctamente fue necesario cambiar el puerto del **3306** al **3307**.

Implementación de las operaciones para la gestión de Asociaciones (incluyendo pruebas)

Para cumplir con el objetivo propuesto se han llevado a cabo numerosas modificaciones en el código.

Comenzamos con la creación de una clase *Association*, la cual extiende de *Element* ya que sus atributos son similares. Como diferencia encontramos que en el constructor hemos añadido un array de entidades que se corresponden con aquellas que participan en la asociación. Además se ha añadido una relación **ManytoMany** con *Entities* y los métodos **get** y **set** necesarios.

Junto a esta clase también se ha creado una *AssociationFactory* que extiende de *ElementFactory*.

Por último se han creado los controladores necesarios para administrar las asociaciones:

- *AssociationCommandController*
- *AssociationQueryController*
- *AssociationRelationsController*

Lo siguiente que se ha realizado son modificaciones en la clase *Entity* para que esta pudiera relacionarse con las asociaciones. Se han añadido los métodos pertinentes y también se han modificado sus controladores para poder administrar las asociaciones con las que se relaciona.

También se han modificado:

Dentro de **demoAjax**:

- *index.html*

Dentro de **scripts**:

- *create.php*
- *list.php*
- *remove.php*

Estos se han modificado para añadir las asociaciones dentro de los métodos pertinentes.

Se ha creado además un script *entityBelongsAssociation.php*, que hace referencia a las entidades que participan en la asociación.

Por último, se han creado los test referentes a los controladores de *Association*, con todos los métodos necesarios para realizar las pruebas pertinentes, habiéndose instalado también los recursos necesarios para hacer **coverage-test**.

En este punto de la práctica sucedieron varios errores en las pruebas, los cuales quedaron resueltos al añadir en las **rutas** la información correspondiente a las asociaciones.

Completar la especificación OpenAPI para la gestión de Asociaciones, Entidades y Personas

Para añadir los elementos solicitados a la API ha sido necesario añadir los apartados correspondientes a *Associations*, *Persons* y *Entities*. Se ha realizado tomando como referencia los datos de *Products*.

Se han creado su apartado y sus componentes específicos, además de todos los métodos necesarios para su gestión.

Al hacer estas modificaciones había un problema en la API en el cual no cargaban los métodos de los elementos creados, este problema se solucionó corrigiendo la tabulación del código.

Para garantizar su correcto funcionamiento se ha desplegado la API usando el comando:

php -S 127.0.0.1:8000 -t public

Una vez desplegada la API se han probado los métodos correspondientes a todos los elementos, comprobando que funcionasen fijándonos en los códigos de estados que devolvían al ejecutarse.

Hay que tener en cuenta que para poder llegar a usar los métodos ha sido necesario autorizar al usuario **adminUser** (el primero que se crea al realizar *composer intall*) para que obtuviese permisos de **reader** y **writer**. Para ello debíamos introducir el usuario y contraseña correspondientes y escribir en **client id**: *tdw-api-client-id*.

Si no realizamos este último paso no se nos permitirá probar muchos de los métodos, ya que nos saldrá el error **401** de inautorizado.