

TDW ACiencia REST API

PRÁCTICA 2: BACKEND

MIGUEL CORONEL FERNÁNDEZ

INDICE

Descripción y finalidad de la practica	2
Repositorio	2
Esquema de relaciones:	2
Listado y descripción de los cambios realizados:.....	3
Openapi.yaml.....	3
RoutesAssociations.php	3
RoutesEntities.php	3
Association.php	3
Entity.php.....	4
AssociationFactory.php.....	4
AssociationQueryController.php.....	4
AssociationCommandController.php	5
AssociationRelationController.php	5
PersonRelationController.php.....	5
EntityRelationsController.php.....	5
EntityTest.php.....	5
EntityRelationsControllerTest.php.....	6
AssociationTest.php.....	6
AssociationControllerTest.php.....	6
AssociationRelationsControllerTest.php.....	6
Problemas encontrados	6
Conclusiones	7

Descripción y finalidad de la práctica

La práctica tiene como objetivo la implementación de una API REST para la gestión de forma estructurada de Productos, Entidades, Personas y Asociaciones relacionadas con las aportaciones a la ciencia, utilizando las tecnologías Slim4 y Doctrine ORM sobre PHP 8.3.

Se ha introducido una nueva entidad denominada Asociación con los mismos atributos que el resto de elementos y con una colección relacionada de entidades. Para ello se ha utilizado el ORM de Doctrine para proporcionar un completo desacoplamiento entre la lógica de negocio y la persistencia de los datos.

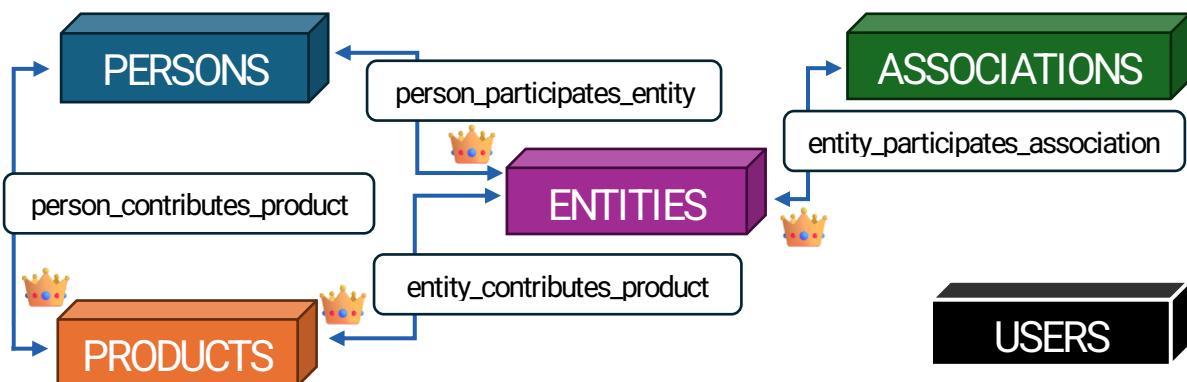
Se ha completado la especificación parcial OpenAPI recibida con el resto de elementos gestionados (Personas, Entidades y Asociaciones) empleando anotaciones y utilizando el editor Swagger.

Se han pasado con éxito los casos de prueba incluidos en la práctica y se han realizado nuevos casos para la clase Asociaciones y su Controller y RelationController. Además de ampliado las pruebas de Entidades y de su relación con las Asociaciones.

Repositorio

GitHub

Esquema de relaciones:



Listado y descripción de los cambios realizados:

[Openapi.yaml](#) (📝 Edit)

- **Añadir los tags faltantes:** Se han añadido en el fichero *openapi.yaml* los tags para las entidades, personas y asociaciones, agrupando y permitiendo visualizar las descripciones de los elementos
- **Definir Endpoints faltantes:** Se han añadido los *Endpoints* de la API de las entidades, personas y asociaciones, definiendo descripciones, parámetros y respuestas

Esto incluye la gestión de los elementos (Devolver todos los elementos, crear uno nuevo, recuperar uno específico por el nombre o el id, actualizarlo y eliminarlo) y la gestión de las relaciones de ese elemento (recuperar una lista de los relacionados, añadir uno nuevo o eliminar uno existente)

- **Definir los modelos de datos:** Se han añadido los modelos de asociación, definiendo su tipo, sus propiedades (incluyendo el array con las entidades relacionadas) y un ejemplo de una asociación (además se ha añadido a entidades su relación con asociaciones)
- **Definir id de asociación:** se ha definido el *associationId* de forma que pueda usarse como parámetro
- **Definir name de los elementos:** se han definido los *elementName* de entidades, personas y asociaciones, de esta forma puedes usarse como parámetros en los diferentes *Endpoints*
- **Definir los elementRelParam:** se han definido los parámetros con elementos relacionados de personas, entidades y asociaciones, permitiendo elegir en la API que ítems están relacionados con el elemento y definiendo un enumerador con el tipo de elemento (Esto incluye añadir en entidades, el nuevo elemento de asociación)
- **Completar esquema de seguridad:** Se ha añadido la asociación a los permisos de escritura y lectura de los diferentes elementos, lo que permite definir el alcance de lo que puede realizar cada tipo de usuario.

[RoutesAssociations.php](#) (✚ New)

- **Definir las rutas:** Se han definido las rutas relacionadas con el manejo de las asociaciones en la API y como interactúan con los controladores de asociaciones para gestionar las operaciones CRUD, incluyendo su relación con las entidades

[RoutesEntities.php](#) (📝 Edit)

- **Añadir rutas de asociación:** Se han definido nuevas rutas para poder gestionar las operaciones CRUD con las asociaciones relacionadas a la entidad

[Association.php](#) (✚ New)

- **Definición de la clase:** Se ha definido la clase asociación con sus atributos (Se han usado los mismos atributos que para el resto de las clases) y constructor, de

forma que pueda ser utilizado por el ORM de doctrine para mapear la clase a una tabla en la base de datos.

- **Añadir relación:** Se ha añadido la relación *ManyToMany* con la entidad, siendo esta inversa, de forma que entidad es la propietaria de la relación, además se ha añadido una colección de entidades
- **Añadir métodos de gestión de entidades:** Se han añadido los métodos que permiten realizar las acciones de devolver entidades que participan en la asociación, ver si existe una entidad específica y añadir o eliminar esa entidad. En este caso como la entidad es propietaria de la relación, se elimina el elemento de la colección de entidades de la asociación, y además llama al método de añadir o eliminar asociación ligada a la entidad que hemos añadido o eliminado de la relación.
- **Añadir métodos de serialización:** Se han añadido los métodos que devuelven una representación en forma de cadena de caracteres de la asociación incluyendo las entidades relacionadas. Se ha añadido el método que devuelve una representación serializable en formato JSON de la asociación, también incluyendo las entidades relacionadas

[Entity.php](#) (📝 Edit)

- **Añadir nueva relación:** Se ha añadido una nueva relación *ManyToMany* con la clase asociación (*entity_participates_association*), siendo entidad la propietaria y por tanto la que tiene la clave primaria, además de añadirse una colección de elementos de asociación
- **Añadir métodos de gestión de asociación:** Se han añadido los métodos que permiten realizar las acciones de devolver las asociaciones en las que participa la entidad, además de ver si una asociación específica existe con esa entidad y añadir o eliminar una asociación ligada a esa entidad. En este caso se actúa únicamente sobre los de entidad ya que es la propietaria de la relación
- **Complementar métodos de serialización:** Se ha añadido las asociaciones al método *toString* heredado de la clase *Element*. Se ha añadido las asociaciones al método *jsonSerializable* de forma que se incluya la colección cuando se convierta el elemento en formato JSON

[AssociationFactory.php](#) (+ New)

- **Definición de la factoría de asociación:** Se ha definido la clase para crear instancias de la clase asociación (*Association.php*).

[AssociationQueryController.php](#) (+ New)

- **Definición de la clase:** Se ha creado la clase controlador *QueryController* que permite manejar las operaciones de consulta sobre las asociaciones. Incluye la definición de la ruta base de la API sobre la que se construirán las rutas de los *Endpoints*

- **Definición de métodos:** Se han definido los métodos para devolver el tag, nombre e id de la asociación

[AssociationCommandController.php](#) (+ New)

- **Definición de la clase:** Se ha creado la clase controlador *CommandController* que permite manejar las operaciones de escritura sobre las asociaciones. Incluye la definición de la ruta base de la API sobre la que se construirán las rutas de los *Endpoints* (Esta hereda de *element* que permite el añadir, eliminar o actualizar el elemento)
- **Definición de métodos:** se han creado los métodos para devolver el tag, nombre e id de la asociación.

[AssociationRelationController.php](#) (+ New)

- **Crear clase de controlador de relaciones:** Se ha creado la clase para controlar las relaciones de asociación, se han añadido los métodos para devolver el id, nombre y tag de asociación definidos en las *QueryController*
- **Añadir métodos de gestión de relaciones:** Se han añadido los métodos para manejar las relaciones con entidades. Estos métodos se encargan de manejar las rutas y utilizar doctrine y los *QueryController* para realizar las acciones contra la base de datos respecto a las relaciones de la asociación, incluyendo recoger todos los elementos relacionados, añadir o eliminarlo.

[PersonRelationController.php](#) (📝 Edit)

- **Completar métodos para gestionar relaciones:** Se ha añadido el contenido de los métodos para manejar las relaciones con entidades y productos. Estos métodos se encargan de manejar las rutas y utilizar doctrine y los *QueryController* para realizar las acciones contra la base de datos respecto a las relaciones de la persona, incluyendo recoger todos los elementos relacionados, añadir o eliminarlo.

[EntityRelationsController.php](#) (📝 Edit)

- **Completar métodos para gestionar relaciones:** Se ha añadido el contenido de los métodos para manejar las relaciones con personas, productos y asociaciones. Estos métodos se encargan de manejar las rutas y utilizar doctrine y los *QueryController* para realizar las acciones contra la base de datos respecto a las relaciones de la entidad, incluyendo recoger todos los elementos relacionados, añadir o eliminarlo.

[EntityTest.php](#) (📝 Edit)

- **Añadir nueva relación:** Se han añadido las pruebas para añadir o eliminar una asociación relacionada a la entidad

[EntityRelationsControllerTest.php](#) (📝 Edit)

- **Añadir test de nueva relación:** Se han añadido los métodos para probar todos los *Endpoints* de la relación de asociación con la entidad incluyendo el devolver todas las asociaciones, eliminar o añadir una o devolver solo la que coincide con el id. Además, se han añadido las pruebas para las rutas de las posibles excepciones

[AssociationTest.php](#) (✚ New)

- **Definir la clase de pruebas:** Se ha creado la clase para realizar las pruebas unitarias sobre la nueva entidad de asociación, en ella se prueba el constructor, los métodos para recoger y modificar los atributos de la clase (*Getters* y *Setters*), para serializar la clase y para comprobar las acciones sobre su relación con Entidad, teniendo en cuenta que Entidad es la propietaria de la relación *ManyToMany*

[AssociationControllerTest.php](#) (✚ New)

- Definir la clase de pruebas del controlador: Se ha creado la clase para realizar pruebas unitarias de las operaciones CRUD sobre los *Endpoints* de la API. El recoger todas las asociaciones, coger una específica por nombre o id, actualizarla, eliminarla o crearla, incluyendo, además, las pruebas de las posibles excepciones que puedan surgir

[AssociationRelationsControllerTest.php](#) (✚ New)

- **Definir clase de pruebas de relaciones:** Se ha creado la clase para realizar las pruebas de los *Endpoints* de la relación de asociación con las entidades, incluyendo los métodos que se ejecutarán antes de las pruebas (como el *login* del usuario), el devolver todas las entidades, eliminar o añadir una entidad o devolver solo la que coincide con el id. Además, se han añadido las pruebas para las rutas de las posibles excepciones que puedan ocurrir

Problemas encontrados

El principal problema que he encontrado ha sido en la definición de las relaciones, no tenía en cuenta cuando era propietaria o no, y por tanto al borrar o añadir un elemento de la relación por la API, la operación salía como correcta, pero no había persistencia. Una vez me di cuenta de porque daba el error pude corregir fácilmente las clases y tenerlo en cuenta para los métodos de *addEntity* y *removeEntity*.

Mas tarde he vuelto a tener el mismo problema creando la prueba de *testGetAddContainsRemoveEntities* de Asociación, pero sabiendo que el problema podría ser la propiedad de la relación ha sido más sencillo resolverlo

El resto de problemas han sido errores menores al escribir una palabra, sencillos de solucionar.

Conclusiones

La realización de esta práctica me ha permitido adquirir conocimientos sobre el desarrollo de *APIs RESTful* mediante el uso de *frameworks* como Slim y herramientas de mapeo objeto-relacional (ORM) como Doctrine

Ha resultado especialmente útil para adquirir buenas prácticas en la definición de rutas para los *endpoints*.

Por ejemplo, he aprendido a estructurar adecuadamente las rutas, distinguiendo las acciones mediante los métodos HTTP (POST, GET, PUT, DELETE), en lugar de definir rutas específicas como /añadirEntidad o /modificarEntidad, práctica que he observado en entornos laborales cercanos.

Considero muy útil disponer de esta práctica como referencia y documentación para futuros proyectos en los que deba desarrollar una API o estructurar correctamente un backend.