DESARROLLO WEB

Evaluación Final: Recuperación 2 EV

Introducción

Para la entrega crea una carpeta nombre_apellido que contenga el proyecto. No olvidarse de quitar todo lo de la carpeta composer y migrations.

Para poder evaluar voy a recuperar las dependencias de composer, luego crear el modelo de datos y luego ejecutar.

Vamos a realizar una HTTP REST API para la compra de pizzas online (4VPizza).

Peso de cada apartado:

- API de Pizzas: 4 Puntos

- API de Pedidos: 5 Puntos

- Postman: 1 Punto

Durante el examen estás obligado a estar permanentemente conectado/a a la red de alumnos de Cuatrovientos, y terminantemente prohibido usar cualquier herramienta o sistema de comunicación, únicamente te deberás centrar en realizar el examen.

Se debe de realizar el proyecto desde 0.

El API REST servirá para obtener la lista de pizzas con su información detallada que están a la venta. También sirve para gestionar la compra de esas pizzas en un pedido.

Por lo tanto tendremos API:

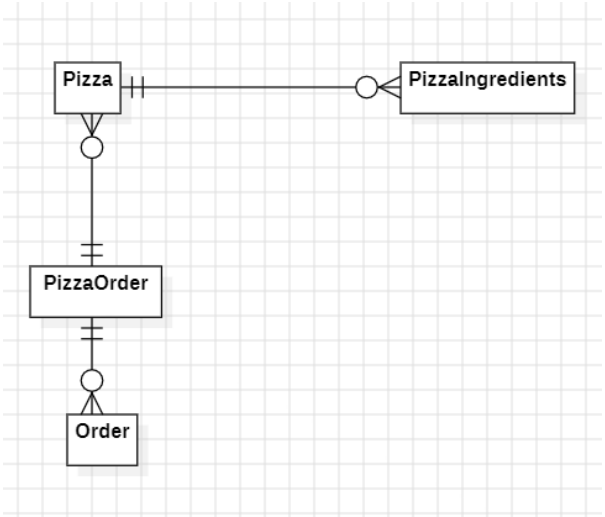
- Pizzas: Para poder visualizar la información de las pizzas. Cada pizza puede tener 1-N ingredientes.
- Pedidos: Para poder hacer un pedido de pizzas. Que incluye la información para el reparto y la información del pago.

Se adjunta el fichero SWAGGER

Se valorará:

- Arquitectura de la solución. Tiene que haber controladores/entidades/repositorios/modelos
- Nombrado coherente de los componentes de código
- Validación correcta de los datos
- Persistencia de datos utilizando ORM/Doctrine.

El modelado E/R será el siguiente:



1. API de Pizzas (5 Puntos)

El objetivo de esta API es tener toda la información de las pizzas que se pueden vender (1,5 Puntos).

Se debe incluir la lista de ingredientes de cada una de las pizzas (1,5 Punto)

Tenemos que indicar si la pizza es apta para celíacos. Este dato se obtiene a partir de los datos de los ingredientes. Para que una pizza sea apta para celíacos entonces todos sus ingredientes tienen que ser aptos para celíacos. (1 punto)

Se podrán buscar las pizzas por 2 parámetros

- El nombre, siendo una búsqueda de tipo contains. O sea no tiene porque ser el nombre exacto. (0,5 Punto)
- Los ingredientes separados por comas. Cada ingrediente no tiene porqué ser con el nombre exacto. Si tiene uno de los ingredientes de la lista es suficiente. Por ejemplo: “queso, pimiento” y la pizza tiene pimiento verde entonces se incluye en el resultado. (0,5 Punto)

Realizar esto por niveles de dificultad. O sea incluyendo relaciones paulatinamente. Mi recomendación es sólo trabajar con las pizzas y luego incluir la lista de ingredientes.

Seguir la especificación del swagger al pie de la letra.

Si os quedáis en la capa de controlador algo os contará, pero obviamente mucho menos que si llegáis hasta la BBDD.

2. API de Pedidos (4 Puntos)

El objetivo de esta API es poder realizar pedidos de pizzas.

Es una única operación POST, que nos va a permitir guardar toda la información de un pedido.

El pedido puede contener 1..N pizzas y cada una de ellas con una cantidad (1 Punto).

El pedido tiene una hora del pedido, una dirección y una información de pago (0,5 Punto)

Por lo tanto la relación entre pedidos y pizzas es N->M

Se deben de realizar las siguientes validaciones:

- Las pizzas deben de estar dados de alta en el sistema (0,5 Puntos)
- La dirección, la hora y el pago son campos obligatorios.(0,5 Puntos)
- El tipo de pago, es credit_card o bizum (0,5 Puntos)
    - La información del pago en credit-card, debe de ser un número de tarjeta con el formato XXXX-XXXX-XXXX-XXXX.
    - La información del pago de bizum debe de ser un número de teléfono de formato XXXXXXXXX

Cualquier tipo de error de validación, debe de devolver un error HTTP 400, con código de error y descripción.

Debe devolver un ID del pedido, con la información detallada de las pizzas pedidas. (1 Punto)

Por favor sean estrictos con el SWAGGER.

3. Postman (1 Punto)

Se debe de crear una colección en POSTMAN que se entregará exportando el fichero e introduciendolo en el resultado dentro de un directorio.

Así se puede probar correctamente este API.

Este POSTMAN deberá tener todas las llamadas a la API, para poder ir probando nuestro desarrollo.