DESARROLLO WEB

Evaluación 2: Examen final

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 gestión de recetas útiles para estudiantes de 4V (4VChef).

Peso de cada apartado:

- API de Recetas: 6 Puntos
- API de Nutrientes: 3 puntos
- Postman: 1 Punto
- SECURIZACIÓN de las API: 1 punto-extra, solo a utilizar si hemos llegado al 5.

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 gestionar recetas de cocina, donde una **receta** se compone de **ingredientes**, **pasos** y **valor nutritivos**

Por lo tanto tendremos API:

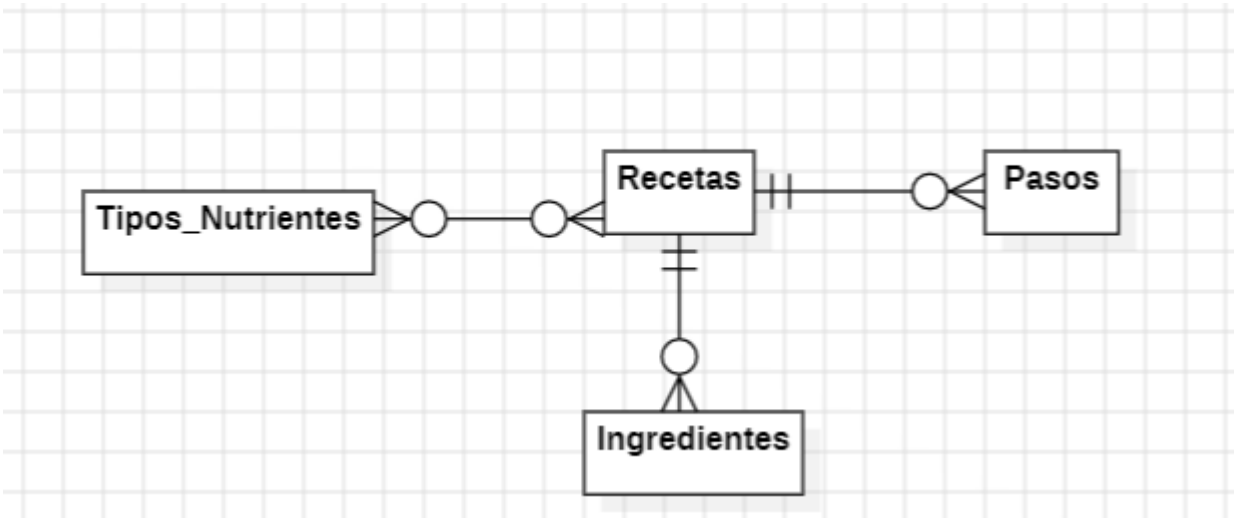
- Tipos-Nutrientes: Para obtener la lista de nutrientes que pueden tener las recetas
- Recetas: Para obtener/crear recetas.

Se adjunta el fichero SWAGGER y tenemos que ser **estrictos** con la especificación.

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:



Por lo tanto se intuye que la BBDD que lo soporte contendrá lo siguiente:

- Tabla de Recetas.
- Tabla de Ingredientes (Relación 1-M sobre recetas).
- Tabla de Pasos (Relación 1-M sobre recetas)
- Tabla de TiposNutrientes. Donde se dan de alta los tipos de nutrientes
- Tabla de Recetas-Nutrientes (Relación N-M entre Nutrientes y Recetas)

1. API de Recetas (6 Puntos)

Realizar esto por niveles de dificultad. O sea incluyendo relaciones paulatinamente

- GET de recetas sin ingredientes ni pasos (1 punto)
- GET de recetas con ingredientes (1 punto)
- GET de recetas con pasos (1 punto)
- POST de recetas sin ingredientes ni pasos (0,5 punto)
- POST de recetas con ingredientes (1,25 puntos)
- POST de recetas con pasos (1,25 puntos)

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

Para crear nuevas recetas hay que hacer una serie de validaciones:

- Tienes al menos 1 Ingrediente
- Tienen al menos 1 Paso.

Si una de estas validaciones es incorrecta se dará el error correspondiente, fijaros en la especificación.

2. API de nutrientes (3 Puntos)

Los nutrientes que van a tener las recetas son en sí una entidad propia que se tiene que gestionar.

Los nutrientes se componen de un nombre y una unidad de medida.

Una receta no puede tener una serie de nutrientes si no están dados de alta anteriormente. La relación entre nutrientes y recetas es N-M.

Por lo tanto se recomienda el uso de una entidad Recetas-Nutrientes que incluya referencias a la receta y a los nutrientes de esa receta y la cantidad de cada nutriente.

Por lo tanto necesitamos los siguientes cambios:

- GET de nutrientes (0,5 puntos)
- GET de recetas incluyendo sus nutrientes. Tendremos un filtro implementado como query params que es minCalories y maxCalories. Calories es un nutriente por lo tanto debe de filtrar las recetas que cumplen ese requisito (1,25 puntos)
- POST de recetas incluyendo sus nutrientes, validando que esté dado de alta el tipo de nutriente incluido en la receta (1,25 puntos)

3. SWAGGER - Postman (1 Punto)

Debe de cumplirse la especificación SWAGGER de la API.

Mi forma de probar el examen es crear una colección POSTMAN a partir de ese SWAGGER y probar API.

Si no se cumple la especificación se irán rebajando décimas de este punto.

4. Seguridad (1 Punto)

Nos han pedido que la API de Reservas esté securizada por un API KEY que le entregamos al cliente de nuestra API.

La API_KEY que hemos dado a nuestro cliente es: 72958417-ee15-42b5-b093-436ccf8ad52c

El cliente deberá incluir como header de nombre apiKey en cada petición a la API de recetas.

Si el APIKEY no es correcto o no esta se da un error HTTP de tipo 403.