AEV2 – Modelo Vista Controlador con Doctrine CRUD

Objetivos

- Demostrar que las bases de PHP y POO están ya consolidadas.
- Demostrar el conocimiento adquirido durante el tema 3 sobre el MVC.
- Demostrar el conocimiento adquirido durante el tema 4 y 5 sobre el ORM de Doctrine.

Recursos generales

- Todas las presentaciones vistas hasta ahora.
- Documentación oficial de PHP y Doctrine.
- Códigos de ejemplo vistos en clase y en actividades previas (AP5-1).
- **No se puede utilizar ninguna IA, si se detecta que se ha utilizado para generar el código se marcará la evaluable como insuficiente.**

Actividad

1. Se debe clonar el repositorio de la invitación de Github Classroom y seguir los pasos del README. Se partirá del código del repositorio y habrá que realizar unas modificaciones para añadir funcionalidad nueva y para modificar parte de la funcionalidad existente.
2. Se debe implementar las especificaciones de autoload de PSR-4 y la instalación de las dependencias se hará mediante Composer.
3. Se deben realizar como mínimo 5 subidas al repositorio de GIT demostrando un progreso adecuado de la aplicación y en esas subidas, debe añadirse un comentario con la descripción de las modificaciones realizadas.

Tareas a realizar

Ejercicio 0 (Configuración inicial - No puntúa)

1. Entrar en la invitación de Github Classroom, seleccionar tu nombre y clonar el repositorio en un nuevo proyecto de PHPStorm nuevo.
2. Seguir el README de la carpeta raíz para poner en marcha el entorno Docker.
3. Configurar Composer en el proyecto:
   - Modificar e instalar las dependencias del fichero `composer.json`.
   - Utilizar el namespace propuesto (AEV2) y usarlo en el código.
4. Verificar que el proyecto funciona correctamente:
   - Ejecutar el servidor y hacer funcionar: `http://localhost:8020`

Ejercicio 1 (Modelado completo de la base de datos)

Crear las entidades y repositorios para todas las tablas: Debes crear las clases necesarias para **todas las tablas** de la base de datos `empresa`:

Nomenclatura:

- Cada entidad deberá ser nombrada con el nombre en inglés y singular de la tabla.
- Los repositorios tendrán el nombre de la entidad seguido del sufijo `Repository`.
- La PRIMARY KEY de cada tabla siempre se llamará `id` en las entidades PHP.

Configuración de atributos:

- Todas las PRIMARY KEY deben indicar el atributo `#[Id]` y `#[GeneratedValue]`, configurando correctamente su estado para funcionar con la base de datos existente.
- Los datos DECIMAL de la BD deben usarse como tipo `float` en PHP y `type: "decimal"` en el modelado de Doctrine, incluyendo todos los atributos necesarios (precision, scale).
- Cuando se configure cada columna o asociación, se debe indicar siempre el nombre (`name`) y tipo de dato (`type`), aunque por defecto no fuera necesario.
- El modelado debe ser lo más fiel posible a la tabla: valores nulos, valores únicos, valores por defecto, enums, etc.

Métodos y constructor:

- Añadir los getters y setters de cada atributo según corresponda, click derecho Generate…
- Configurar correctamente el constructor de cada entidad.

Asociaciones:

- **Todas las asociaciones deben ser bidireccionales.**

Ejercicio 2 (Vistas y Controladores - Consulta y CRUD Productos)

Implementar el sistema completo de vistas y controladores siguiendo el patrón MVC.

Completar la ruta main (`/`) añadiendo un nav con todos los listados disponibles y que aparezca en todas las futuras vistas.

2.1. Consulta (Solo lectura)

Implementar listados de **solo consulta** para las siguientes entidades:

Clientes (`/clientes`):

- Listado completo de todos los clientes con toda su información.
- Mostrar el nombre del representante (empleado) si tiene asignado.

Pedidos (`/pedidos`):

- Listado de todos los pedidos.
- Link para acceder a los detalles de cada pedido.

Detalle de Pedido (`/pedido/read/{id}`):

- Muestra información completa del pedido y todos sus detalles (productos).

Empleados (`/empleados`):

- Listado completo de todos los empleados con toda su información.
- Mostrar el departamento al que pertenecen.
- Mostrar el jefe (si tiene).
- Mostrar el número de clientes que tienen asignados.

Departamentos (`/departamentos`):

- Listado completo de todos los departamentos.
- Mostrar el número de empleados de cada departamento.

2.2. Operaciones CRUD (Solo para PRODUCTO)

Implementar el **CRUD completo** únicamente para la entidad **PRODUCTO**:

Create (Crear) (`/producto/create`):

- Formulario para crear nuevos productos.
- **Importante**: El ID del producto NO es autogenerado, debe proporcionarse manualmente.

Read (Leer) (`/productos`):

- Listado completo de todos los productos.
- Muestra ID y descripción de cada producto.

- Muestra el listado de pedidos relacionados con los productos.

Update (Actualizar) (`/producto/update/{id}`):

- Formulario pre-rellenado para editar productos existentes.
- El ID no se puede modificar (campo de solo lectura).
- Solo se puede actualizar la descripción.

Delete (Eliminar) (`/producto/delete/{id}`):

- Pantalla de confirmación antes de eliminar.
- Muestra advertencia visual.
- Requiere confirmación explícita del usuario.
- Mostrar aviso si existe un error al eliminar (por ejemplo, si tiene detalles de pedido asociados).

Ejercicio 3 (CRUD de Empleados y Departamentos)

Implementar el **CRUD completo** para **Empleados y Departamentos**.

3.1. CRUD de Empleados

Implementar todas las operaciones CRUD para la entidad de Empleado:

Create (`/empleado/create`):

- Formulario con **selects** para elegir:
  - Departamento (obligatorio)
  - Jefe/Manager (opcional, select con lista de empleados)
- Campos: ID, Apellidos, Oficio, Fecha Alta, Salario, Comisión.
- Validación: un empleado no puede ser su propio jefe.

Update (`/empleado/update/{id}`):

- Formulario pre-rellenado con **selects** para:
  - Departamento (obligatorio)
  - Jefe/Manager (opcional)
- El ID no se puede modificar (campo de solo lectura).
- Todos los campos del empleado pueden ser actualizados.
- Validación: un empleado no puede ser su propio jefe.

Delete (`/empleado/delete/{id}`):

- Pantalla de confirmación antes de eliminar.
- Verificar que no tenga subordinados o clientes asignados antes de eliminar.
- Mostrar advertencia si tiene relaciones.

3.2. CRUD de Departamentos

Implementar todas las operaciones CRUD para la entidad de Departamento:

Create (`/departamento/create`):

- Formulario para crear nuevos departamentos.
- Campos: ID (manual), Nombre, Localización, Color.
- **Importante**: El ID del departamento NO es autogenerado, debe proporcionarse manualmente.

Update (`/departamento/update/{id}`):

- Formulario pre-rellenado para editar departamentos existentes.
- El ID no se puede modificar (campo de solo lectura).
- Todos los campos del departamento pueden ser actualizados.

Delete (`/departamento/delete/{id}`):

- Pantalla de confirmación antes de eliminar.
- Verificar que no tenga empleados asignados antes de eliminar.
- Mostrar advertencia si tiene empleados.
- Requiere confirmación explícita del usuario.

Ejercicio 4 (CRUD de Clientes y Pedidos)

Implementar el **CRUD completo** para Clientes y Pedidos.

4.1. CRUD de Clientes

Implementar todas las operaciones CRUD para la entidad de Cliente:

Create (`/cliente/create`):

- Formulario para crear nuevos clientes.
- Formulario con **select** para elegir representante (empleado, opcional).
- Campos: ID (manual), Nombre, Dirección, Ciudad, Estado, Código Postal, Área, Teléfono, Límite de Crédito, Observaciones.
- **Importante**: El ID del cliente NO es autogenerado, debe proporcionarse manualmente.

Update (`/cliente/update/{id}`):

- Formulario pre-rellenado para editar clientes existentes.
- Formulario con **select** para elegir representante (empleado, opcional).
- El ID no se puede modificar (campo de solo lectura).
- Todos los campos del cliente pueden ser actualizados.

Delete (`/cliente/delete/{id}`):

- Pantalla de confirmación antes de eliminar.
- Mostrar número de pedidos asociados.
- Verificar que no tenga pedidos antes de eliminar.
- Mostrar advertencia si tiene pedidos asociados.
- Requiere confirmación explícita del usuario.

4.2. CRUD de Pedidos

Implementar todas las operaciones CRUD para la entidad Pedido:

Create (`/pedido/create`):

- Formulario para crear nuevos pedidos.
- Formulario con **select** para elegir cliente (obligatorio).
- Formulario con **select** para elegir tipo de pedido (A, B, C, opcional).
- Campos: ID (manual), Fecha Pedido, Tipo (A, B, C), Fecha Envío, Total.
- **Importante**: El ID del pedido NO es autogenerado, debe proporcionarse manualmente.

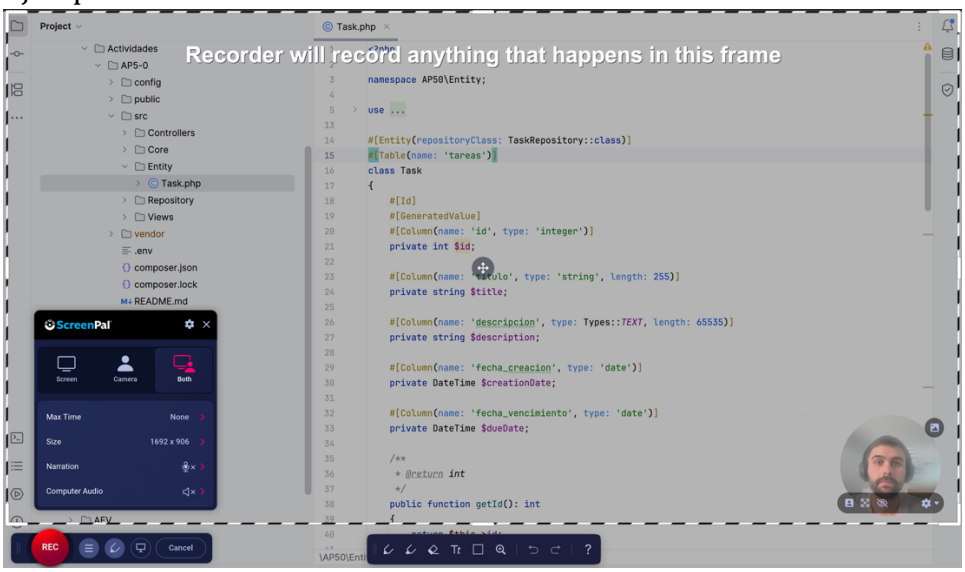
Update (`/pedido/update/{id}`):

- Formulario pre-rellenado para editar pedidos existentes.
- Formulario con **select** para elegir cliente (obligatorio).
- Formulario con **select** para elegir tipo de pedido (A, B, C, opcional).
- El ID no se puede modificar (campo de solo lectura).
- Todos los campos del pedido pueden ser actualizados.

Delete (`/pedido/delete/{id}`):

- Pantalla de confirmación antes de eliminar.
- Mostrar número de detalles asociados.
- Verificar que no tenga detalles antes de eliminar.
- Mostrar advertencia si tiene detalles asociados.
- Requiere confirmación explícita del usuario.

Video explicativo (OBLIGATORIO)

- Realizar un video explicando el modelado de cada una de las entidades, indicando el porqué de cada uno de los atributos configurados.
- **Duración: máximo 10 minutos.**
- El video debe crearse con **ScreenPal** y proporcionar el link.
- **Requisitos del contenido del video:**
  - Añadirte la cara y la pantalla.
- Ejemplo:
- 

Entregables

1. **Archivo `.zip`:**
   - Todo el proyecto comprimido **sin la carpeta `/vendor`**.
   - Esta versión debe coincidir con la que esté en GitHub.
2. **Repositorio GitHub:**
   - Realizar **mínimo 5 commits** al repositorio proporcionado por el profesor.
   - Los commits deben demostrar un progreso adecuado del proyecto.
3. **Actualizar este README.md con:**
   - Explicación de los cambios realizados en cada apartado.
   - Problemas encontrados y cómo se resolvieron.
   - Link para acceder al vídeo explicativo.
   - Autorúbrica cumplimentada.

Rúbrica de evaluación

1. Insuficiente (< 5) - Este nivel es excluyente

Si no se cumplen TODOS los puntos, la actividad se considera suspendida con una nota máxima de 4.

- Realizar el Ejercicio 0 correctamente (configuración inicial).
- Realizar el Ejercicio 1 completo:
  - Modelar **todas las tablas** como entidades de Doctrine.
  - Configurar correctamente todas las columnas con sus tipos y restricciones.
  - Las PRIMARY KEY deben estar configuradas correctamente.
- Entregar el video.
- Realizar **mínimo 5 commits** al repositorio.
- El proyecto debe ejecutarse sin errores en la ruta raíz `/`.

2. Suficiente (5 - 5.9)

- Cumplir **todos los puntos de INSUFICIENTE**.
- Completar el Ejercicio 2:
  - Implementar consulta (solo lectura) de: Clientes, Pedidos, Detalle de Pedido.
  - Implementar listado (solo lectura) de: Empleados y Departamentos.
  - Implementar **CRUD completo** de Productos (Create, Read, Update, Delete).
  - Todas las rutas de consulta funcionando correctamente.

3. Notable (6 - 8.9)

- Cumplir **todos los puntos de SUFICIENTE**.
- Completar el Ejercicio 3:
  - Implementar **CRUD completo** de Empleados (con selects para departamento y jefe).
  - Implementar **CRUD completo** de Departamentos.
- Las asociaciones bidireccionales de **DEPT**, **EMP** y **CLIENTE** deben estar implementadas.
- Completar el Ejercicio 4.1:
  - Implementar **CRUD completo** de Clientes (con select para representante).

4. Excelente (9 - 10)

- Cumplir **todos los puntos de NOTABLE**.
- Completar el Ejercicio 4.2:
  - Implementar **CRUD completo** de Pedidos (con select para cliente).
- Control de errores y validaciones completas.

- **Todas las asociaciones bidireccionales** de todas las tablas deben estar implementadas y funcionando.

5. EXTRAS (Estos puntos suman o restan en cualquiera de los cuatro niveles)

Aspectos que suman:

- No debe faltar ningún getter o setter necesario, ni estar ninguno que no sea necesario.
- Los nombres de las clases deben ser en inglés y seguir las convenciones.
- Se debe cumplir el nombre de las variables y repositorios según las normas previas.
- Debe configurarse los registros de la BD lo más fiel posible (nullable, unique, default, etc.).
- Las asociaciones deben ser bidireccionales.
- El tipeado de las variables en PHP y en Doctrine debe ser el correcto.
- Contenido del video debe ser entendible y adecuado a lo solicitado.
- Incluir la hoja de autorúbrica rellena y argumentada.
