EXAMEN TRANSVERSAL FINAL

Resultados de Aprendizaje:

1.1. Aplica los principios del desarrollo backend y su rol en la arquitectura de una aplicación de software.
1.2. Utiliza el lenguaje de programación backend para interactuar con el servidor web.
2.1. Construye un servicio API, incluyendo su estructura y patrones de diseño.
2.2. Utiliza el framework seleccionado para construir un servicio y APIs (Interfaz de Programación de Aplicaciones) que interactúen con el frontend y gestionen datos en una base de datos.
3.1. Utiliza el framework definido para construir operaciones CRUD (Crear, Leer, Actualizar y Eliminar), manipulando datos mediante la aplicación.

En esta situación evaluativa, se espera evidenciar que los estudiantes:

Indicador 1.1.1: Identifica los componentes principales de una arquitectura de backend y su función en el desarrollo de una aplicación de software.

Indicador 1.2.1. Diseña solicitudes HTTP donde el servidor responde a las solicitudes del cliente de manera precisa y eficiente.

Indicador 2.1.1. Diseña API de manera funcional y cumpliendo con las especificaciones técnicas y de diseño.

Indicador 2.2.3. Implementa APIs que permiten al frontend interactuar con el servicio web, siguiendo un diseño consistente y estando documentadas de forma clara.

Indicador 3.1.1. Construye operaciones CRUD (Crear, Leer, Actualizar y Eliminar) de manera eficiente y precisa para acceder y modificar datos en la aplicación.

Indicador 3.2.2. Optimiza el uso de las operaciones CRUD para mejorar el rendimiento y la eficiencia de la aplicación.

Tiempo: según se indique en la plataforma.
Puntaje: 100 puntos.

Características de la evaluación

- Esta es una actividad calificada de la unidad y es de carácter grupal hasta 3 personas.
- Para conocer cómo serás evaluado, revisa la rúbrica disponible en plataforma.

- Esta es una actividad evaluativa de desarrollo extendida donde se debe presentar una propuesta para responder a la necesidad de un caso de un cliente real.

- Esta actividad evaluativa implicará la entrega de un producto que deberá cargar a la plataforma.

Instrucciones

Para desarrollar esta evaluación, deberá basarse en el caso que presenta una necesidad a cubrir por un cliente:

TodoCamisetas

La empresa TodoCamisetas, fundada en 2023 y con sede en Santiago, Chile, es un proveedor mayorista especializado en camisetas de fútbol de clubes nacionales e internacionales. Sus fundadores, Matías López y Camila Fuentes, detectaron que muchos vendedores locales carecían de acceso directo a productos de calidad a precios competitivos, por lo que decidieron crear un modelo de negocio basado en ventas B2B a tiendas minoristas en varias regiones del país.

Actualmente, TodoCamisetas cuenta con dos clientes principales: Tienda 90minutos y la Tienda tdeportes. Para sostener esta operación, TodoCamisetas requiere desarrollar una **API RESTful en PHP puro** (sin frameworks) que sirva como columna vertebral de su sistema de gestión de inventario y de relación con clientes.

La API debe contar con CRUD para:

1. **Gestión de Camisetas (stock):**
    - CRUD completo de productos, donde cada “camiseta” tiene atributos como:
        - **Título:** Nombre descriptivo (p. ej., “Camiseta Local 2025 – Selección Chilena”)
        - **Club:** Nombre del club al que pertenece (p. ej., “Selección Chilena”)
        - **País:** Procedencia del diseño (p. ej., “Chile”, “España”).
        - **Tipo:** Clasificación del modelo (p. ej., “Local”, “Visita”, “3era Camiseta”, “Femenino Local”, “Niño”).
        - **Color:** Combinación principal de colores (p. ej., “Blanco y Negro”).
        - **Precio:** Valor base en pesos chilenos (p. ej., 45 000).
        - **Tallas disponibles:** Una o varias tallas asociadas (ej.: “S”, “M”, “L”, “XL”).
        - **Detalles:** Texto adicional (ej.: “Edición aniversario 2025”).
        - **Código de Producto:** SKU único (ej.: “SCL2025L”).
2. **Gestión de Clientes:**
    - CRUD completo de clientes B2B, almacenando:

- **Nombre Comercial:** (p. ej., “90minutos”).
- **RUT o ID Comercial:** Identificador tributario.
- **Dirección:** Ciudad y región (p. ej., “Providencia, Santiago”).
- **Categoría:** “Regular” o “Preferencial”.
- **Contacto:** Nombre y correo del encargado de compras.
- **Porcentaje de oferta:** Campo que permite obtener un descuento en porcentaje para los clientes en todos los productos.

3. **Logística de Precios de Oferta:**
   - La API debe calcular dinámicamente el **precio final** de una camiseta según el cliente que realiza la consulta.
     - Si el cliente_id corresponde a “90minutos” (Categoría Preferencial) y la camiseta tiene precio_oferta definido, entonces el sistema debe devolver, al consultar el recurso, el campo precio_final igual a dicho precio_oferta.
     - Si el cliente_id corresponde a “tdeportes” (Categoría Regular) o no existe ninguna oferta para esa camiseta, el sistema debe devolver precio_final igual a precio (precio base).

Tareas a Ejecutar

Tarea 1 - Identifica los componentes principales de una arquitectura de backend y su función en el desarrollo de una aplicación de software

Diseño de la arquitectura de archivos. Trace un diagrama de alto nivel (puede ser un screenshot) que muestre la estructura de directorios completa del proyecto (Vistas, controladores, rutas, modelos, etc) y en en un breve documento (o mediante comentarios) explique cómo encaja cada componente en el proyecto.

Diseña un modelo de datos sencillo que permita almacenar los datos cubriendo la relación entre ambas entidades

Tarea 2 - Diseña solicitudes HTTP donde el servidor responde a las solicitudes del cliente de manera precisa y eficiente.

Enrutamiento en PHP puro. Implemente un enrutamiento con expresiones regulares que, según la ruta, llame a métodos estáticos de controladores.

```
$method = $_SERVER['REQUEST_METHOD'];

if ($method === 'GET'){ //Código }
```

Documente cada ruta en el informe con el detalle que indique:

- La expresión regular usada.
- El método HTTP.
- El propósito del endpoint

En los controlares, defina métodos estáticos como void, show, store, update, destroy y valide los datos obligatorios del request en cada caso.

Tarea 3 - Diseña API de manera funcional y cumpliendo con las especificaciones técnicas y de diseño.

Implementación de modelos. En models, defina los metodos que concuerden con cada tabla en la Dase de Datos. Por ejemplo.

```
public static function update(int $id, array $data): bool {
    /* UPDATE WHERE id = :id */
}
/**
    * Elimina la camiseta con ID dado.
    * @param int $id
    * @return bool
*/
```

Definición y documentación de endpoints RESTful. Con la ayuda de swagger, Debe tener los detalles de:

- Método HTTP.
- Ruta exacta (incluyendo parámetros de ruta o query).
- Request (ruta, query o JSON en body).
- Response: JSON mínima.
- Errores: Detalle de cada error y codigo asociado.

Tarea 4 - Implementa APIs que permiten al frontend interactuar con el servicio web, siguiendo un diseño consistente y estando documentadas de forma clara.

Endpoints consumibles desde un cliente HTTP. Implemente los metodos en cada contralador asegurándose de devolver siempre la cabecera HTTP Content-Type: application/json

Genera Una colección en Postman detallada para consumir cada servicio con ejemplos de Request y Response detallada y lista para importar.

Tarea 5 - Construye operaciones CRUD (Crear, Leer, Actualizar y Eliminar) de manera eficiente y precisa para acceder y modificar datos en la aplicación.

Construye operaciones CRUD para Camisetas y Clientes, donde existan validaciones como, por ejemplo, que no se pueda eliminar clientes con camisetas. Se debe poder Listar camisetas por clientes, Crear, Actualizar y Eliminar Camisetas al igual que Clientes.

Endpoints para gestionar tallas. Construye operaciones CRUD para la gestión de tallas y permita la relación de muchos a muchos.

Tarea 6 - Optimiza el uso de las operaciones CRUD para mejorar el rendimiento y la eficiencia de la aplicación.

Endpoint para obtener camiseta con precio final. Procura generar las reglas de negocio que permita poder obtener el precio final de cada camiseta asociado a alguna oferta en particular del cliente consultado.

Formato de entrega

- Comprima su proyecto en un archivo .zip (no se aceptará otro formato de compresión, siga las instrucciones) llamado eva3_xyz.zip donde xyz es el nombre de su grupo.
- Deberá colocar un nombre a su grupo e indique el nombre de los integrantes en el recuadro de texto de la evaluación. (solo entrega 1 del grupo)
- El nombre a dar al archivo es el siguiente: **Eval_U3A_NOMBRE DEL EQUIPO**