**Informe Evaluación Proyecto Tejelanas Vivi**

**Asignatura BACKEND**

Alumnos:

* Efrén Tovar Silva - Rut 25.698.445-8
* Eduardo Ahumada Catalán - Rut 17.304.258-2

Profesor: Iván Bilbao

**Resumen Técnico de la Evaluación**

Presento la evaluación del proyecto "Tejelanas Vivi", desarrollado para la asignatura BACKEND. Este informe detalla el cumplimiento de los requisitos técnicos, la implementación y la documentación de la API.

**Criterios de Evaluación y Evidencia**

**1.** Documentación CRUD para Múltiples Entidades

Hemos documentado las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) para un mínimo de tres entidades principales: Productos, FAQ y Contacto. La estructura de los endpoints en el archivo swagger.yaml es clara y refleja las operaciones disponibles para cada recurso.

Evidencia:

1. YAML

2. /productos/productos\_DELETE.php:

3. delete:

4. summary: Eliminar producto

5. responses:

6. '200': Producto eliminado correctamente

7.

**2.** Estructura OpenAPI 3.0

La especificación OpenAPI 3.0 está correctamente estructurada, lo que asegura una descripción precisa y estandarizada de la API.

Evidencia:

1. YAML

2. openapi: 3.0.0

3. info:

4. title: Tejelanas Vivi API

5. version: 1.0.0

6.

**3.** Respuestas HTTP Detalladas

Cada endpoint define explícitamente sus posibles respuestas HTTP, incluyendo códigos de estado (200, 400, 404, 500) y sus respectivas descripciones. Esto facilita el entendimiento del comportamiento de la API en diferentes escenarios.

Evidencia:

1. YAML

2. responses:

3. '200':

4. description: Producto actualizado exitosamente

5. '400':

6. description: Datos inválidos

7. '404':

8. description: Producto no encontrado

9. '500':

10. description: Error interno del servidor

11.

**4.** Ejemplos de Respuestas (JSON y XML)

Se han proporcionado ejemplos de respuestas tanto en formato JSON como XML para los endpoints, demostrando la estructura de los datos esperados.

Evidencia:

YAML

1. example:

2. - id: 1

3. nombre: 'Lana Merino'

4. descripcion: 'Lana suave de alta calidad'

5. application/xml:

6. example: |

7. <Producto><id>1</id></Producto>

8.

**5.** Definición de Headers

Hemos incluido la definición de headers relevantes, como Cache-Control y Retry-After, lo que optimiza la gestión de caché y el manejo de límites de peticiones.

Evidencia:

1. YAML

2. headers:

3. Cache-Control:

4. schema:

5. type: string

6. example: max-age=3600

7.

**6.** Esquemas de Datos (Schemas)

Los schemas para las entidades están correctamente definidos en la sección components/schemas, especificando los tipos de datos y los campos obligatorios.

Evidencia:

1. YAML

2. components:

3. schemas:

4. Producto:

5. type: object

6. required:

7. - id

8. - nombre

9.

**7.** Implementación Funcional en PHP

El backend está implementado en PHP, con una lógica clara y funcional para cada operación.

Evidencia:

1. productos\_GET.php (para "Productos"):

2. PHP

3. $stmt = $conn->prepare('SELECT \* FROM productos');

4. $stmt->execute();

5. $productos = $stmt->fetchAll();

6. FAQ\_POST.php (para "FAQ"):

7. PHP

8. if (!isset($data['question']) || !isset($data['answer'])) {

9. http\_response\_code(400);

10. echo json\_encode(['error' => 'Campos requeridos']);

11. }

12. contacto\_DELETE.php (para "Contacto"):

13. PHP

14. $data = json\_decode(file\_get\_contents('php://input'), true);

15. $id = $data['id'];

16. $stmt = $conn->prepare('DELETE FROM contacto WHERE id = ?');

17. $stmt->execute([$id]);

18.

**8.** Validación de Datos en Endpoints

Se ha implementado la validación de datos en los endpoints, asegurando que solo se procesen datos válidos y se devuelvan respuestas apropiadas en caso de errores.

Evidencia:

1. productos\_POST.php:

2. PHP

3. if (!isset($data['nombre']) || empty($data['nombre'])) {

4. http\_response\_code(400);

5. echo json\_encode(['error' => 'Nombre requerido']);

6. }

7.

**9.** Colección Postman Documentada

Hemos proporcionado una colección Postman que documenta y permite probar todas las rutas CRUD, lo que facilita la interacción y verificación de la API.

Evidencia:

1. Tejelanas Vivi API.postman\_collection\_backend.json:

2. JSON

3. "name": "productos\_POST",

4. "request": {

5. "method": "POST",

6. "url": "http://localhost/.../productos\_POST.php"

7. }

8.

**10.** Buenas Prácticas y Estructura del Proyecto

El proyecto sigue buenas prácticas de desarrollo, con archivos PHP separados por entidad y verbo HTTP (ej. productos\_GET.php, FAQ\_DELETE.php). Además, se incluye un README que describe la estructura general del proyecto.

**Conclusión Final**

El proyecto "Tejelanas Vivi" cumple exhaustivamente con todos los criterios de la rúbrica de evaluación EVA3 para la asignatura BACKEND. La implementación en PHP es robusta y coherente con la documentación OpenAPI, demostrando un manejo adecuado de validación, errores y una estructura profesional.