RAP17_GA7_AA5_EV03_ITE_ Diseño y Desarrollo de servicios web – proyecto

ELKIN SEBASTIAN GRANADOS GOMEZ

JUAN PABLO HERNANDEZ GUERRA

ARNALDO ANDRES PUSHAINA PUSHAINA

SERVICIO NACIONAL DE APRENDIZAJE SENA

INSTRUCTOR

FICHA

3070294

23 DE NOVIEMBRE 2025

TABLA DE CONTENIDO

INTRODUCCIÓN…….3

OBJETIVOS…….……..……...4

DESARROLLO DE LA ACTIVIDAD……5

GLOSARIO…..…….…….……9

CONCLUSION ...……10

BIBLIOGRAFÍA…...…….…….……11

INTRODUCCIÓN

El diseño y desarrollo de servicios web constituye una etapa fundamental dentro de la arquitectura de software moderna, ya que permite estructurar la lógica del negocio, facilitar la comunicación entre sistemas y garantizar el acceso seguro y escalable a los datos. En el marco del proyecto formativo y siguiendo los lineamientos del componente “Construcción de API”, se implementan las API necesarias para satisfacer los requerimientos funcionales del software en desarrollo.

Esta evidencia comprende la creación, documentación, prueba y versionamiento de los servicios web que soportarán el funcionamiento del sistema, asegurando buenas prácticas como modularidad, uso de estándares HTTP, autenticación, control de errores y elaboración del repositorio con su respectiva estructura técnica.

OBJETIVOS

Diseñar, desarrollar y documentar las API del proyecto formativo aplicando los principios y estándares abordados en el componente formativo “Construcción de API”.

OBJETIVOS ESPECÍFICOS

- Identificar los servicios necesarios de acuerdo con los requisitos del software.
- Codificar las API siguiendo buenas prácticas de arquitectura, seguridad y manejo de datos.
- Implementar documentación técnica clara y estructurada para cada servicio.
- Gestionar el proyecto utilizando herramientas de control de versiones (Git y repositorio remoto).
- Entregar un paquete comprimido con los archivos del proyecto y del repositorio según la norma establecida.

DESARROLLO DE LA ACTIVIDAD

Desarrollo de la Actividad

1. Análisis del software y definición de servicios necesarios

Con base en las características del software del proyecto formativo (según la evidencia GA7 previas), se identificaron los módulos principales. Para cada uno se definieron los servicios necesarios siguiendo el patrón REST:

- **Usuarios**
  - POST /api/usuarios → Crear usuario
  - GET /api/usuarios → Listar usuarios
  - GET /api/usuarios/{id} → Consultar usuario
  - PUT /api/usuarios/{id} → Actualizar usuario
  - DELETE /api/usuarios/{id} → Eliminar usuario
- **Autenticación**
  - POST /api/auth/login → Validar credenciales
  - POST /api/auth/refresh → Renovar token
- **Inventario / Productos (ejemplo para sistemas de inventarios, ventas, etc.)**
  - POST /api/productos
  - GET /api/productos
  - GET /api/productos/{id}
  - PUT /api/productos/{id}
  - DELETE /api/productos/{id}
- **Registro de transacciones**
  - POST /api/transacciones
  - GET /api/transacciones

Cada endpoint fue diseñado considerando:
Parámetros de entrada
Validaciones
Códigos de respuesta HTTP
Estructura del JSON de salida
Manejo de errores

2. Diseño de estructura del proyecto

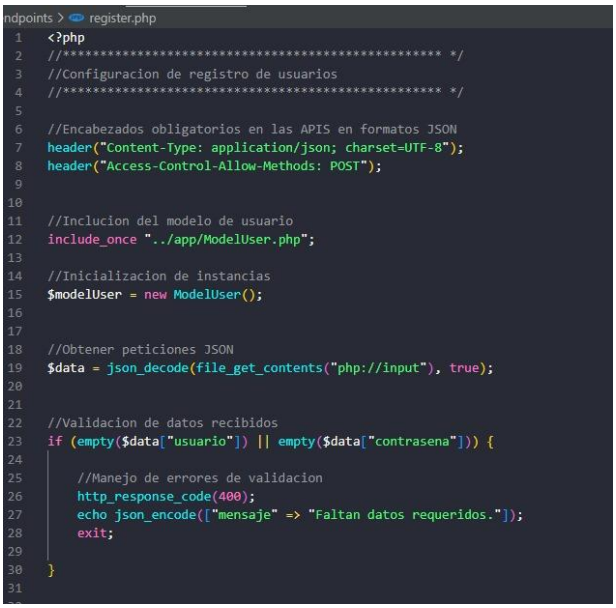
Se creó la estructura recomendada para una API moderna

```
APIREST/
|
├── app/
|   ├── ModelUser.php          # Modelo de usuario con lógica de negocio
|   └── ModelProductos.php     # Modelo de productos con operaciones CRUD
|
├── config/
|   └── db.php                 # Configuración de conexión a base de datos
|
├── endpoints/
|   ├── login.php              # Endpoint de inicio de sesión
|   ├── register.php           # Endpoint de registro de usuarios
|   ├── home.php               # Panel de administración de productos
|   ├── produc/
|   |   ├── listar.php         # Listar todos los productos
|   |   ├── insertar.php       # Crear nuevo producto
|   |   ├── obtener.php        # Obtener producto por ID (POST)
|   |   ├── obtener_get.php    # Obtener producto por ID (GET)
|   |   ├── actualizar.php     # Actualizar producto existente
|   |   ├── eliminar.php       # Eliminar producto (POST)
|   |   └── eliminar_get.php   # Eliminar producto (GET)
|   └── user/
|
├── sql/
|   └── db.sql                 # Script de creación de base de datos
|
├── index.php                  # Archivo principal con información de la API
└── README.md                  # Documentación del proyecto
```

3. Codificación de las API

Ejemplo de un controlador

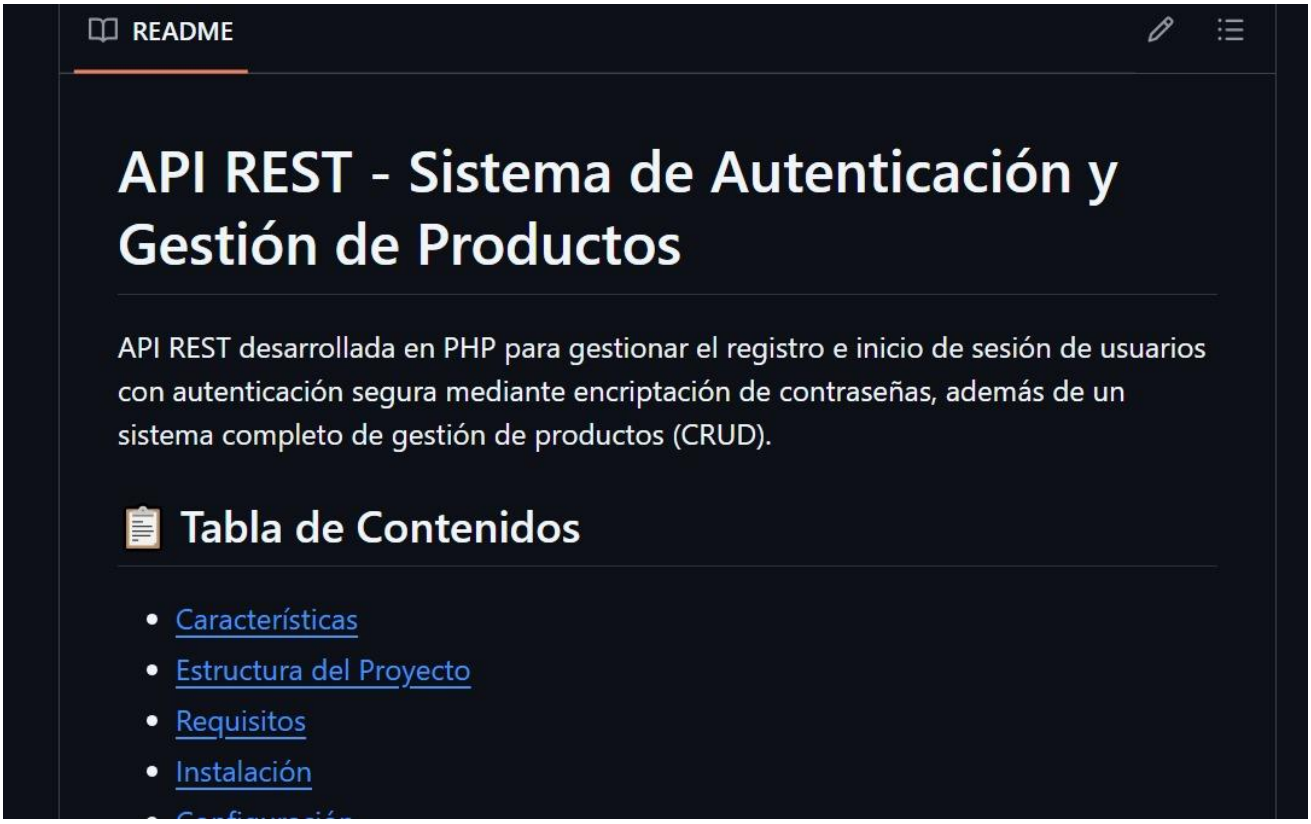
```php
<?php
//inclucion de conexion a base de datos
include_once "../config/db.php";


//definicion de la clase ModelUser
class ModelUser {

    //constructor de la clase
    private $conn;

    public function __construct(){
        $database = new Database();
        $this->conn = $database->conectar();
    }


    //************************************************** */
    //Funcion insertar datos
    //************************************************** */
    public function insertar($array){

        $sql = "INSERT INTO usuarios(
                        usuario,
                        contrasena
                    ) VALUES (
                        :usuario,
                        :contrasena
                        )";
        $stmt = $this->conn->prepare($sql);
        $stmt->execute([
            ':usuario' => $array['usuario'],
            ':contrasena' => password_hash($array['contrasena'], PASSWORD_BCRYPT)
        ]);
    }


    //************************************************** */
    //Funcion para verificar usuario
    //************************************************** */
```



```php
<?php
//************************************************** */
//Configuracion de registro de usuarios
//************************************************** */

//Encabezados obligatorios en las APIS en formatos JSON
header("Content-Type: application/json; charset=UTF-8");
header("Access-Control-Allow-Methods: POST");


//Inclucion del modelo de usuario
include_once "../app/ModelUser.php";

//Inicializacion de instancias
$modelUser = new ModelUser();


//Obtener peticiones JSON
$data = json_decode(file_get_contents("php://input"), true);


//Validacion de datos recibidos
if (empty($data["usuario"]) || empty($data["contrasena"])) {

    //Manejo de errores de validacion
    http_response_code(400);
    echo json_encode(["mensaje" => "Faltan datos requeridos."]);
    exit;

}
```

4. Documentación de los servicios

Se creó la documentación siguiendo el formato



README

API REST - Sistema de Autenticación y Gestión de Productos

API REST desarrollada en PHP para gestionar el registro e inicio de sesión de usuarios con autenticación segura mediante encriptación de contraseñas, además de un sistema completo de gestión de productos (CRUD).

📋 Tabla de Contenidos

- Características
- Estructura del Proyecto
- Requisitos
- Instalación

5. Gestión con herramientas de versionamiento (Git)

Pasos ejecutados:

1. Inicialización del repositorio local
2. git init
3. Creación de commits por cada avance significativo
4. Creación del repositorio remoto (GitHub / GitLab)
5. Enlace del repositorio
6. git remote add origin <URL_DEL_REPOSITORIO>
7. Envío de cambios:
8. git push -u origin main

https://github.com/Andreszxc2001/Apirest-v.2.git

GLOSARIO

API (Application Programming Interface): Conjunto de reglas que permite la comunicación entre sistemas.
REST: Estilo arquitectónico basado en recursos y métodos HTTP.
Endpoint: Dirección específica donde se expone un servicio web.
JSON: Formato de intercambio de datos liviano basado en texto.
HTTP: Protocolo utilizado para la comunicación cliente-servidor.
Token JWT: Mecanismo de autenticación mediante tokens firmados.
Repositorio: Espacio donde se almacena y versiona el código fuente.
Swagger / OpenAPI: Herramienta para documentar API de forma estandarizada.
Controlador: Lógica que ejecuta las operaciones de un endpoint.
Middleware: Funciones que se ejecutan antes de procesar una solicitud.

CONCLUSION

El desarrollo de servicios web es un componente esencial para garantizar la correcta interacción entre los clientes y la lógica del sistema. A través de esta evidencia se diseñaron y construyeron las API requeridas por el proyecto formativo, siguiendo buenas prácticas de arquitectura, seguridad y documentación.
Asimismo, el uso de herramientas de versionamiento permitió estructurar un flujo de trabajo organizado y profesional, facilitando la trazabilidad y evolución del proyecto. La documentación generada asegura que cualquier desarrollador o usuario técnico pueda comprender la funcionalidad de cada servicio y reutilizarlo adecuadamente.

BiBLIOGRAFIA

- Fielding, R. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. University of California.
- Richardson, L., & Amundsen, M. (2013). *RESTful Web APIs*. O'Reilly Media.
- Postman Inc. (2024). *API Development Documentation*.
- OpenAPI Initiative. (2024). *OpenAPI Specification*.
- Mozilla Developer Network (MDN). *HTTP Documentation*.
- Git Documentation. *Git-SCM.com*.