Informe Técnico del Proyecto TodoCamisetas API

1. Descripción General

El proyecto TodoCamisetas es una API RESTful desarrollada en PHP puro, orientada a la gestión de camisetas, clientes y tallas para una tienda de ropa. La arquitectura sigue el patrón MVC (Modelo-Vista-Controlador), separando la lógica de negocio, la gestión de datos y el enrutamiento. Utiliza una base de datos MySQL y está preparada para ser consumida por clientes externos, como aplicaciones web o móviles. El proyecto incluye una colección de Postman para facilitar la prueba de los endpoints.

---

2. Estructura de Archivos y Funcionalidad

2.1. Carpeta `controllers/`

Contiene los controladores que gestionan la lógica de negocio y la interacción entre los modelos y las rutas.

- **CamisetaController.php**
  Gestiona las operaciones CRUD para camisetas.
  Métodos principales:
  - `index()` : Lista todas las camisetas.
  - `show($id)` : Muestra una camiseta específica.
  - `store($data)` : Crea una nueva camiseta.
  - `update($id, $data)` : Actualiza una camiseta existente.
  - `destroy($id)` : Elimina una camiseta.

- **ClienteController.php**
  Gestiona las operaciones CRUD para clientes.
  Métodos principales:
  - `index()` : Lista todos los clientes.
  - `show($id)` : Muestra un cliente específico.

- `store($data)` : Crea un nuevo cliente.
- `update($id, $data)` : Actualiza un cliente existente.
- `destroy($id)` : Elimina un cliente.

- **TallaController.php**
  Gestiona las operaciones CRUD para tallas.
  Métodos principales:

  - `index()` : Lista todas las tallas.
  - `show($id)` : Muestra una talla específica.
  - `store($data)` : Crea una nueva talla.
  - `update($id, $data)` : Actualiza una talla existente.
  - `destroy($id)` : Elimina una talla.

2.2. Carpeta `models/`

Contiene los modelos que representan las entidades de la base de datos y encapsulan la lógica de acceso a datos.

- **Camiseta.php**
  Define la entidad "Camiseta" y sus operaciones sobre la base de datos.
  Métodos:

  - `all()` : Obtiene todas las camisetas.
  - `find($id)` : Busca una camiseta por ID.
  - `create($data)` : Inserta una nueva camiseta.
  - `update($id, $data)` : Actualiza una camiseta.
  - `delete($id)` : Elimina una camiseta.

- **Cliente.php**
  Define la entidad "Cliente" y sus operaciones sobre la base de datos.
  Métodos:

  - `all()` : Obtiene todos los clientes.
  - `find($id)` : Busca un cliente por ID.
  - `create($data)` : Inserta un nuevo cliente.
  - `update($id, $data)` : Actualiza un cliente.
  - `delete($id)` : Elimina un cliente.

- **Talla.php**
  Define la entidad "Talla" y sus operaciones sobre la base de datos.

Métodos:

- `all()` : Obtiene todas las tallas.
- `find($id)` : Busca una talla por ID.
- `create($data)` : Inserta una nueva talla.
- `update($id, $data)` : Actualiza una talla.
- `delete($id)` : Elimina una talla.

2.3. Carpeta `database/`

- **conexion.php**
  Contiene la clase `Database` con el método estático `connect()` , que establece y retorna una conexión PDO a la base de datos MySQL. Centraliza la gestión de la conexión para todos los modelos.

2.4. Carpeta `routes/`

- **routes.php**
  Define el enrutamiento de la API.
  - Analiza la URI y el método HTTP de cada petición.
  - Redirige la petición al método correspondiente del controlador.
  - Gestiona rutas para camisetas, clientes y tallas, soportando los métodos GET, POST, PUT y DELETE.
  - Devuelve un error 404 si la ruta no existe.

2.5. Carpeta `public/`

- **index.php**
  Es el punto de entrada de la aplicación.
  - Incluye el archivo de rutas.
  - Todas las peticiones HTTP pasan por este archivo, que delega el manejo a las rutas definidas.

2.6. Carpeta `sql/`

- **schema.sql**
  Script SQL para la creación de la base de datos y sus tablas:

- `clientes` : Almacena los datos de los clientes.
- `camisetas` : Almacena los datos de las camisetas.
- `tallas` : Almacena los diferentes tamaños disponibles.
- `camiseta_talla` : Tabla intermedia para la relación muchos a muchos entre camisetas y tallas.

2.7. Otros Archivos

- **TodoCamisetas_Postman_Collection.json**
  Colección de Postman que permite probar todos los endpoints de la API.
  - Incluye ejemplos de peticiones GET, POST, PUT y DELETE para camisetas, clientes y tallas.

---

3. Flujo de Funcionamiento

1. El usuario realiza una petición HTTP a un endpoint de la API.
2. El archivo `public/index.php` recibe la petición y carga las rutas.
3. El archivo `routes/routes.php` analiza la URI y el método HTTP, y llama al método correspondiente del controlador.
4. El controlador valida los datos y llama al modelo correspondiente.
5. El modelo interactúa con la base de datos utilizando la clase `Database` .
6. La respuesta se devuelve en formato JSON al cliente.

---

4. Consideraciones Técnicas

- **Separación de responsabilidades:**
  El proyecto separa claramente la lógica de negocio (controladores), el acceso a datos (modelos) y el enrutamiento (routes).
- **Escalabilidad:**
  La estructura modular permite agregar nuevas entidades o funcionalidades de forma sencilla.
- **Pruebas:**
  La colección de Postman facilita la validación manual de todos los endpoints.
- **Seguridad:**
  Actualmente, la API no implementa autenticación ni validación avanzada de datos. Se recomienda agregar estas capas para entornos productivos.