Contexto

Desarrollamos y mantenemos productos para clínicas dentales. Tu tarea es implementar una pequeña API realista para gestionar citas dentales, con un componente de legado/refactorización parecido a lo que encontramos en proyectos reales.

Stack y API

El stack constará de **PHP 8.x** preferiblemente. **Laravel 10** o superior. Base de datos **MariaDB**.

La API REST a implementar deberá tener endpoints para las historias de usuario solicitadas estándar. Los resultados ya sean válidos o erróneos deben devolver un cuerpo JSON estandarizado con un mensaje entendible y respuesta HTTP apropiada.

Historias de usuario

Se te pide construir un backend mínimo para sistema de agenda de clínica dental con las siguientes entidades: **recepcionista** (el usuario de la aplicación), **paciente**, **cita** y **tratamiento**

Como recepcionista se han de permitir las siguientes operaciones:

- **Creación paciente:** Nombre, email, teléfono, nota sobre el paciente
- **Creación dentista:** Nombre, especialidad (p. ej.: ortodoncia, implantes, etc.).
- **Creación de cita:** Selección paciente, dentista, fecha/hora, duración (minutos) y motivo.
- **Listar citas por día:** mostrando información básica
- **Mostrar datos de cita** mostrando información completa
- **[opcional] Asociar uno o varios tratamientos a cita** (p. ej.: "Limpieza", "Radiografía", "Implante") con precio unitario, cantidad y cálculo automático del coste total
- **[opcional] Marcado de citas como completadas o canceladas**

Código legacy

También trabajamos con código legado y con proyectos sin framework. Sobre el siguiente fragmento:

```php
function crearCita($datos)
{
    $db = mysqli_connect('localhost', 'root', '', 'clinica');
    if (!$datos['paciente_id'] || !$datos['dentista_id']) return 'ERROR';
    $inicio = $datos['fecha'] . ' ' . $datos['hora'];
    $fin = date('Y-m-d H:i:s', strtotime($inicio) + ($datos['duracion'] ?: '30') *
60);
    $sql = "INSERT INTO citas (paciente_id, dentista_id, inicio, fin, motivo) VALUES
(" .
        $datos['paciente_id'] . "," . $datos['dentista_id'] . ",'" . $inicio . "','" .
$fin . "','" . $datos['motivo'] . "')";
    mysqli_query($db, $sql);
    return mysqli_insert_id($db);
}
```

Tarea 1 - Crea una **alternativa moderna y orientada a objetos** para esta función en php puro con tipos claros adaptando el código de laravel para que las citas funcionen con la nueva implementación

[opcional] Tarea 2 - Crea un nuevo componente en **PHP puro** para revisar si hay solapamientos entre citas a la hora de agendar.

Arquitectura, código, robustez y seguridad

Nos interesa evaluar cómo estructuras un backend real, buscando código mantenible y escalable, no solo funcional:

- **Separación de responsabilidades:** Uso correcto de capas (Controladores, Servicios/Casos de uso, Repositorios, DTOs). ¿El código es desacoplado?
- **Principios:** Aplicación de **SOLID** y otros principios de **código limpio** así como **PSR-12**
- **Uso del Framework:** Uso de herramientas nativas del framework (Service Container, Eloquent, Migraciones, Seeders)
- **Entradas:** Validación estricta (Form Requests/Validators) y saneamiento de datos.
- **Manejo de Errores:** Gestión de excepciones, try-catch bloques lógicos y respuestas HTTP coherentes (no exponer stack traces en entornos no locales).
- **Diseño de Esquema:** Correcto uso de migraciones, claves foráneas/primarias, tipos de datos y normalización.
- **Eficiencia:** Prevención del problema N+1 (Eager Loading), uso adecuado de índices y optimización de consultas.
- **Documentación:** en código y proceso
- **Git y Versionado:** Histórico de commits limpio, atómico y con mensajes descriptivos en el repositorio local.

Puesta en marcha y entrega

Proporciona una configuración basada en **Docker** para ejecutar el proyecto. Deberá contener un docker-compose.yml con los servicios **(1) PHP-FPM** + servidor web (Nginx o similar), **(2)** MariaDB, **(3) [opcional]**: Redis (si usas caché/colas). También incluye el Dockerfile para la aplicación PHP.

Para la puesta en marcha añade un **README** explicando **(1)** cómo iniciar el proyecto, **(2)** cómo ejecutar las migraciones y seeders, **(3)** ejemplos de llamadas a la API o colección postman.

La entrega consta de una de estas dos opciones:

A) Directorio comprimido. **En este caso el directorio contendrá carpeta .git local**
B) Enlace a repositorio git **privado** (GitLab, GitHub, otro)

Criterios de validación del entregable

Queremos ver cómo trabajas y el proceso que sigues para implementar lo solicitado. Te pedimos que dediques un **tiempo razonable (7-8 horas) para el desarrollo**. **Es aceptable no terminar todo**, aunque si dejas partes sin hacer **explica tus decisiones y prioridades en el README**.

Explicación de lo implementado, trade-offs tomados y siguientes pasos

Puntos que valoraremos:

- Funcionamiento de historias de usuario principales.
- Limpieza de código, legibilidad, estructura y arquitectura
- Diseño de la base de datos y las consultas
- Manejo del código legacy y componentes en PHP puro
- Documentación del proyecto e histórico de git
- Tests en caso de que se entreguen