Diseño de Arquitectura del Sistema Web - NutriGym

**Fecha:**

**Autor:** **Benjamin Miranda Quispe**

**Brandon Jair Chipana Salazar**

**John Manuel Villagarcia Mendoza**

**Victor Fernando Masias Baca**

**Versión:** 1.0

# 1. Introducción

El presente documento, explica con detalle todas las tecnologías que se usaran para el desarrollo del presente proyecto, además, explicaremos como las utilizamos y de que manera resultan beneficiosas para que el proyecto tenga éxito. La inteligencia artificial juega un papel fundamental en el sistema, ya que permitirá que los usuarios puedan acceder a información que antes se accedía con dificultad. Por ende creemos que las tecnologías que usaremos en este proyecto son las adecuadas para construir el sistema.

# 2. Visón General de la Arquitectura

## 2.1. Estilo Arquitectónico

**1. Arquitectura en Capas**

* **Presentación (Frontend)**: Blade + Tailwind + Alpine.js: Se encargaran de la interfaz del chatbot.
* **Lógica de Aplicación (Backend Laravel)** : controladores, servicios y casos de uso que gestionan las conversaciones.
* **Persistencia (Base de Datos)** : MySQL para almacenar usuarios, historiales de chat, dietas generadas, etc.
* **Integración (API / Microservicios)** : en caso de que el chatbot necesite consumir un motor de IA externo (por ejemplo, OpenAI API).

## 2.2. Diagrama General de Arquitectura:

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

# 3. Descripción de Componentes Principales

## 3.1. Frontend (Presentación)

- Tailwind CSS\*\* - Framework CSS utility-first

- Alpine.js\*\* - Framework JavaScript para interactividad

- Blade Templates\*\* - Motor de plantillas de Laravel

- Responsabilidades:

* Plantillas intuitivas para el usuario
* Mensajes de error por si algún campo no es bien rellenado
* Interactividad con el chat bot de una manera fluida
* Colores amigables para el usuario
* Implementar componentes reutilizables (formularios de registro, menús, paneles de control).
* Generar páginas dinámicas a partir de datos enviados por Laravel.

## 3.2. Backend (Lógica de negocio)

- Laravel 10 - Framework PHP

- MySQL 8.0 - Base de datos relacional -> phpMyAdmin

- Responsabilidades:

* Registrar, autenticar y autorizar usuarios (deportistas, entrenadores, administradores).
* Validar datos antes de guardarlos (ejemplo: email único, contraseñas seguras) .
* Recibir mensajes desde el frontend y procesarlos.
* Consultar historial de conversaciones en la base de datos.
* Generar dietas personalizadas según objetivos del usuario (ganar masa, perder peso, mantener).
* Responder en formato json para que el frontend (Alpine.js) lo consuma.

### 3.2.1. Módulos del Backend

### ****- Módulo de Autenticación y Seguridad****

* Registro, login y logout de usuarios.
* Roles y permisos (deportista, entrenador, administrador).
* Recuperación y cambio de contraseñas.
* Protección CSRF, XSS y control de sesiones.

- **Módulo de Gestión de Usuarios**

* Perfil de usuario (datos personales, actividad física, objetivos).
* Actualización de información.
* Historial de progreso y preferencias.

- **Módulo de Chatbot**

* Recepción de mensajes desde el frontend.
* Procesamiento de conversaciones:
  + Reglas predefinidas.
  + Conexión a motor de IA externo (ej. OpenAI API).
* Gestión del historial de chat.
* Orquestación de respuestas y envío al frontend.
* **Módulo de Planes Nutricionales**
* Generación de dietas personalizadas (según objetivos).
* Cálculo de calorías y macronutrientes.
* Administración de alimentos y recetas.
* Registro de planes asignados a cada usuario.

## 3.3. Base de Datos (Persistencia)

MySQL 8.0 - Base de datos relacional -> phpMyAdmin

# 4. Integraciones Externas (Opcionales)

* No tenemos integraciones externas para el proyecto.

# 5. Seguridad

* Uso de HTTPS.

# 6. Escalabilidad y Despliegue

Posibles problemas que pueden surgir durante el desarrollo.

* **Base de Datos**

Optimizar índices y consultas SQL.

Implementar **replicación de MySQL** (master-slave) para balancear lectura/escritura.

Uso de **caché de consultas** (Redis o Memcached).

Particionamiento de tablas para grandes volúmenes de datos.

* **Aplicación (Laravel)**

Activar **caché de configuración, rutas y vistas** (php artisan config:cache).

Uso de **queues y jobs** con Redis/RabbitMQ para procesos pesados (ej: generación de dietas).

Escalamiento horizontal: múltiples instancias de Laravel detrás de un **balanceador de carga** (Nginx/HAProxy).

* **Frontend**

Minificación y optimización de assets con **Laravel Mix / Vite**.

Uso de **CDN** para servir imágenes y librerías estáticas.

# 7. Conclusiones

El diseño del sistema NutriGym permite una escalabilidad progresiva desde un único servidor hasta un sistema distribuido con balanceadores de carga, contenedores y orquestación; a la vez que la incorporación de cachés, colas de procesos y optimización de consultas garantiza buen rendimiento en las operaciones críticas, mientras que un esquema de despliegue seguro y automatizado a través de CI/CD, entornos diferenciados y buenas prácticas de seguridad asegura fiabilidad, velocidad y resistencia en el despliegue del sistema en producción.