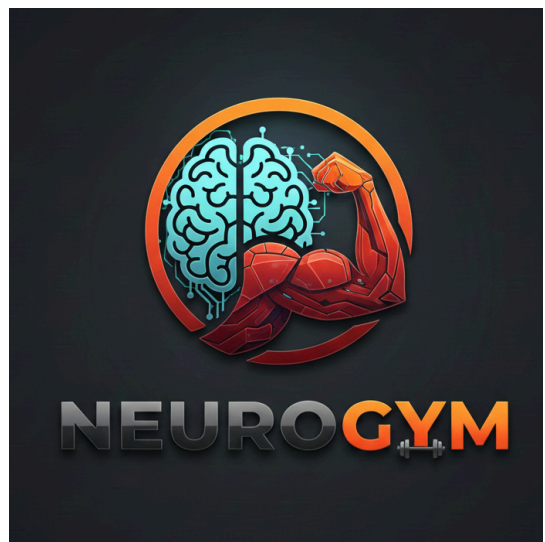


# NEUROGYM

Manual Tecnico



# Índice

## **1. Introducción**

## **2. Análisis del Problema**

### **2.1 Problemática**

### **2.2 Clientes Potenciales**

### **2.3 Análisis DAFO**

### **2.4 Monetización y Beneficios**

## **3. Diseño de la Solución**

### **3.1 Tecnologías Elegidas**

### **3.2 Arquitectura del Sistema**

### **3.3 Diagrama de Clases**

### **3.4 Persistencia de Datos**

### **3.5 Consideraciones Técnicas**

## **4. Enlaces de Interés**

# 1. Introducción

**NeuroGym** es una aplicación de entrenamiento personal que fue ideada como una solución gratuita y accesible para la planificación, seguimiento y optimización del progreso en los entrenamientos en el gimnasio dirigida a tanto usuarios principiantes como a atletas avanzados.

Esta aplicación esta desarrollada completamente con **Flutter** y se fundamenta en tres pilares funcionales principales:

- **Registro de Sesiones de entrenamiento:** Proporcionando un sistema de registro en tiempo real de ejercicios, series, repeticiones y pesos.
- **Gestión de Rutinas:** Que permite la creación y edición de programas de entrenamiento totalmente personalizados.
- **Optimización de rutinas por Inteligencia Artificial(Groq):** Con e objetivo de maximizar el progreso y asegurar una mejora continua.

En resumen, **NeuroGym** no es solo una aplicación de gestión de rutina accesible, **NeuroGym** es una hoja de ruta personalizada e accesible para cualquiera pueda potenciar sus entrenamientos.

## 2. Análisis del Problema

### 2.1 Problemática

Se detecto que la mayoría de las aplicaciones de gestión de rutinas, estaban limitadas con suscripciones muy costosas para lo que te aporta realmente la aplicación.

### 2.2 Clientes Potenciales

- **Entusiastas del fitness:** Que buscan maximizar su rendimiento y necesitan una herramienta de seguimiento detallada y precisa.
- **Usuarios Principiantes/Intermedios:** Que necesitan guía y planes claros para iniciarse en el mundo fitness.
- **Deportistas orientados a datos:** Que necesitan tener los datos de sus entrenamientos pasados para programar los siguientes.

## 2.3 Análisis DAFO



## 2.4 Monetización y Beneficios

En un principio este proyecto está realizado por el bien de la comunidad fitness y no espero monetizarlo de forma directa, la monetización irá a través de anuncios poco invasivos y donaciones.

## 3. Diseño de la Solución

### 3.1 Tecnologías Elegidas

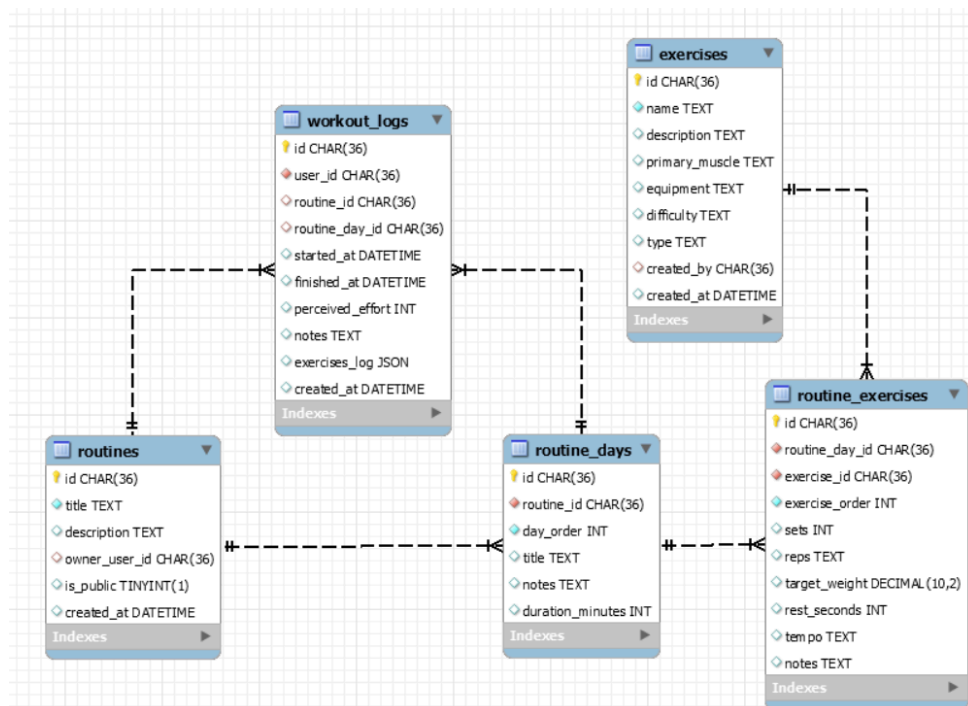
- FrameworkFlutter (Dart)
- BackendSupabase (PostgreSQL)
- IAGroq API (LLaMA 3.1)

## 3.2 Arquitectura del Sistema

El sistema opera bajo una arquitectura de Cliente (**Flutter**) - BBDD (**Supabase**) - Servicios Externos (**Groq**), asegurando escalabilidad y separación de responsabilidades.

- **Capa Cliente (Flutter):** Gestiona la UI y el estado local (StatefulWidget). Contiene la lógica para construir los *prompts* de la IA y parsear el JSON de vuelta.
- **Capa BaaS (Supabase):** Aloja la base de datos PostgreSQL y ejecuta la lógica compleja de cálculo de fuerza a través de Procedimientos Almacenados (RPC).
- **Capa IA (Groq):** Se utiliza estrictamente como un motor de inferencia para generar contenido estructurado (rutinas JSON).

## 3.3 Diagrama de Clases



## 3.4 Persistencia de Datos

La persistencia de datos en **NeuroGym** se gestiona en Remoto (**Supabase**).

- Base de Datos: PostgreSQL 15(**Supabase**).
  - Almacena los datos de cada usuario, como credenciales y los datos de entrenamiento y se sincroniza con la interfaz mediante de las operaciones (**INSERT**, **UPDATE**, **DELETE**) que se maneja con la comunicación por HTTPS/WebSockets.

## 3.5 Consideraciones Técnicas

- **Rendimiento de Flutter**
  - **Manejo asíncrono:** Con el uso de **Future<T>** y **await** para todas las interacciones de Red, base de datos e IA.
- **Estructura del Backend con SupaBase**
  - **Optimización de Queries:** El uso de procedimientos para centralizar la lógica de calculo pesada aprovechando la eficiencia de **PostgreSQL**.
  - **Seguridad RLS:** Las políticas de **Row Level Security** donde un usuario solo puede modificar datos de tablas específicas.
- **Integración de IA:** Que después muchas pruebas con modelos de IA de la pagina **Hugging Faces**, donde dio diversos problemas para implementarla con la interfaz, optamos por cambiar de modelo a **Groq**.
  - **Robustez de Parsing:** La respuesta de **Groq** es un **JSON** parseado para insertarlo en la base de datos.

## 4. Enlaces de Interés

[Flutter Widgets Catalogue](#)

[Documentación de Supabase](#)

[Groq API Cookbook](#)

[PostgreSQL JSONB Type](#)