

# NUTRI wizard

George Albadr

Milton Beltrán

# Agenda

1. Introducción del proyecto
2. Retos Encontrados
3. Benchmark - Datos de Lectura
4. CRUDS
5. Domain Driven Design

# ¿NutriWizard?

NutriWizard es una aplicación web diseñada para gestionar información nutricional y de salud de los usuarios, permitiendo un seguimiento detallado de sus hábitos y objetivos alimenticios. Su propósito es ayudar a los usuarios a alcanzar metas de salud personalizadas mediante recomendaciones basadas en sus características físicas y necesidades específicas.

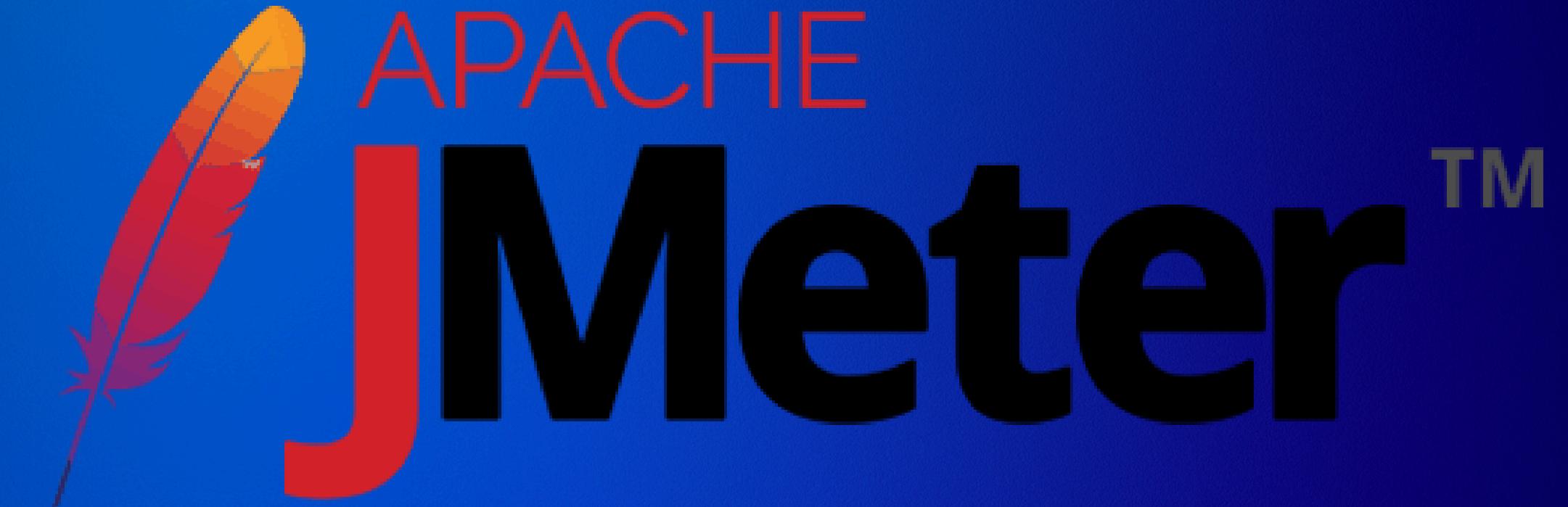


# Datos de Lectura

Average user Response Time: 8ms

Max user calls capacity: 3,429

Max user post calls: 2,002





## Desafíos

1. Entornos de desarrollo no iguales
  - a. Solución: Dockerización
2. UnSync con tablas debido a caché
  - a. Solución: RefresCache
3. Problema de conexión duplicada
  - a. Solución: Centralizar la config



## Oportunidades

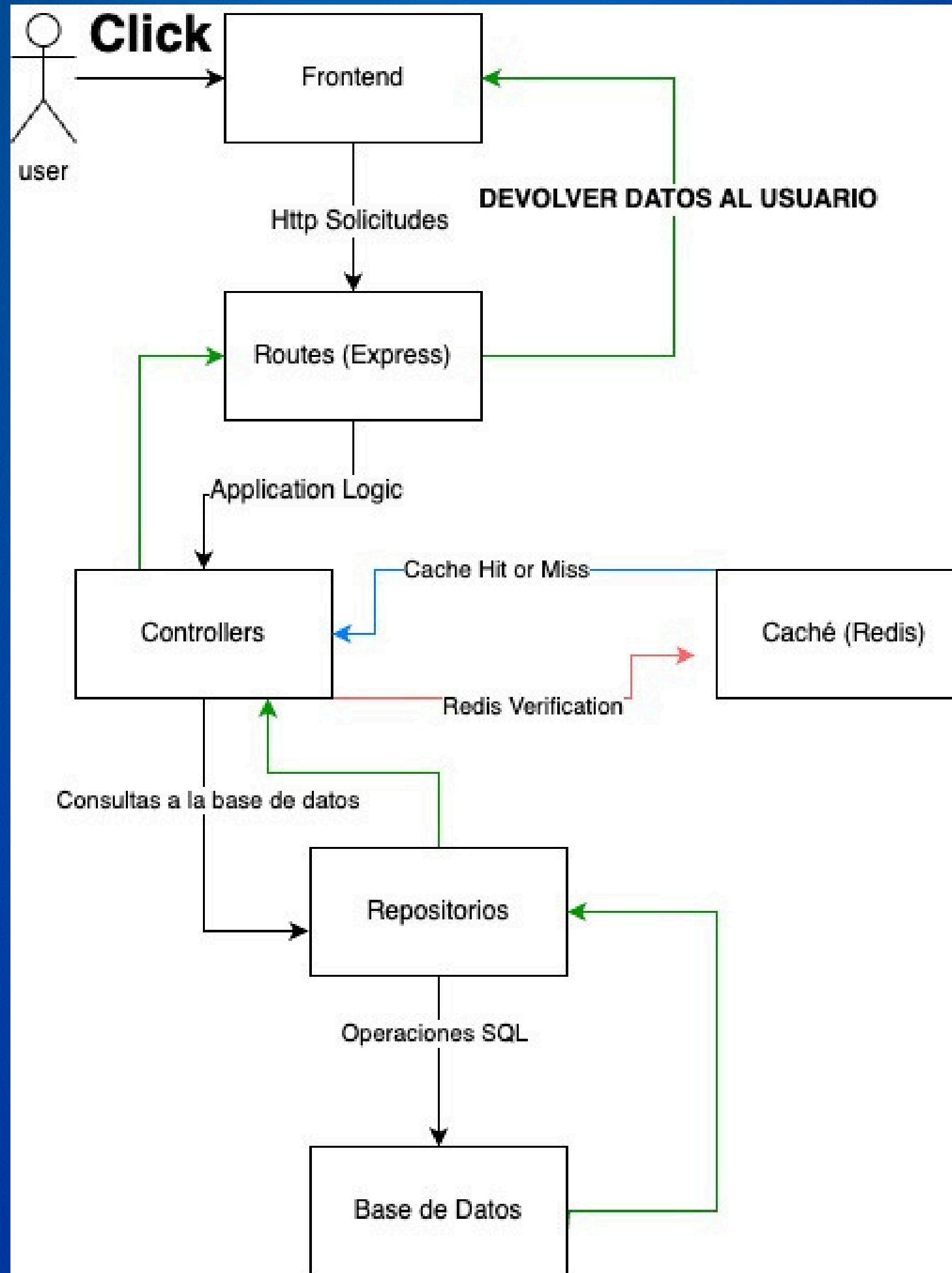
### Áreas de Mejora para el Proyecto:

1. **Testing Automatizado:**
  - Implementar pruebas unitarias y de integración para garantizar la calidad del código, especialmente en los controladores y repositorios.
2. **Documentación del Código:**
  - Mejorar los comentarios en el código y generar documentación más detallada para facilitar la colaboración y la incorporación de nuevos desarrolladores.
3. **Optimización del Caching:**
  - Revisar y mejorar la lógica de invalidación y actualización del cache para evitar inconsistencias y mantener la eficiencia del sistema.

# Domain Driven Design

- Capas Definidas:
  - **Modelos:** Representan el núcleo del dominio, como Usuario y Meals. Contienen la estructura y lógica relacionadas con las entidades principales.
  - **Controladores:** Gestionan la lógica de aplicación y procesan las solicitudes HTTP, actuando como intermediarios entre el frontend y la lógica del negocio.
  - **Repositorios:** Encapsulan las interacciones con la base de datos, garantizando una separación clara entre la lógica del dominio y las operaciones CRUD.
  - **Infraestructura:** Incluye herramientas externas como Sequelize para ORM y Redis para mejorar el rendimiento mediante caching.
- Separación de Responsabilidades:
  - Cada capa tiene responsabilidades claras, minimizando dependencias cruzadas y mejorando la mantenibilidad.
- Enfoque Modular:
  - Permite escalar el proyecto fácilmente, con una estructura que facilita la adición de nuevas funcionalidades sin afectar las existentes.





# CRUDS

## Usuarios

- Create: Registro de nuevos usuarios.
- Read: Obtención del perfil del usuario y listado de usuarios.
- Update: Actualización de la información personal y objetivos nutricionales.
- Delete: Eliminación de usuarios.

## Meals

- Create: Registro de nuevas comidas en la base de datos.
- Read: Obtención de todas las comidas y consulta por ID.
- Update: Edición de atributos de una comida existente.
- Delete: Eliminación de comidas por ID.

## Menu

- Create: Agregar comidas al menú del usuario.
- Read: Obtener el menú del usuario con comidas asignadas.
- Update: Modificar las comidas dentro de un menú.
- Delete: Eliminar comidas específicas del menú.



# ¿Dudas o comentarios?

Dedícale un momento a responder cualquier duda o comentario. Es importante terminar con un espacio para interactuar con tu audiencia.

