**Documentación Técnica del Microservicio: PlanesAlimentacion**

**Introducción**

El microservicio PlanesAlimentacion ha sido desarrollado en .NET 8 con PostgreSQL como motor de base de datos, siguiendo los principios de Domain-Driven Design (DDD), Clean Architecture y el patrón CQRS (Command Query Responsibility Segregation). Su propósito es gestionar planes alimentarios personalizados, permitiendo registrar, consultar y estructurar tiempos de comida con recetas asignadas.

**Funcionalidades Implementadas**

Registrar un nuevo plan de alimentación: Se permite definir la duración del plan (en días), los tiempos de comida (ej. desayuno, almuerzo) y las recetas asociadas a cada tiempo.

Asignar recetas a los tiempos de comida: Cada tiempo de comida puede contener múltiples recetas, lo que permite flexibilidad y personalización.

Consultar los planes registrados: Se expone un endpoint para recuperar todos los planes con sus tiempos y recetas, estructurados en formato DTO.

**Aplicación de Domain-Driven Design (DDD)**

La capa de dominio define las entidades centrales del negocio:

PlanAlimentacion: Agregado raíz que representa un plan completo.

TiempoComida: Subentidad que agrupa recetas por momento del día.

Receta: Entidad simple que representa una preparación alimenticia.

Se utiliza el patrón Repository para abstraer el acceso a datos (IPlanRepository), permitiendo persistencia sin acoplar el dominio a la infraestructura.

**Arquitectura Limpia**

El proyecto está dividido en capas bien definidas:

Domain: Contiene las entidades y contratos del negocio.

Application: Implementa los casos de uso mediante comandos y consultas.

Infrastructure: Provee la persistencia con Entity Framework Core y PostgreSQL.

WebApi: Expone los endpoints HTTP y configura los servicios.

Cada capa depende únicamente de las capas interiores, respetando el principio de inversión de dependencias.

**Patrón CQRS con MediatR**

Se implementa CQRS mediante la librería MediatR, separando claramente:

Comandos (CreatePlanCommand) para modificar el estado del sistema.

Consultas (GetPlanesQuery) para recuperar información sin efectos secundarios.

Cada comando y consulta tiene su propio handler, evitando saturación de clases y manteniendo la lógica aislada.

**Diagrama de Clases del Dominio**

El dominio se compone de tres entidades principales y una interfaz de repositorio. El diagrama refleja relaciones de agregación entre PlanAlimentacion, TiempoComida y Receta, y la dependencia del repositorio.

*(Aquí puedes insertar el diagrama generado como imagen)*

**Flujo de Datos**

El cliente envía una solicitud HTTP al controlador (PlanesController).

El controlador delega a MediatR, que invoca el handler correspondiente.

El handler construye las entidades del dominio y llama al repositorio.

El repositorio persiste los datos usando PlanesDbContext.

En consultas, el repositorio recupera los datos y el handler los transforma en DTOs.

**Tecnologías Utilizadas**

.NET 8.0

PostgreSQL

Entity Framework Core 9.0

MediatR 13.0

Swagger (Swashbuckle.AspNetCore)

**Buenas Prácticas Aplicadas**

Inyección de dependencias con AddScoped y AddDbContext.

Encapsulamiento de lógica en entidades.

Separación de responsabilidades en capas y clases.

Uso de DTOs para evitar exposición directa del modelo de dominio.

Validación estructural mediante tipos fuertes y constructores.

**Conclusión**

El microservicio PlanesAlimentacion cumple con los principios de diseño moderno, es extensible, mantenible y preparado para escalar. La separación de responsabilidades, el uso de patrones como CQRS y DDD, y la aplicación de Clean Architecture demuestran un dominio sólido de las metodologías requeridas.