Sprint 4

Modelado del Proyecto (SDLC)

Objetivo del sprint

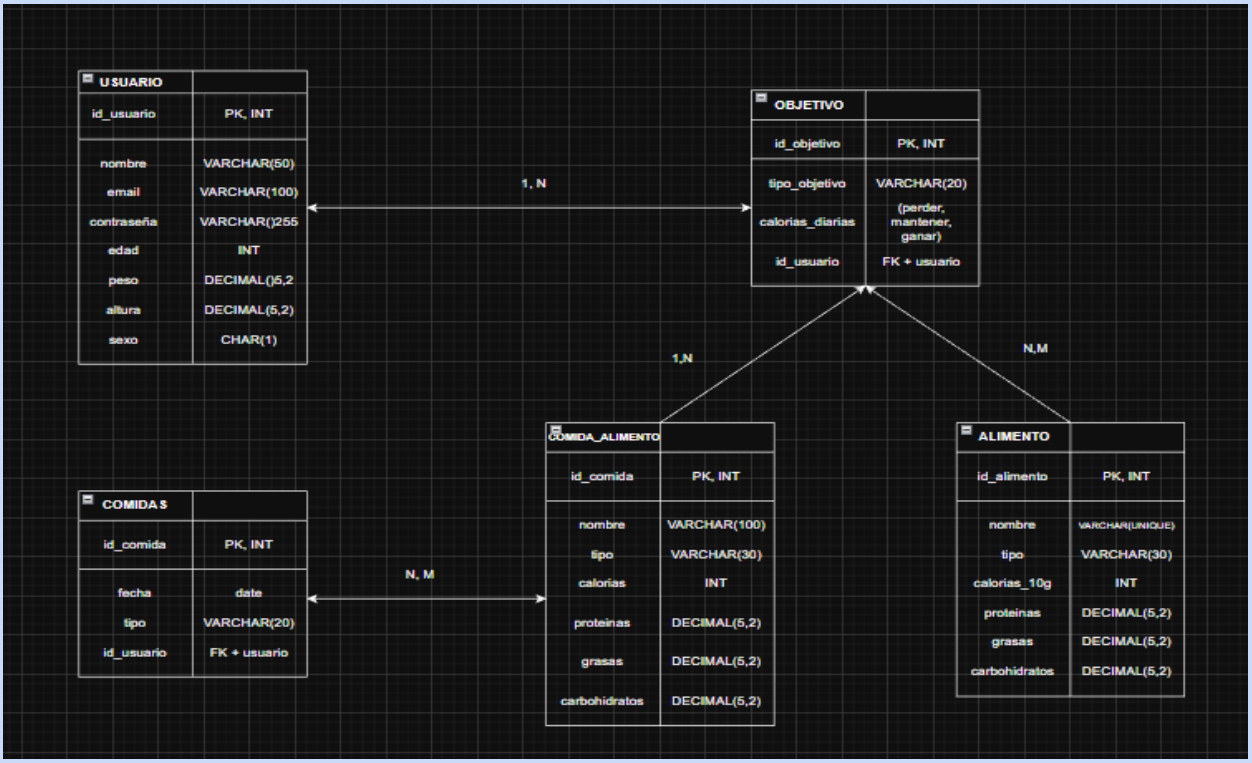
- **Definir el modelo de dominio y de datos mediante UML/DER.**
- **Diseñar los servicios y contratos API (OpenAPI), incluyendo seguridad, errores, paginación y versión.**
- **Descomponer HUs a tareas técnicas estimadas y planificadas en Jira Cloud, con DoR/DoD.**

Actividades clave

1) Modelado de dominio y base de datos (UML / DER)

- Identificar entidades y relaciones desde épicas/HUs.
- Definir atributos, tipos, PK, FK, unicidad, checks, cardinalidades y cascadas.

IMAGEN DEL DIAGRAMA



fuente editable:
https://app.diagrams.net/?mode=google#G1clybknlDz-2GqPi_Eb8G5C4m8jE52-Lr#%7B%22pageId%22%3A%22HMswIBA9uO_Mr5R0tA-5%22%7D

2) Diseño de servicios y API (enfoque API-first)

- Derivar recursos y operaciones desde HUs; separar comandos (creación/cambios) y consultas (lectura/filtrado) si aplica.
- Definir OpenAPI (openapi.yaml): paths, métodos, request/response, DTO/JSON Schema, códigos de estado, errores y paginación.
- Establecer seguridad (p. ej., bearer/JWT), roles/scopes, versionado (/v1), y naming consistente (kebab-case en paths, snake_case en DB).
- Acordar guía de errores (formato { code, message, details[] }) y convenciones (orden de campos, envoltorios, metadatos total/page/limit).
- Proponer pruebas de contrato (Postman), y casos mínimos por HU crítica.

ruta del proyecto:
C:\Proyectos\Repos\VMRC-PI-BACK\t1\RidaoChavesVictorManuel-PI-SPRINT4

Enfoque general

- La API se ha diseñado siguiendo un enfoque **API-first**.
- Los recursos y operaciones se derivan directamente de las HUs del proyecto (calculadora de calorías).
- Se separan **comandos** (creación, actualización, eliminación) de **consultas** (lectura, filtrado y listados).

Recursos principales

- **Users**: CRUD de usuarios (**POST, GET, PUT, DELETE**).
- **Foods**: CRUD de alimentos (**POST, GET, PUT, DELETE**).
- **Exercises**: CRUD de ejercicios (**POST, GET, PUT, DELETE**).
- **Consumptions**: registrar y listar consumos de alimentos (**POST, GET, PUT, DELETE**).
- **Activities**: registrar y listar actividades físicas (**POST, GET, PUT, DELETE**).
- **Goals**: gestión de objetivos nutricionales (**POST, GET, PUT**).

OpenAPI

Especificación completa disponible en: **docs/api/openapi.yaml**.
Contiene todos los paths, métodos, request/response en JSON, códigos de estado, paginación y seguridad JWT.

Guía de errores

- Formato uniforme: **{ code, message, details[] }**.
- Principales códigos:
  - **VALIDATION_ERROR** → Error de validación de datos
  - **NOT_FOUND** → Recurso no encontrado
  - **CONFLICT** → Conflicto, ej. email ya registrado
  - **UNAUTHORIZED** → Token inválido o no proporcionado
  - **FORBIDDEN** → Usuario sin permisos
  - **SERVER_ERROR** → Error interno del servidor

Guía completa en: docs/api/errores.md.

Convenciones

- Versionado: todos los endpoints comienzan con **/v1/**.
- Naming: paths en **kebab-case**, base de datos en **snake_case**.
- Seguridad: JWT Bearer, roles **admin/user**.
- Listados: paginación con **total, page, limit**.
- Orden de campos en JSON: identificadores → datos principales → metadatos.
- Convenciones completas: **docs/api/convenciones.md**

3) Descomposición técnica de HUs → Tareas (con Jira Cloud)

Enlace al proyecto de Jira:
https://adaits-team.atlassian.net/jira/software/projects/GDCS2AYD/boards/2/timeline

Se han realizado las subtareas de las **HUs** en la sección de **EP1- Gestión de usuarios**.