

IES Doctor Balmis

Aplicación web gestora de comidas Smart DieT

Estibaliz Etxaburu Lekube

Tutor Vicente Martínez Martínez

23/05/2025



Abstract	4
Introducción	5
Necesidades empresariales para el desarrollo	6
Recursos humanos	6
Prevención de riesgos laborales	6
Iniciativa emprendedora	6
Requisitos funcionales de la aplicación	7
Explicación detallada del problema y funcionalidad de la aplicación	7
Target y tipo de usuario	7
Análisis y diseño	8
Esquema general del sistema:	8
Diagrama de casos de usos.	13
Diagrama de clases	14
Diagramas de arquitectura de persistencia	15
Diagrama Entidad relación	22
Diagrama de Controlador Spring MVC	24
Diagrama de flujo de autenticación con JWT	25
Flujo de autenticación	25
Diagramas de flujo de cada vista	27
Login	28
Registro	28
Productos	29
Productos de API Open food facts	30
Perfil	32
Editar perfil	33
Despensa	34
Consumos	36
Codificación	37
Entorno de Desarrollo y Tecnologías del Proyecto	37
Aspectos relevantes de la implementación	38
Manual de usuario.	39
Manual de SmartDieT	39
1. Página de Inicio	39
2. Inicio de sesión	41
3. Registro de Usuario	42
4. Perfil de Usuario	43
5. Editar Perfil	44
6. Productos (Local)	44
7. Productos (API Open Food Facts)	46
8. Despensa Personal	47

9. Registro de Consumos	48
Requisitos e instalación	49
Entorno de Desarrollo	49
Configuración de Base de Datos	49
Variables de Entorno y Seguridad	49
Ejecución del Proyecto	49
Pruebas	49
Resumen de Tecnologías Clave	50
Conclusiones	51
Bibliografía	52

Abstract

Our project is a web application designed to provide users with a comprehensive nutritional product consultation service, combining information from both an internal database and an external API. Aimed at individuals interested in managing their dietary habits, the application not only enables users to search and explore nutritional products, but also allows them to add selected items for personal consumption and pantry logs for easy tracking.

Each user has access to a personalized interface where they can manage and visualise their pantry and daily consumption history. The application is built using a hexagonal architecture; which improves maintainability, scalability, and simplifies future adaptations or enhancements. Security and data privacy are managed through the implementation of Spring Security in conjunction with JWT (JSON Web Tokens), ensuring that user sessions and data are protected efficiently.

Looking ahead, the application will introduce different user roles, including a higher-level user type with access to feedback on consumption patterns based on their activity level and dietary goals. Additionally, users will be able to register commonly used recipes to streamline tracking and support consistent eating habits. An Android mobile version is also planned to improve accessibility, although the platform is designed for brief, intentional use at specific times of the day—such as in the morning or evening— rather than continuous engagement.

Introducción

El proyecto tiene como objetivo desarrollar una aplicación web que facilite la consulta y gestión de información nutricional de productos, combinando una base de datos interna con una API externa. Busca ofrecer a los usuarios una herramienta intuitiva para registrar y monitorear su consumo diario y despensa personal, promoviendo hábitos alimenticios más conscientes. La aplicación, basada en una arquitectura hexagonal, prioriza la escalabilidad, mantenibilidad y seguridad mediante Spring Security y JWT. En el futuro, se planea incorporar roles de usuario avanzados, análisis de patrones de consumo y una versión móvil para mayor accesibilidad, enfocándose en un uso puntual y eficiente.

Este proyecto surge de la creciente necesidad de promover una alimentación más consciente, brindando a los usuarios una herramienta centralizada para consultar y gestionar su consumo nutricional. Se eligió debido a la falta de aplicaciones que combinen bases de datos propias con APIs externas, ofreciendo información más completa y personalizable. Satisface la demanda de quienes buscan controlar su dieta de manera organizada, segura y adaptada a sus metas. Además, su arquitectura hexagonal y enfoque en seguridad garantizan un sistema robusto y escalable para futuras mejoras.

Nivel de innovación: Este proyecto destaca por integrar una base de datos propia con una API externa en un único servicio, combinado con un sistema de gestión de despensa personalizado y análisis de hábitos alimenticios, lo que lo diferencia de soluciones más genéricas.

Aplicaciones similares en el mercado: Plataformas como MyFitnessPal, Yazio o FatSecret ofrecen seguimiento nutricional, pero ninguna combina consulta de productos externos con un sistema interno de despensa y futuros análisis basados en metas dietéticas personalizadas.

Diferenciación clave: A diferencia de las apps existentes, este proyecto prioriza la arquitectura modular, la seguridad con JWT y una experiencia de uso breve e intencional, en lugar de un enfoque en engagement constante.

Necesidades empresariales para el desarrollo

Recursos humanos

El proyecto requerirá desarrolladores full-stack, un diseñador UX/UI y un administrador de bases de datos, además de los promotores.

Se formalizarán contratos laborales según la carga de trabajo, priorizando contratos indefinidos o temporales según las necesidades del desarrollo.

Se cumplirá con todas las obligaciones de cotización a la Seguridad Social, incluyendo altas de empleados y pagos de cuotas correspondientes.

Prevención de riesgos laborales

Los principales riesgos laborales están asociados al trabajo prolongado frente a pantallas, posibles problemas ergonómicos y estrés por plazos ajustados.

Como medidas preventivas, se implementarán pausas activas, ajustes ergonómicos en los puestos de trabajo y formación en gestión del tiempo para evitar sobrecarga.

Iniciativa emprendedora

La forma jurídica elegida será una Sociedad Limitada (S.L.) por su flexibilidad y limitación de responsabilidad frente a deudas.

Los trámites administrativos incluirán la inscripción en el Registro Mercantil, obtención del NIF y licencias pertinentes, así como la legalización de la actividad ante Hacienda y la Seguridad Social.

Requisitos funcionales de la aplicación

Explicación detallada del problema y funcionalidad de la aplicación

La aplicación busca resolver la falta de herramientas integrales para gestionar la nutrición personal, combinando datos de una base de datos propia con una API externa para ofrecer información detallada sobre productos alimenticios. Permite a los usuarios registrar y organizar su despensa, monitorear su consumo diario y recibir recomendaciones nutricionales basadas en sus metas. La arquitectura hexagonal empleada (evidenciada por la separación clara entre controladores, servicios y repositorios) garantiza modularidad y facilidad para integrar nuevas funcionalidades, como el futuro análisis de patrones de consumo o la expansión a móvil.

Target y tipo de usuario

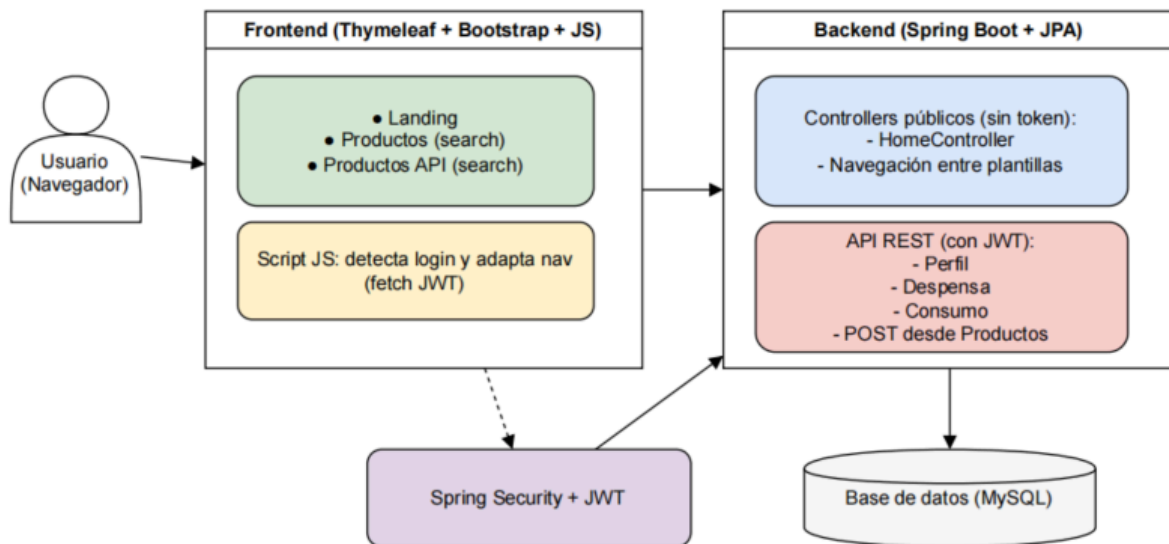
El sistema está dirigido a:

- **Personas interesadas en mejorar sus hábitos alimenticios**, desde quienes buscan mantener una dieta equilibrada hasta aquellos con objetivos específicos (ej. pérdida de peso o control de alergias).
- **Usuarios con nivel básico-medio de experiencia tecnológica**, ya que la interfaz prioriza la simplicidad y usabilidad en momentos puntuales (ej. registro de comidas). No requiere conocimientos avanzados, pero sí motivación para llevar un seguimiento constante.
- **Futuros roles avanzados, como nutricionistas o entrenadores**, que podrían acceder a funciones de análisis y recomendaciones personalizadas en próximas iteraciones.

La aplicación se diferencia de soluciones existentes al integrar gestión de despensa, seguimiento nutricional y recomendaciones en un único sistema seguro (con JWT) y adaptable, diseñado para un uso breve pero efectivo.

Análisis y diseño

Esquema general del sistema:



Mapa de navegación de rutas públicas y privadas:

Vista	Ruta	Método en Controller	Protegido JWT
landing	/	mostrarInicio()	✗ No
registro	/registro	mostrarFormularioRegistro()	✗ No
login	/login	loginPage()	✗ No
productos	/productos	verProductos()	✗ No (pero con partes JS privadas)
productos2 (API)	/productosapi	verProductosApi()	✗ No (con partes privadas vía JS)
perfil	/perfil	mostrarPerfil()	✓ Sí (solo el fetch de datos)
editar-perfil	/editar-perfil	editarPerfil()	✓ Sí (requiere datos de usuario)
despensa	/despensa	verDespensa()	✓ Sí (fetch de datos)
consumo	/consumo	verConsumo()	✓ Sí (fetch de datos)

El siguiente esquema representa la arquitectura del backend del sistema organizada en capas. Controladores (rosa), servicios (azul), repositorios de dominio/interfaces (amarillo) e implementaciones de repositorios (verde/morado). Muestra el flujo de dependencias entre componentes, donde los controladores consumen servicios, los servicios dependen de interfaces de repositorio, y estas se implementan en la capa de infraestructura. La estructura sigue principios de diseño limpio, con dependencias que apuntan hacia el centro del dominio.

Durante el desarrollo se han seguido principios de arquitectura hexagonal, también conocida como ports and adapters. Los elementos que coinciden con dicha arquitectura son los siguientes:

1. Separación clara de capas:

- Dominio (core): Los repositorios de dominio (interfaces) actúan como puertos (contratos que definen cómo interactuar con el exterior).
- Aplicación: Los servicios contienen la lógica de negocio y dependen de abstracciones (interfaces).
- Infraestructura: Las implementaciones repositorios son adaptadores que conectan con bases de datos o APIs externas.

2. Los servicios no dependen directamente de implementaciones, sino de interfaces.

3. Puertos de entrada (controladores) y salida (repositorios):

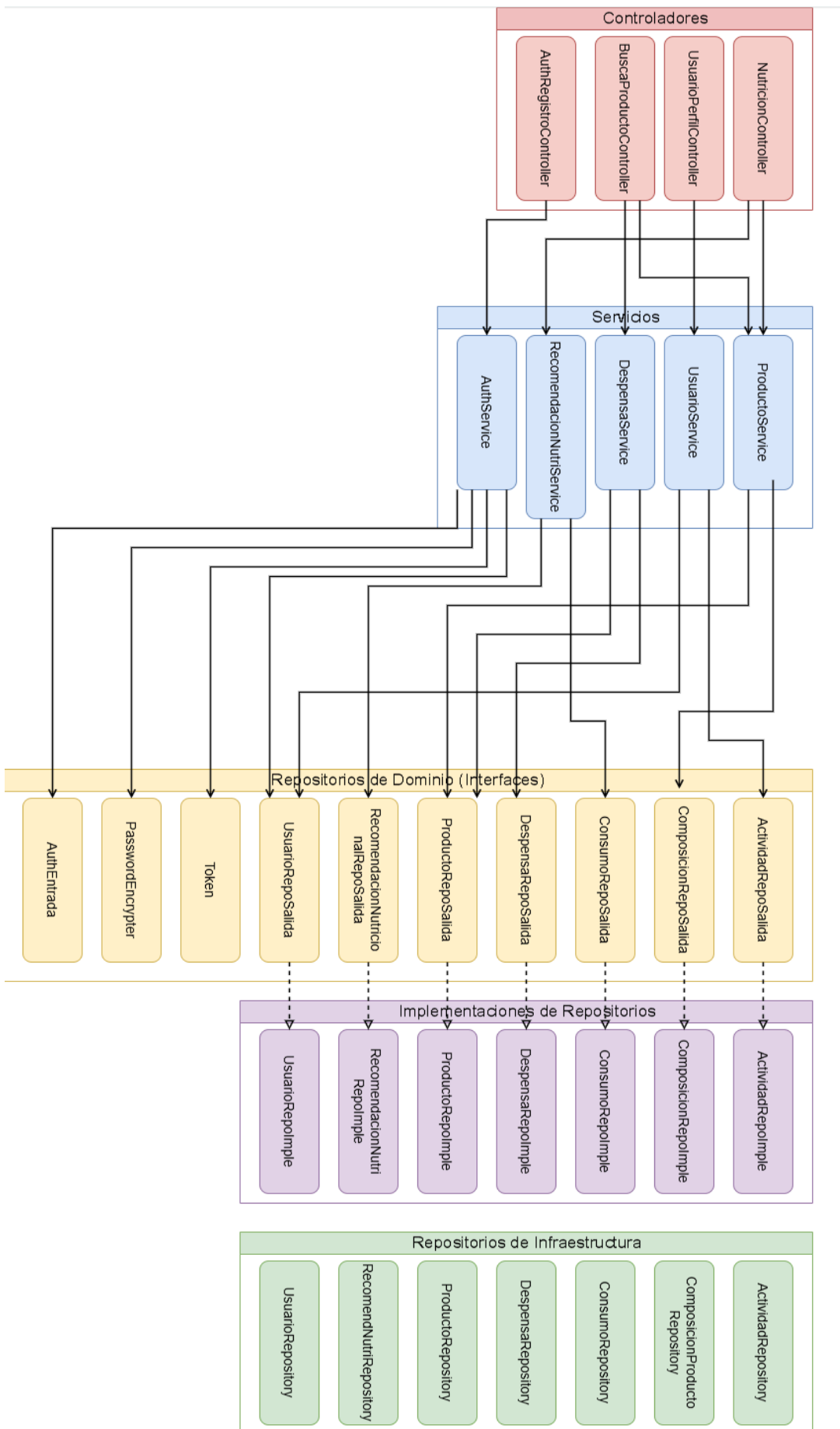
Puertos de entrada: Los controladores (como **AuthRegistroController**) son el punto de entrada de las peticiones HTTP.

Puertos de salida: Las interfaces (como **UsuarioRepoSalida**) definen cómo la aplicación se comunica con almacenamiento externo.

Entre los puntos a mejorar en siguientes versiones estaría entre otros que los controladores también deberían depender de interfaces (no directamente de servicios), pero aquí los controladores llaman a servicios concretos.

El sistema sigue principios de **arquitectura hexagonal**, pero con un enfoque pragmático más cercano a una *Clean Architecture* simplificada.

Se puede observar lo comentado en el siguiente diagrama:



Esta API REST ofrece endpoints para autenticación, gestión de productos, despensa personal y perfil de usuario. Permite a los usuarios registrarse, iniciar sesión con JWT, buscar alimentos con sus datos nutricionales, administrar su despensa y consumos, y consultar recomendación personalizada basada en su perfil. Todos los endpoints están diseñados para ser seguros, eficientes y fáciles de integrar en aplicaciones frontend.

Controlador	Método HTTP	Ruta	Descripción
Auth Registro Controller	POST	/api/auth/login	Autentica al usuario con email y contraseña, devuelve un JWT si es exitoso.
	POST	/api/auth/registro	Registra un nuevo usuario (valida email único y encripta contraseña).
Busca Producto Controller	GET	/api/producto?nombre={nombre}	Busca productos por nombre (ignora mayúsculas/minúsculas) y devuelve sus datos nutricionales.
Nutricion Controller	GET	/api/despensa/existe?productid={id}	Verifica si un producto existe en la despensa del usuario autenticado.
	GET	/api/usuario/despensa	Obtiene todos los productos de la despensa del usuario autenticado.
	POST	/api/despensa	Agrega un producto a la despensa del usuario.
	PUT	/api/despensa	Actualiza un producto en la despensa del usuario.
	GET	/api/usuario/consumos	Obtiene el historial de consumos del usuario

Controlador	Método HTTP	Ruta	Descripción
Usuario Perfil Controller	POST	/api/consumos	Agrega un producto al historial de consumos del usuario.
	GET	/api/usuario/perfil	Devuelve el perfil del usuario autenticado.
	PUT	/api/usuario/perfil	Actualiza el perfil del usuario y recalcula recomendaciones nutricionales.
	GET	/api/usuario/recomendacion-nutricional	Obtiene la recomendación nutricional personalizada del usuario.

Diagrama de casos de usos.

En este diagrama vemos un sistema con dos actores principales: **Usuario No Autenticado** (acceso público) y **Usuario Autenticado** (funcionalidades privadas). Los casos de uso incluyen acciones como registrarse, buscar productos, gestionar la despensa y obtener recomendación nutricional. La estructura diferencia claramente las funcionalidades accesibles para cada tipo de usuario, organizadas dentro del límite del sistema.

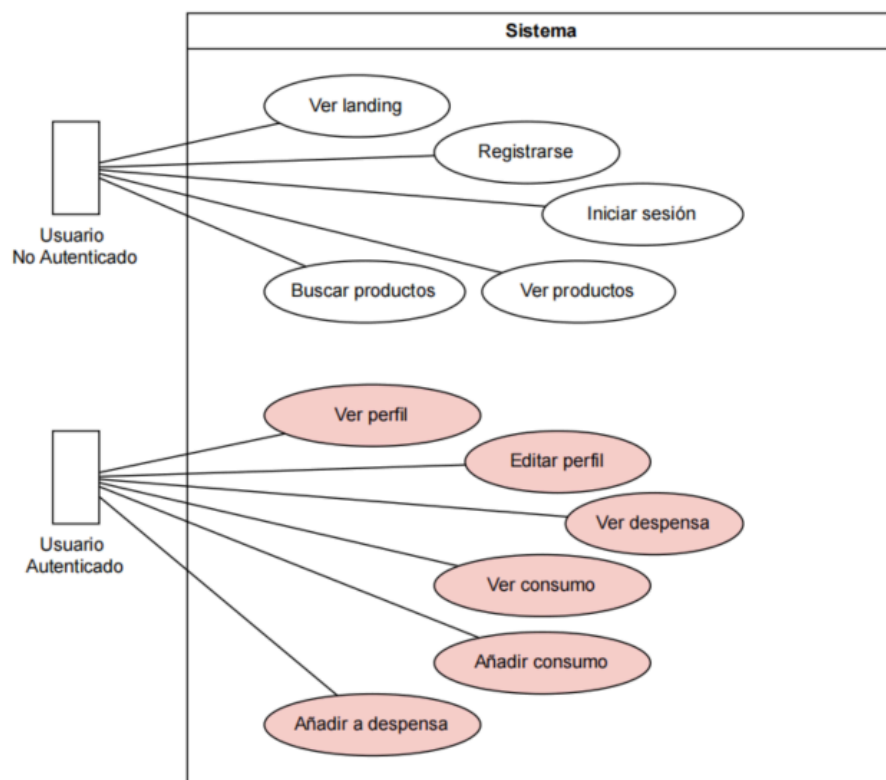
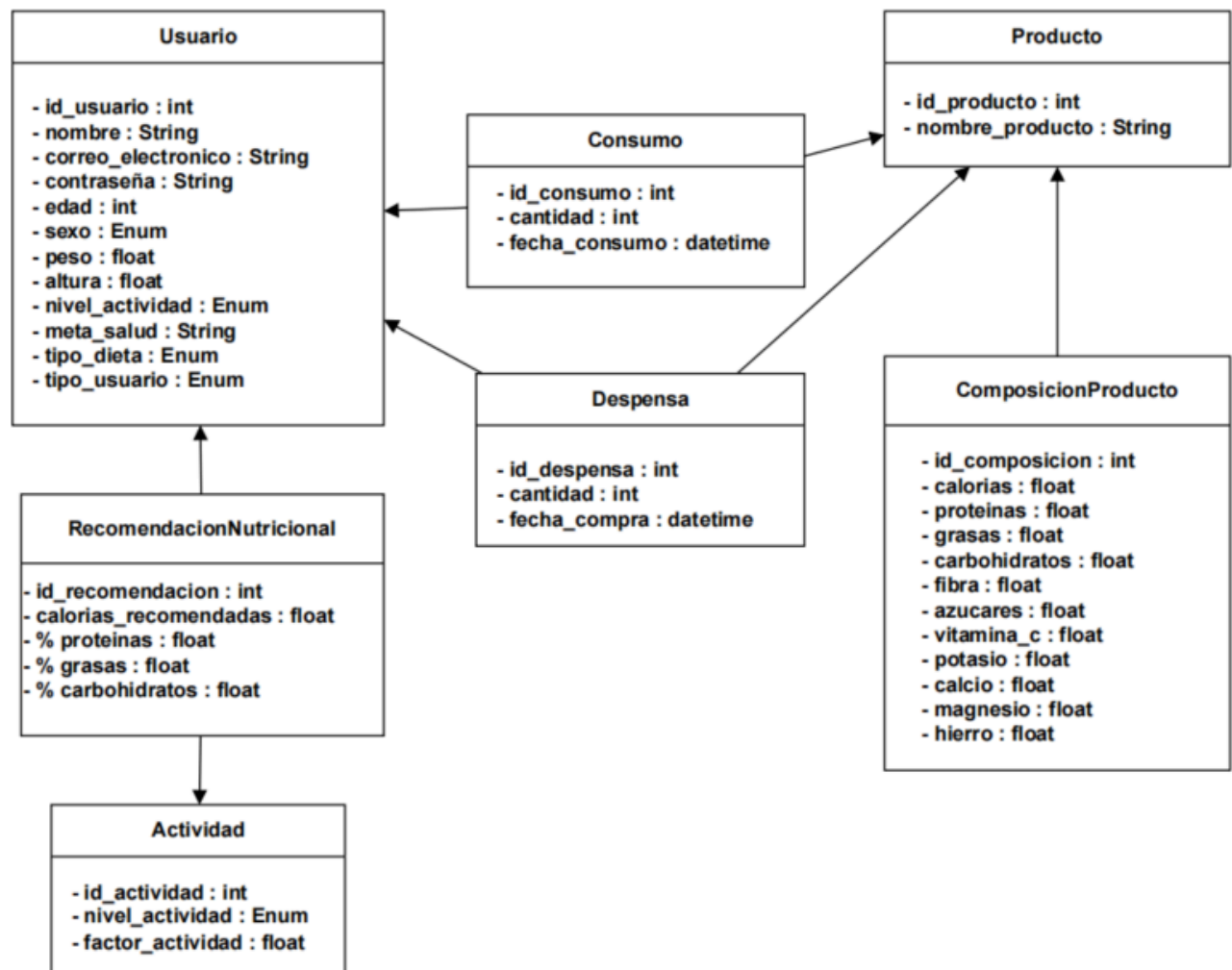


Diagrama de clases

Este diagrama representa el núcleo del sistema diseñado para ofrecer información de productos y seguimiento alimenticio. La arquitectura gira en torno al modelo Usuario, que almacena datos biométricos (peso, altura) y preferencias dietéticas y de actividad, y se relaciona con:

- **Producto** y su **ComposiciónProducto**
- **Despensa** (gestión de existencias) y **Consumo** (registro de ingestas)
- **RecomendacionNutricional**, generada según datos biométricos y el nivel de **Actividad** del usuario.



Diagramas de arquitectura de persistencia

El siguiente esquema detalla la arquitectura de persistencia del módulo de Usuario siguiendo un diseño hexagonal que separa claramente:

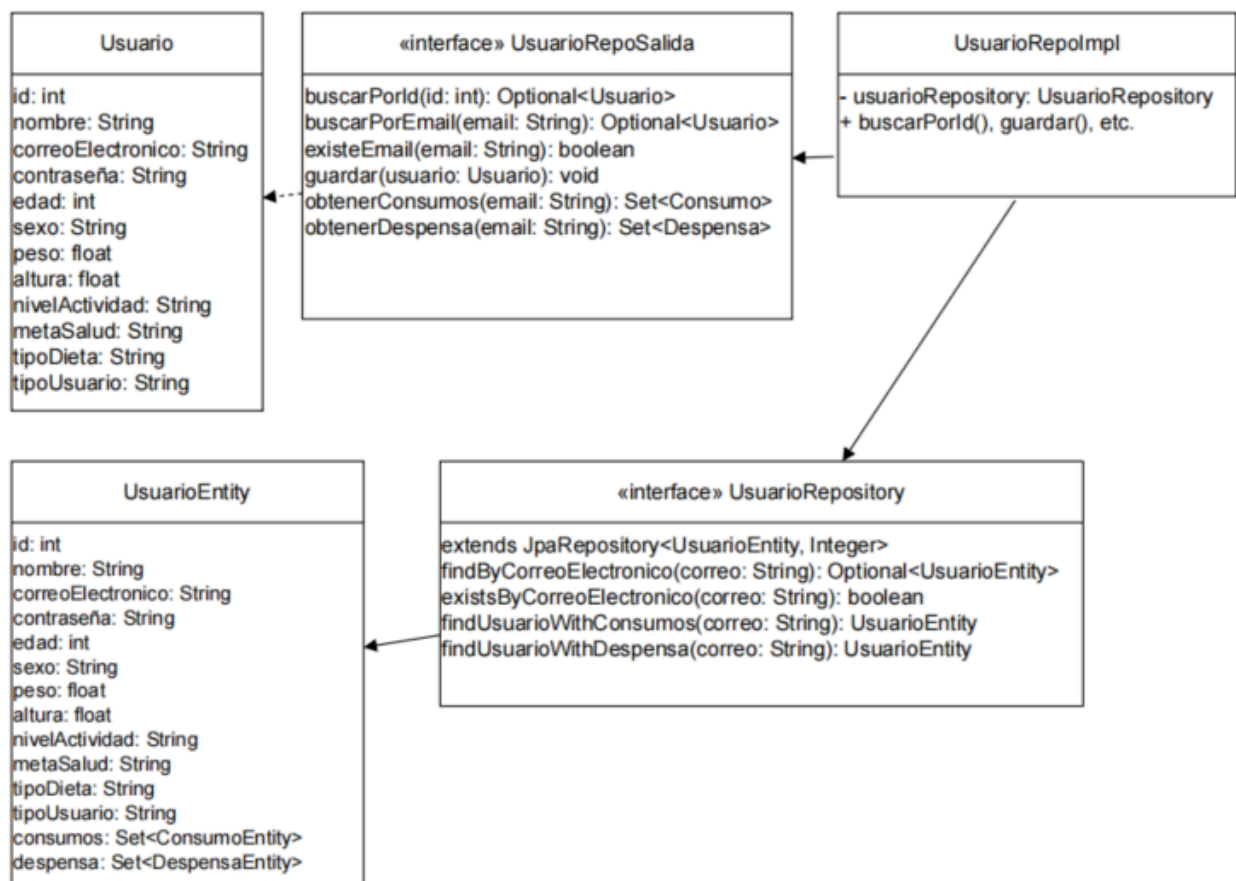
Capa de Dominio: La entidad **Usuario**(con atributos biométricos y preferencias dietéticas) define el modelo central.

La interfaz **UsuariosRepoSalida** actúa como puerto para operaciones de persistencia.

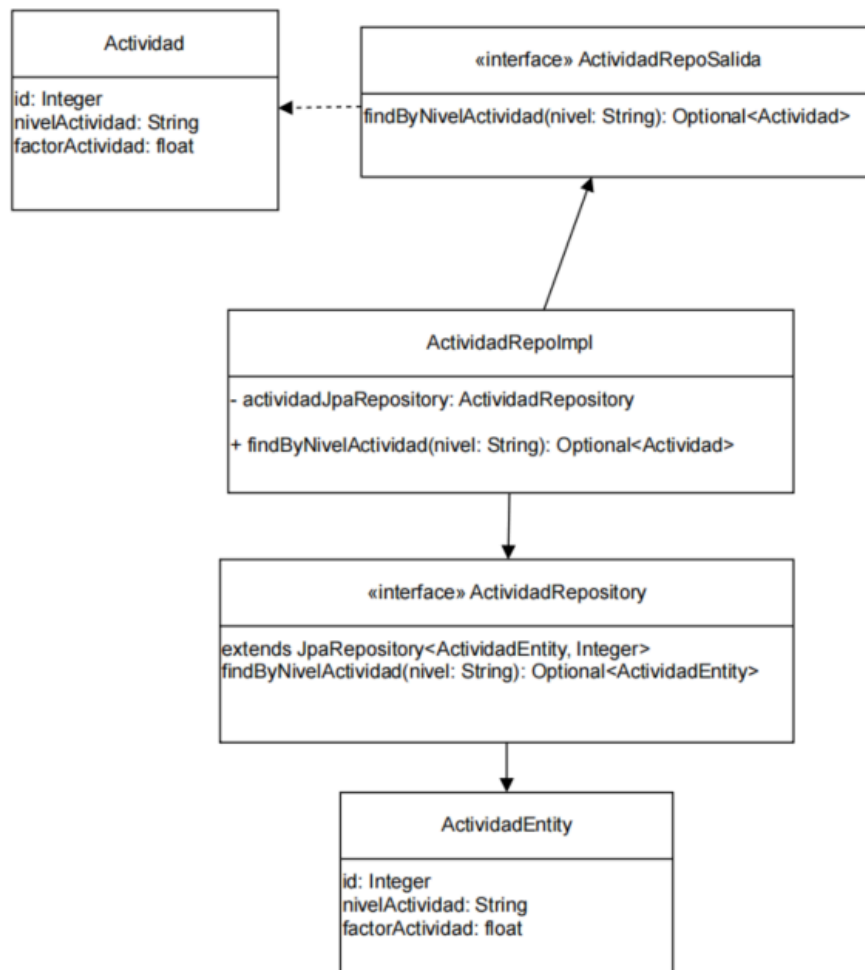
Capa de Infraestructura: `UsuarioRepoImpl` implementa el puerto anterior, delegando en `usuarioRepository` `usuarioRepository` (de Spring Data JPA) y transformando datos entre la entidad JPA (`UsuarioEntity` `UsuarioEntity`) y el modelo de dominio.

Servicio de Aplicación: UsuarioService orquesta la lógica (registro, login, gestión de perfiles), utilizando el puerto **UsuarioRepoSalida** y otros repositorios. Incluye métodos para conversión DTO/Entity y encriptación de contraseñas.

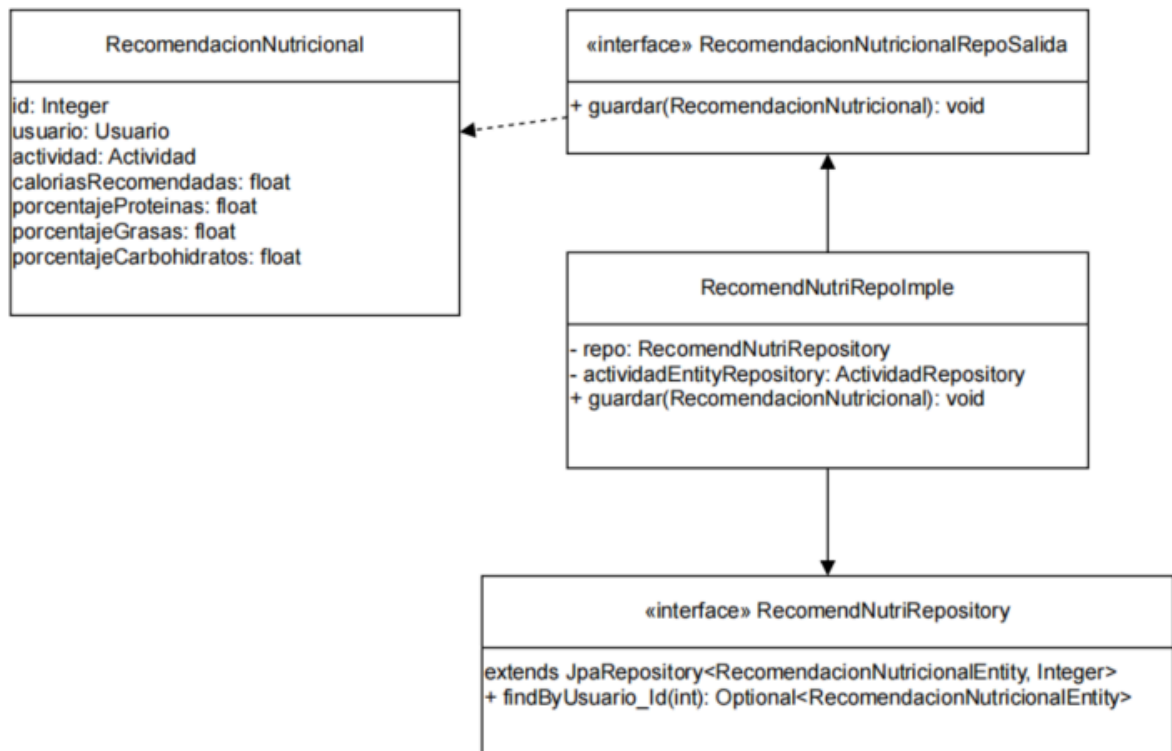
Las dependencias apuntan hacia el dominio (hexagonal), garantizando que la capa de aplicación no dependa directamente de frameworks externos. Las líneas discontinuas indican implementación de interfaces, mientras que las sólidas muestran llamadas concretas.



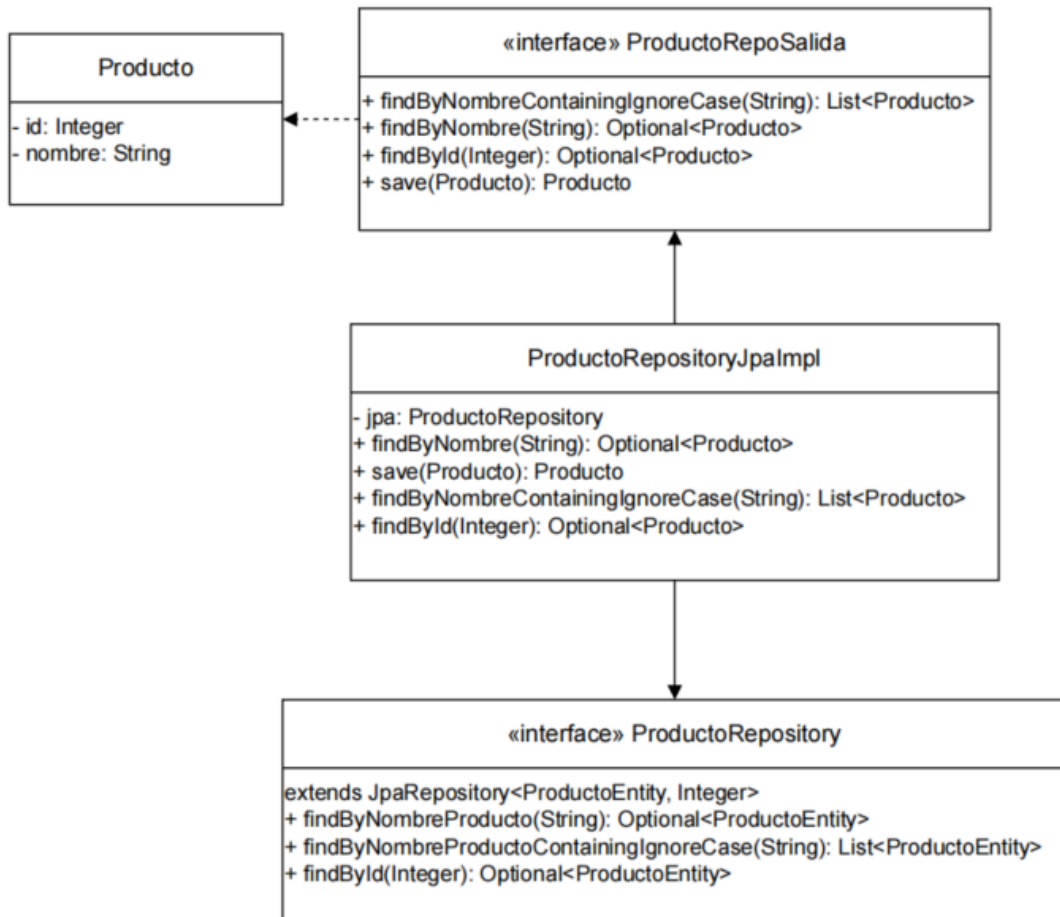
El siguiente diagrama representa la estructura de dominio e infraestructura para gestionar los niveles de **actividad física** en el sistema. Muestra desde la entidad de dominio hasta su persistencia en base de datos pasando por interfaces de repositorio y su implementación con Spring Data JPA. Las flechas indican dependencias, destacando la separación clara entre lógica de negocio y detalles técnicos.



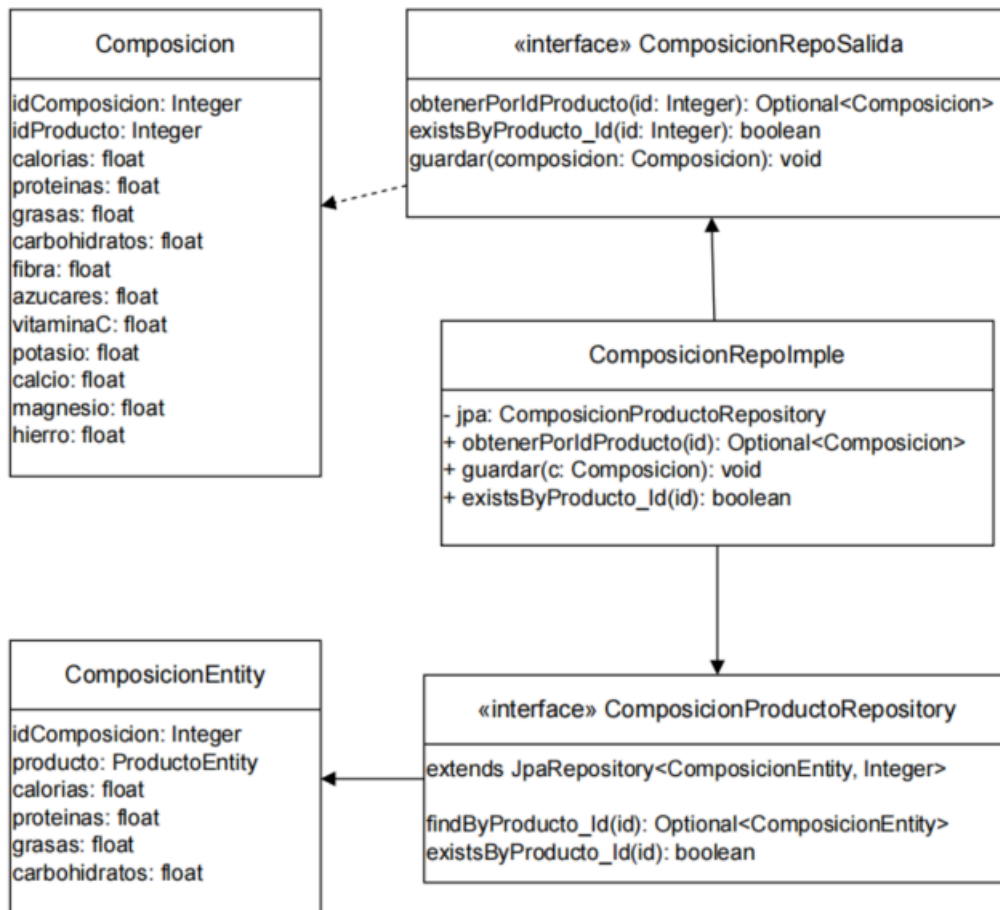
El siguiente diagrama describe la arquitectura para gestionar **recomendaciones nutricionales** personalizadas. La entidad de dominio (con atributos como calorías y macronutrientes) se comunica con la capa de persistencia a través de interfaces, mientras que la implementación concreta utiliza Spring Data JPA para almacenar los datos.



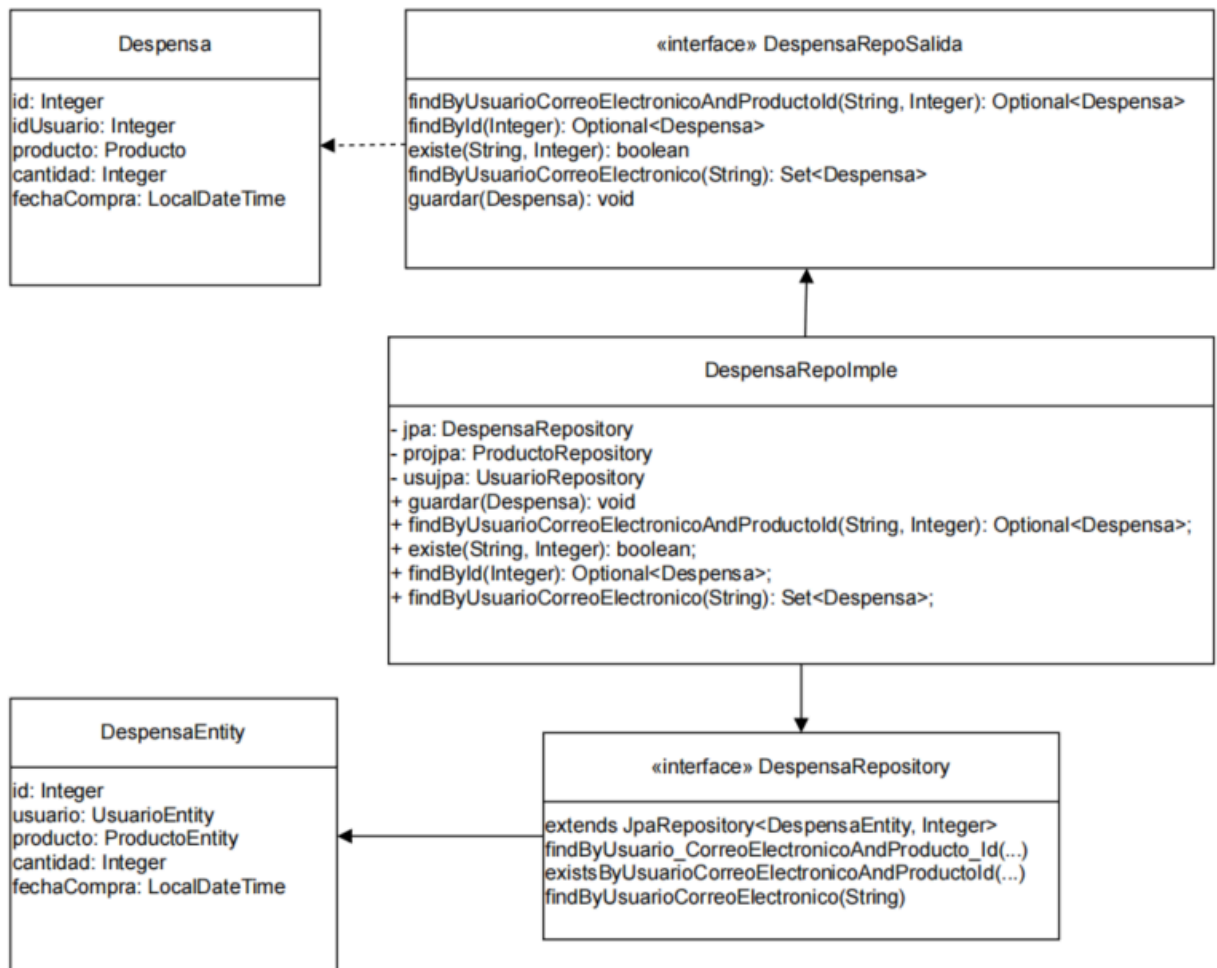
El siguiente diagrama describe la arquitectura para gestionar **productos** alimenticios. La entidad de dominio Producto (con atributos como ID y nombre) se comunica con la capa de persistencia a través de interfaces, mientras que la implementación concreta utiliza Spring Data JPA para almacenar y recuperar los datos de productos.



El siguiente diagrama describe la arquitectura para gestionar la información de composición de productos alimenticios. La entidad de dominio **Composición** (con atributos como calorías, proteínas, grasas y micronutrientes) se comunica con la capa de persistencia a través de interfaces, mientras que la implementación concreta utiliza Spring Data JPA para almacenar y recuperar los datos nutricionales.



El siguiente diagrama describe la arquitectura para gestionar los productos almacenados en la **despensa** de los usuarios. La entidad de dominio Despensa (con atributos como producto, cantidad y fecha de compra) se comunica con la capa de persistencia a través de interfaces, mientras que la implementación concreta utiliza Spring Data JPA para almacenar y recuperar los datos.



El siguiente diagrama describe la arquitectura para registrar el **consumo** de productos por parte de los usuarios. La entidad de dominio Consumo (con información de producto, cantidad y fecha de consumo) se conecta con la capa de persistencia mediante interfaces, implementando el almacenamiento mediante Spring Data JPA.

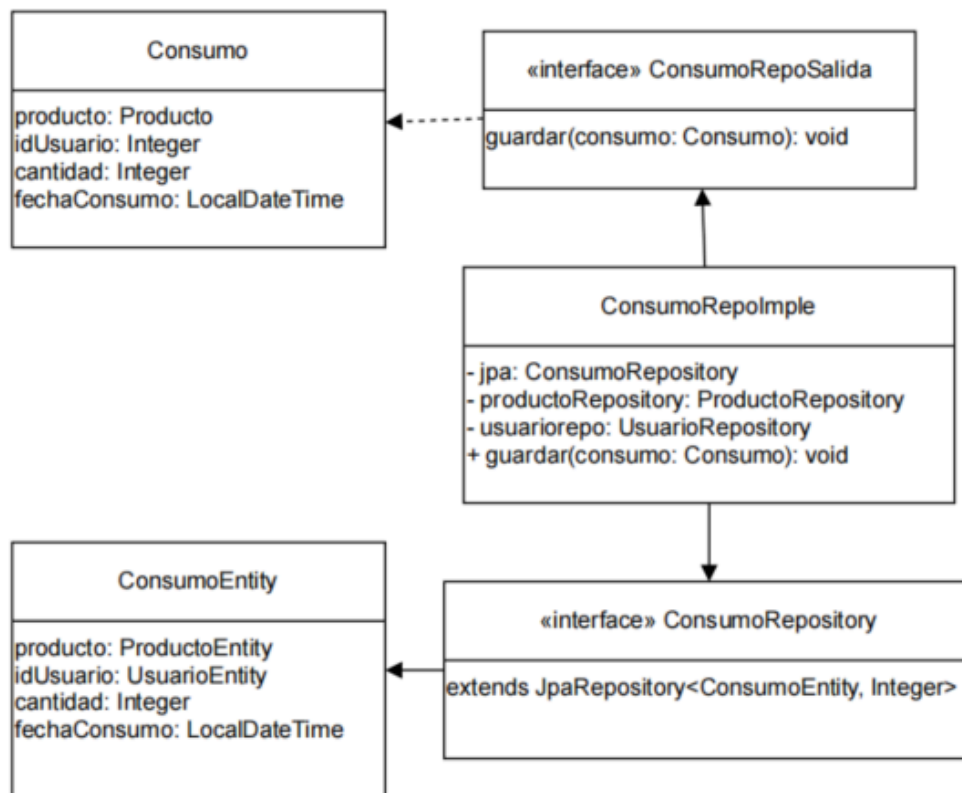
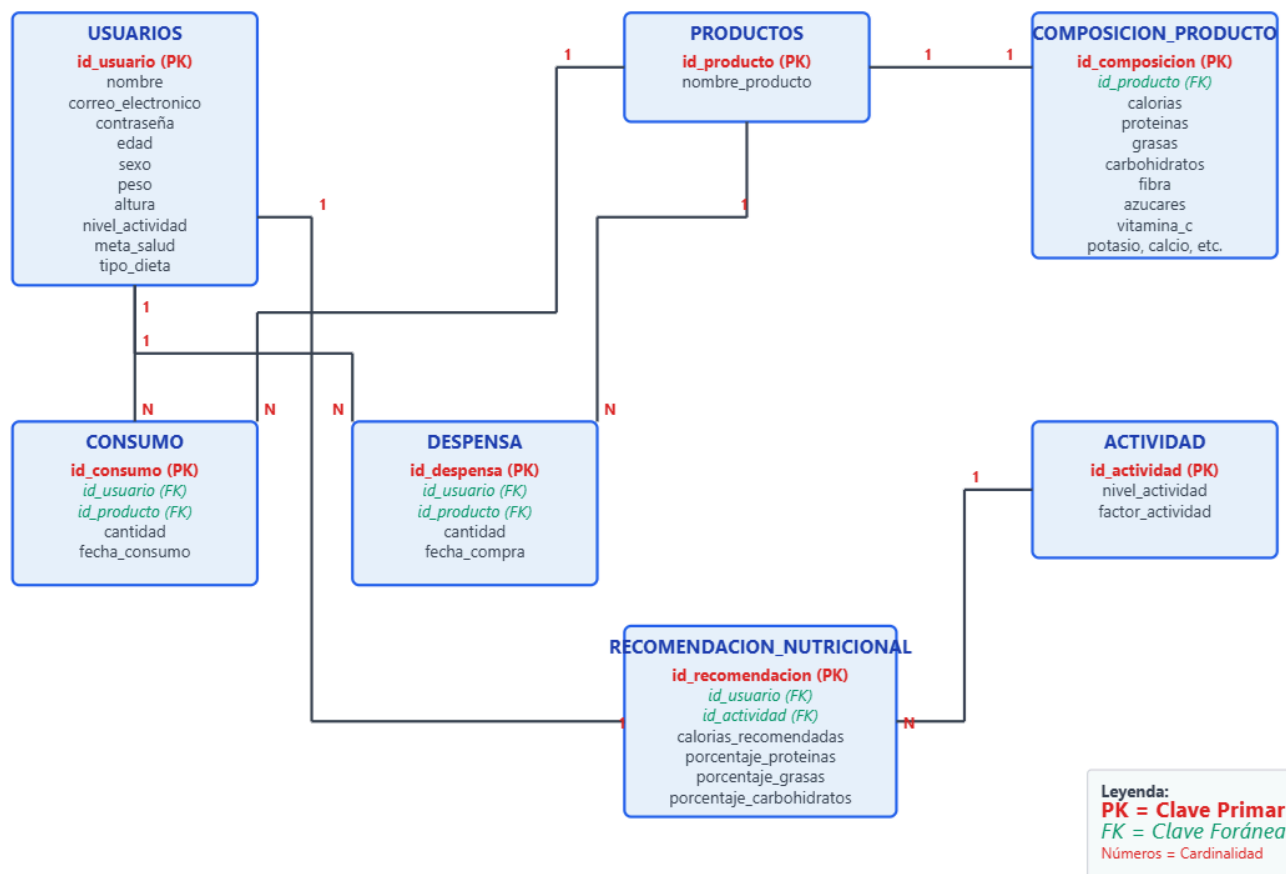


Diagrama Entidad relación

El esquema muestra cómo los usuarios se relacionan con sus productos (tanto en despensa como en consumo), sus recomendaciones nutricionales basadas en su nivel de actividad, y cómo cada producto tiene su información nutricional detallada.



Las cardinalidades principales son:

Relaciones 1:1:

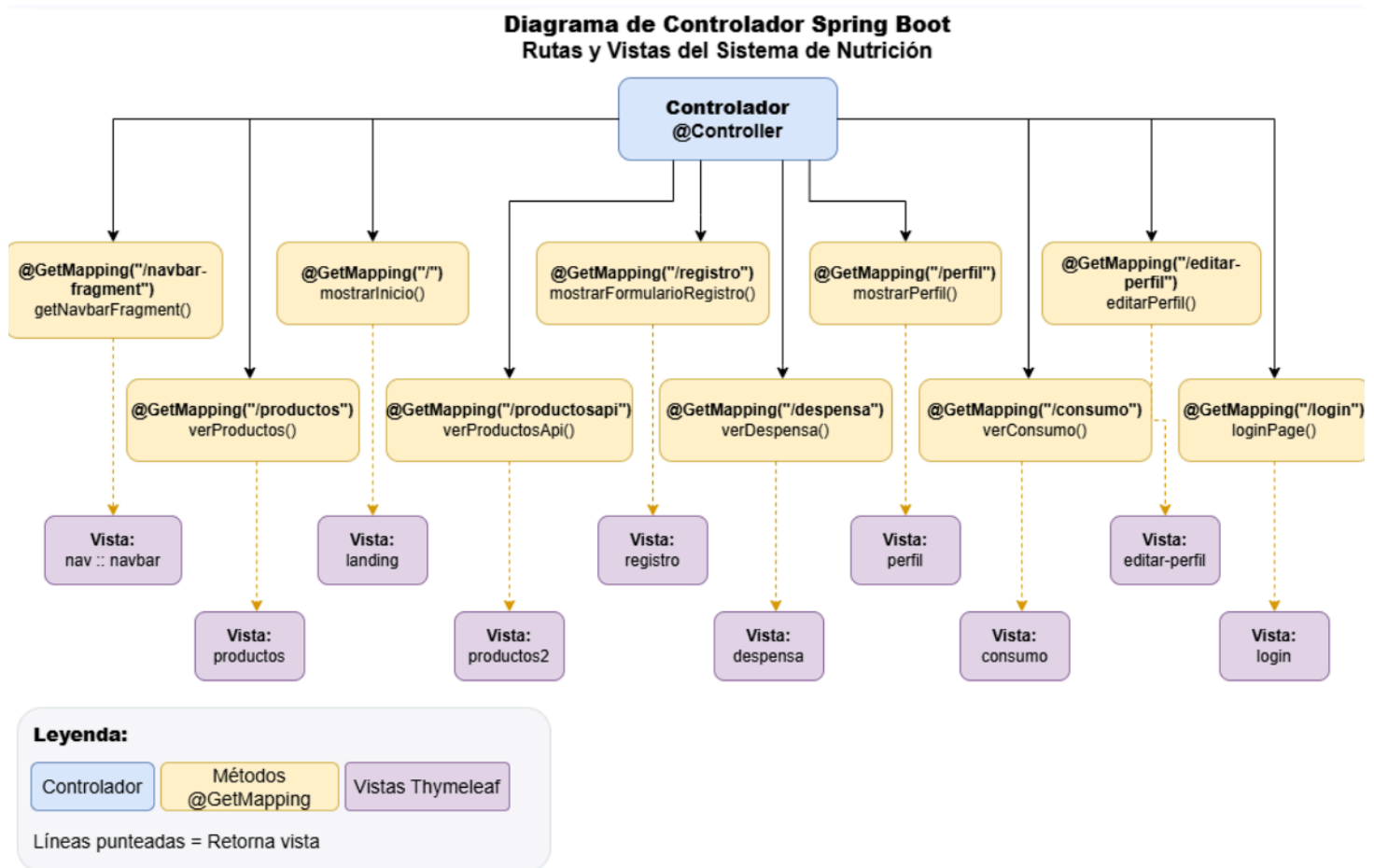
- Usuarios ↔ Recomendacion_Nutricional (cada usuario tiene una recomendación nutricional)
- Productos ↔ Composicion_Producto (cada producto tiene una composición nutricional)

Relaciones 1:N:

- Usuarios → Despensa (un usuario puede tener muchos productos en su despensa)
- Usuarios → Consumo (un usuario puede tener muchos registros de consumo)
- Productos → Despensa (un producto puede estar en muchas despensas)
- Productos → Consumo (un producto puede ser consumido muchas veces)
- Actividad → Recomendacion_Nutricional (un nivel de actividad puede tener varias recomendaciones)

Diagrama de Controlador Spring MVC

Este diagrama muestra un controlador Spring MVC que sirve plantillas Thymeleaf para el renderizado inicial, pero la arquitectura va más allá.



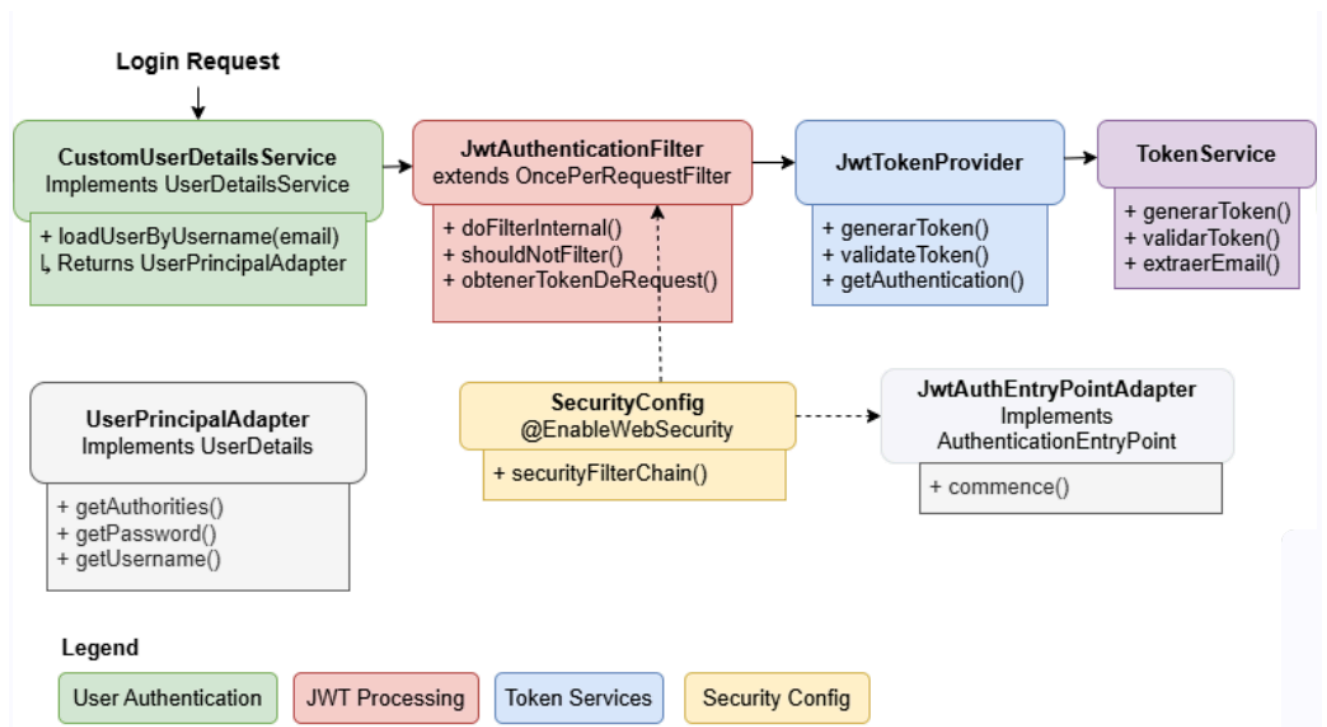
Cada vista entregada por estos endpoints incorpora lógica JavaScript que enriquece la experiencia ofreciendo:

- **Dinamismo post-carga:** Tras el renderizado inicial, scripts específicos realizan llamadas `fetch()` a endpoints REST `api/prod` para actualizar datos sin recargar la página. Por ejemplo, la vista de productos muestra primero un esqueleto básico con Thymeleaf, y luego JavaScript carga dinámicamente los detalles nutricionales desde una API.
- **Interacción fluida:** Validaciones de formularios, filtros en tiempo real o animaciones (como añadir productos a la despensa) se gestionan del lado del cliente, reduciendo la carga del servidor y mejorando la percepción de velocidad.

- **SEO y accesibilidad:** El renderizado inicial con Thymeleaf garantiza que los motores de búsqueda y usuarios sin JavaScript vean contenido completo.
- **Flexibilidad:** Las plantillas pueden incluir datos contextuales iniciales (como el ID de usuario) que los scripts aprovechan para personalizar las llamadas API posteriores.

Diagrama de flujo de autenticación con JWT

En cuanto a la seguridad y la autenticación, el sistema implementa un flujo completo de autenticación basado en tokens JWT, diseñado con una clara separación de responsabilidades.



Flujo de autenticación

1. La solicitud de inicio de sesión es gestionada por **CustomUserDetailsService**.
2. **JwtFilter** valida el token en las solicitudes posteriores.
3. Los servicios de token se encargan de la generación y validación del **JWT**.
4. **SecurityConfig** orquesta el flujo.

Proceso de Autenticación Inicial: El **CustomUserDetailsService** actúa como núcleo de autenticación, validando las credenciales del usuario contra el repositorio y generando un objeto **UserPrincipalAdapter** que encapsula los detalles del usuario y sus permisos.

Filtrado y Validación de Tokens: El **JwtAuthenticationFilter**, integrado en la cadena de seguridad de Spring, intercepta cada solicitud para extraer el token **JWT** del header **Authorization**. Mediante el **JwtTokenProvider**, verifica la validez del token (firma, expiración) y establece la autenticación en el contexto de seguridad.

Gestión de Tokens: La capa de servicios (**TokenService** y **JwtTokenProvider**) se encarga de toda la lógica criptográfica: generación de tokens firmados, validación de firma, y extracción de claims. Utiliza propiedades configurables como la clave secreta y el tiempo de expiración.

Configuración de Seguridad Centralizada: La clase **SecurityConfig** define el comportamiento global: deshabilita CSRF para APIs RESTful, configura el manejo stateless de sesiones, especifica rutas públicas/protegidas, e integra el filtro **JWT** y el manejador de excepciones (**JwtAuthEntryPoint**).

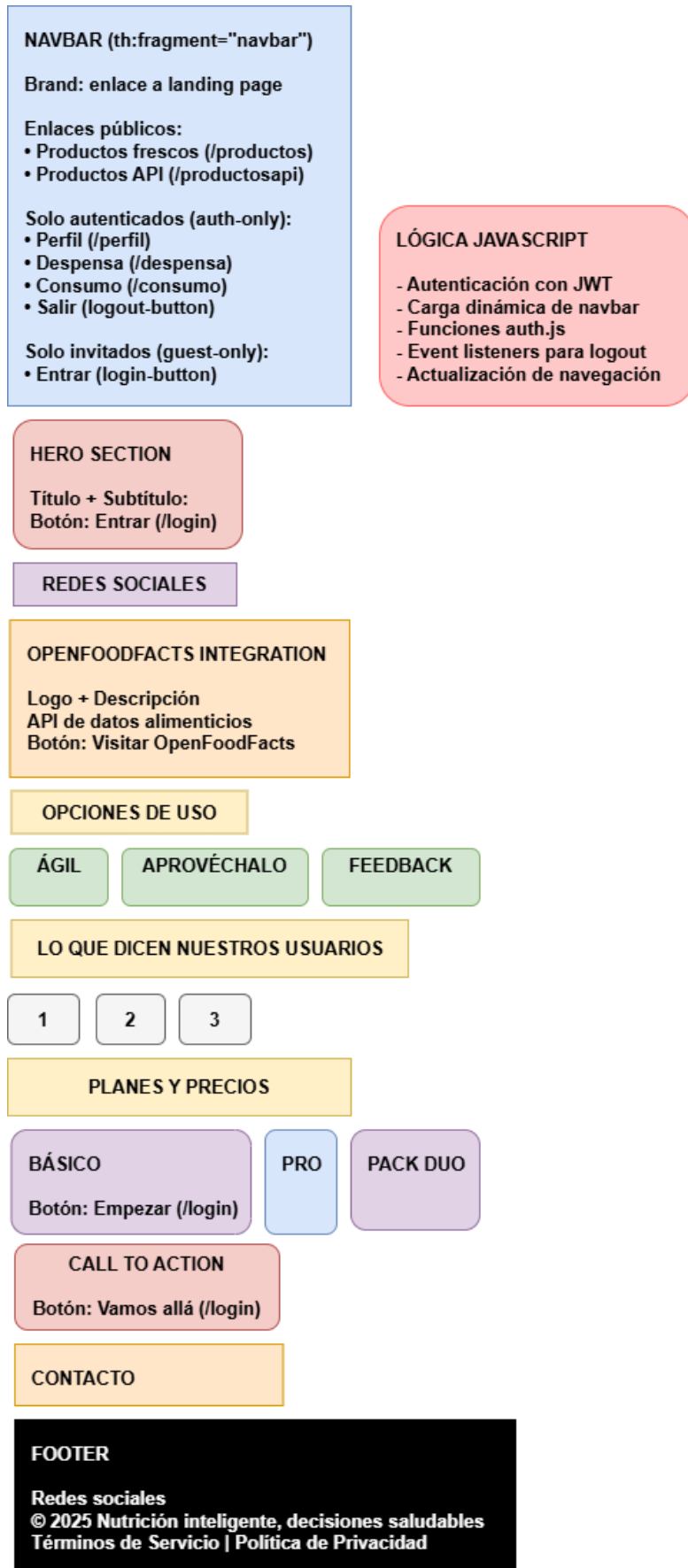
Integración con el Contexto de Spring: Una vez autenticado, el sistema permite el acceso a los detalles del usuario en cualquier punto de la aplicación a través del **SecurityContextHolder**, mientras que las vistas interactúan dinámicamente mediante JavaScript, consumiendo los endpoints protegidos.

UserPrincipalAdapter y **JwtAuthEntryPointAdapter** cumplen roles clave en la autenticación JWT.

El **UserPrincipalAdapter** adapta la entidad usuario al modelo que Spring Security entiende. Implementa **UserDetailUserDetails** para proveer métodos. Por ejemplo, cuando un usuario hace login, este **Adapter** permite a Spring Security validar su contraseña encriptada contra la recibida.

JwtAuthEntryPointAdapter maneja errores de autenticación (401 Unauthorized). Implementa **commence()** que se ejecuta cuando el token **JWT** es inválido o malformado.

Diagramas de flujo de cada vista



Presenta una página inicial estructurada en flujo vertical, comenzando con un navbar dinámico que adapta sus opciones según el estado de autenticación del usuario (invitado vs autenticado), seguido de una sección hero con call-to-action principal.

La página continúa con bloques informativos sobre características del producto, testimonios de usuarios y planes de precios, culminando en secciones de contacto y footer, todo integrado con la API de OpenFoodFacts para datos nutricionales.

El flujo está diseñado para guiar al usuario desde el descubrimiento inicial hasta la conversión (registro/login), con múltiples puntos de entrada distribuidos estratégicamente a lo largo de la página.

Login

Inicio con Thymeleaf:

La página de login se carga inicialmente desde el servidor con Thymeleaf, mostrando la estructura base y el formulario de acceso.

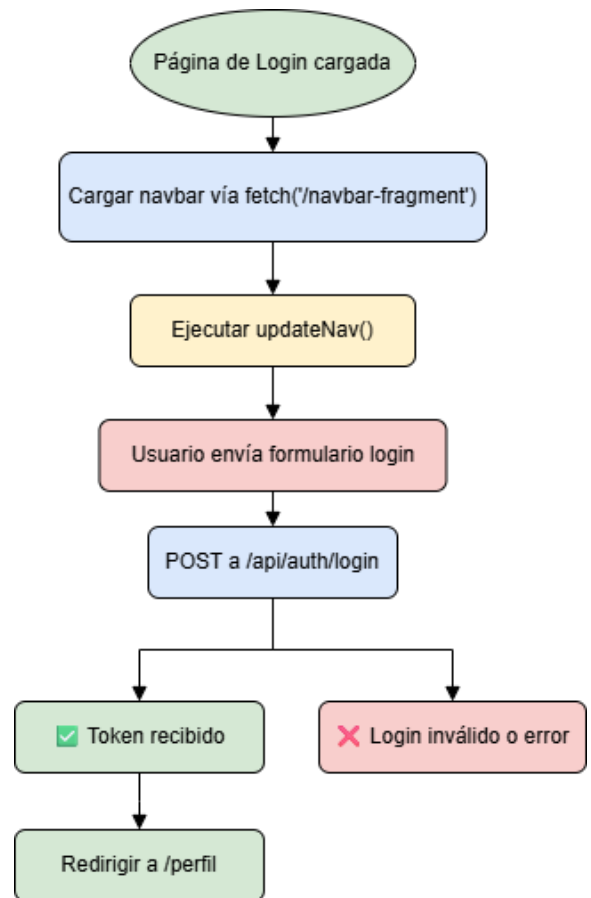
Dinamismo con JavaScript:

El navbar se carga de forma asíncrona mediante `fetch('/navbar-fragment')` para mantenerlo actualizado sin recargar la página.

`updateNav()` personaliza el navbar según el estado de autenticación.

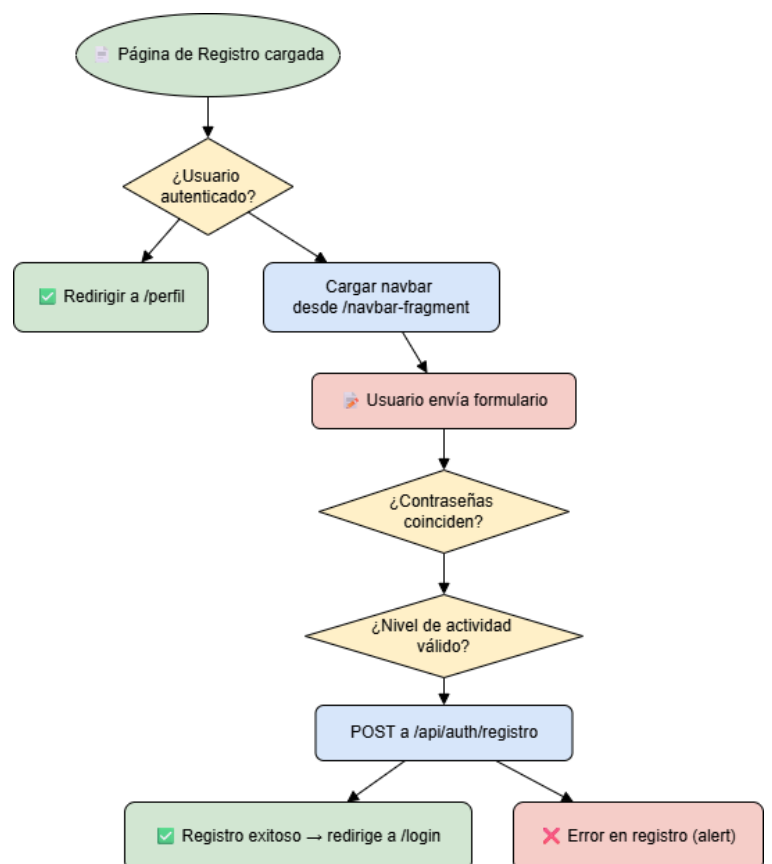
Al enviar el formulario, se realiza una llamada POST `/api/auth/login` para autenticar al usuario.

Muestra caminos para éxito (token JWT recibido → redirección a `/perfil`) y fallo (feedback visual de error), destacando la integración frontend-backend mediante llamadas REST.



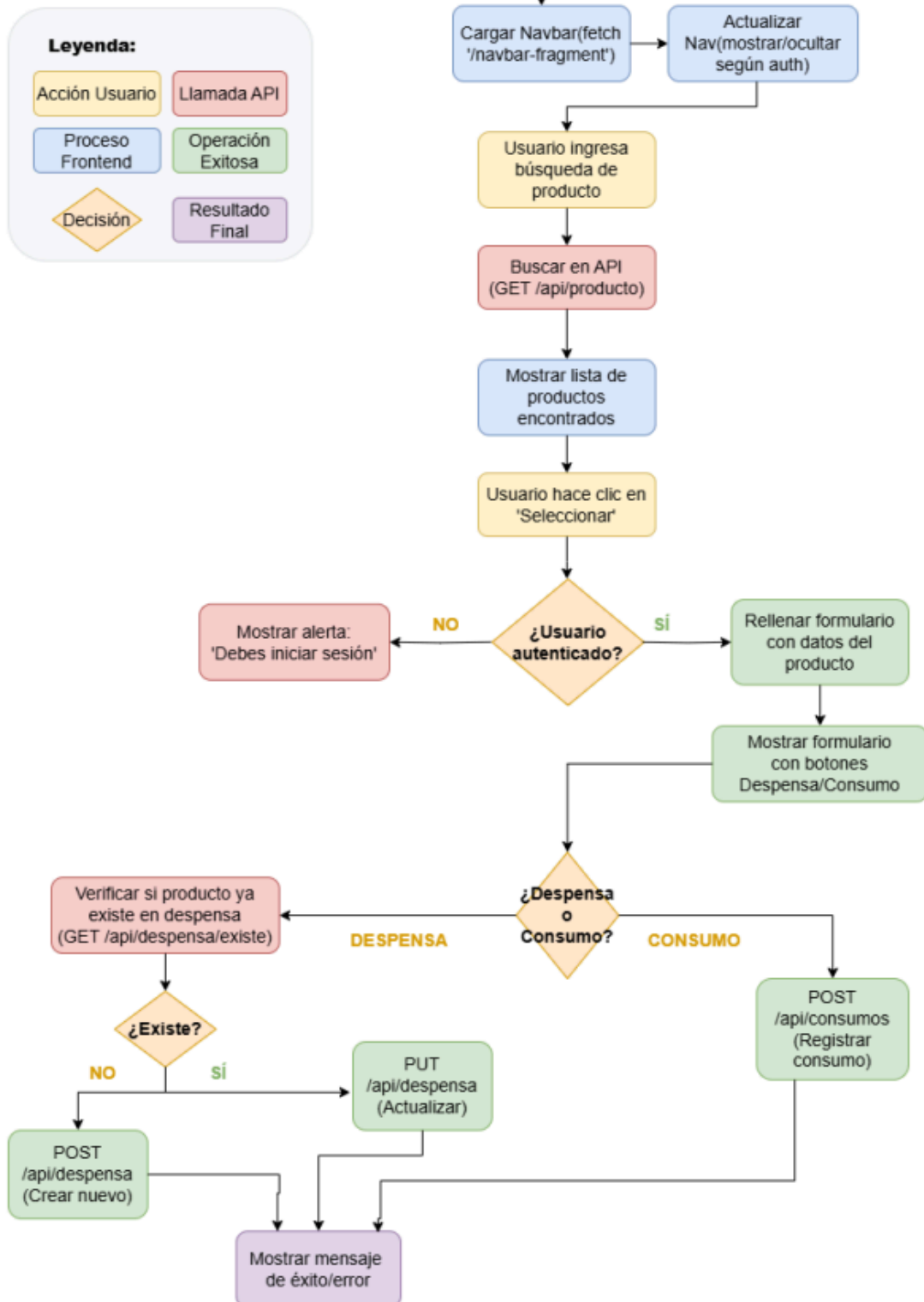
Registro

Se carga verificando primero si el usuario ya está autenticado (mediante token JWT). Si es así, redirige directamente al perfil; si no, carga dinámicamente el navbar para obtener consistencia. El formulario chequea que las contraseñas coinciden y que los campos de actividad física sean válidos antes de enviar datos a `POST /api/auth/registro`. El endpoint verifica unicidad de email y requisitos nutricionales. Si hay éxito redirige a login, sino muestra alert.



Productos

Flujo del Sistema de Productos Búsqueda, Selección y Gestión de Despensa/Consumo



La página carga inicialmente el navbar mediante `fetch('/navbar-fragment')` y adapta su contenido según el estado de autenticación.

Las búsquedas se realizan mediante llamadas a la API (`GET /api/producto`) mientras el usuario escribe, mostrando resultados al instante.

Al seleccionar un producto Frontend verifica si el usuario está autenticado (muestra alerta si no lo está). Después Backend, para despensa, chequea si el producto ya existe (`GET /api/despensa/existe`) para decidir entre crear (`POST`) o actualizar (`PUT`).

Despensa: Gestiona cantidades y fechas de compra, con actualizaciones en tiempo real.

Consumo: Registra ingesta nutricional vinculada al perfil del usuario.

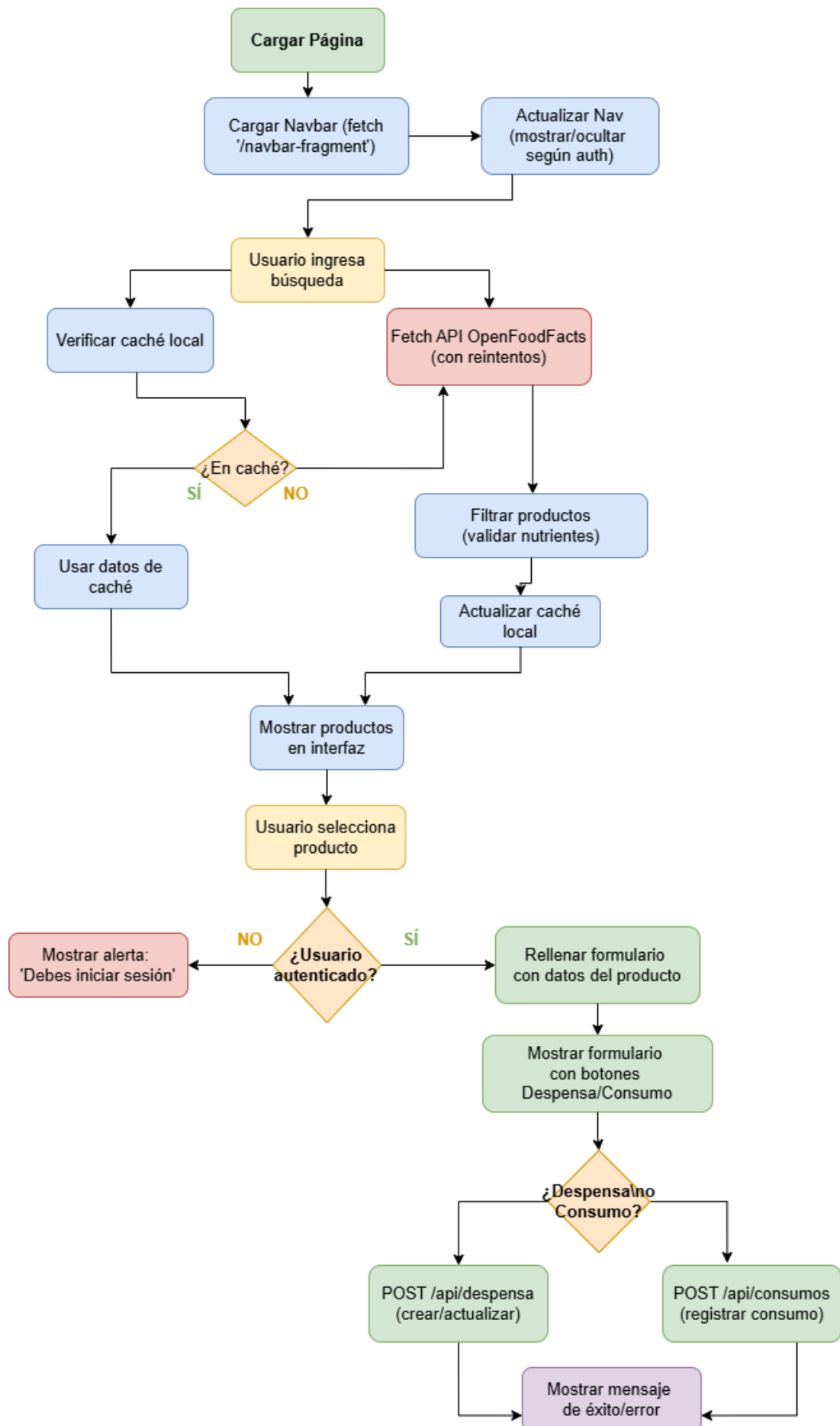
Productos de API Open food facts

El sistema primero verifica si los productos buscados están en caché local antes de llamar a la API externa (OpenFoodFacts), optimizando el rendimiento y reduciendo llamadas innecesarias. Los resultados se filtran para validar datos nutricionales completos antes de mostrarse.

La interfaz se actualiza dinámicamente: desde la carga inicial del navbar (adaptado al estado de autenticación) hasta la visualización de productos, usando JavaScript para mantener fluidez.

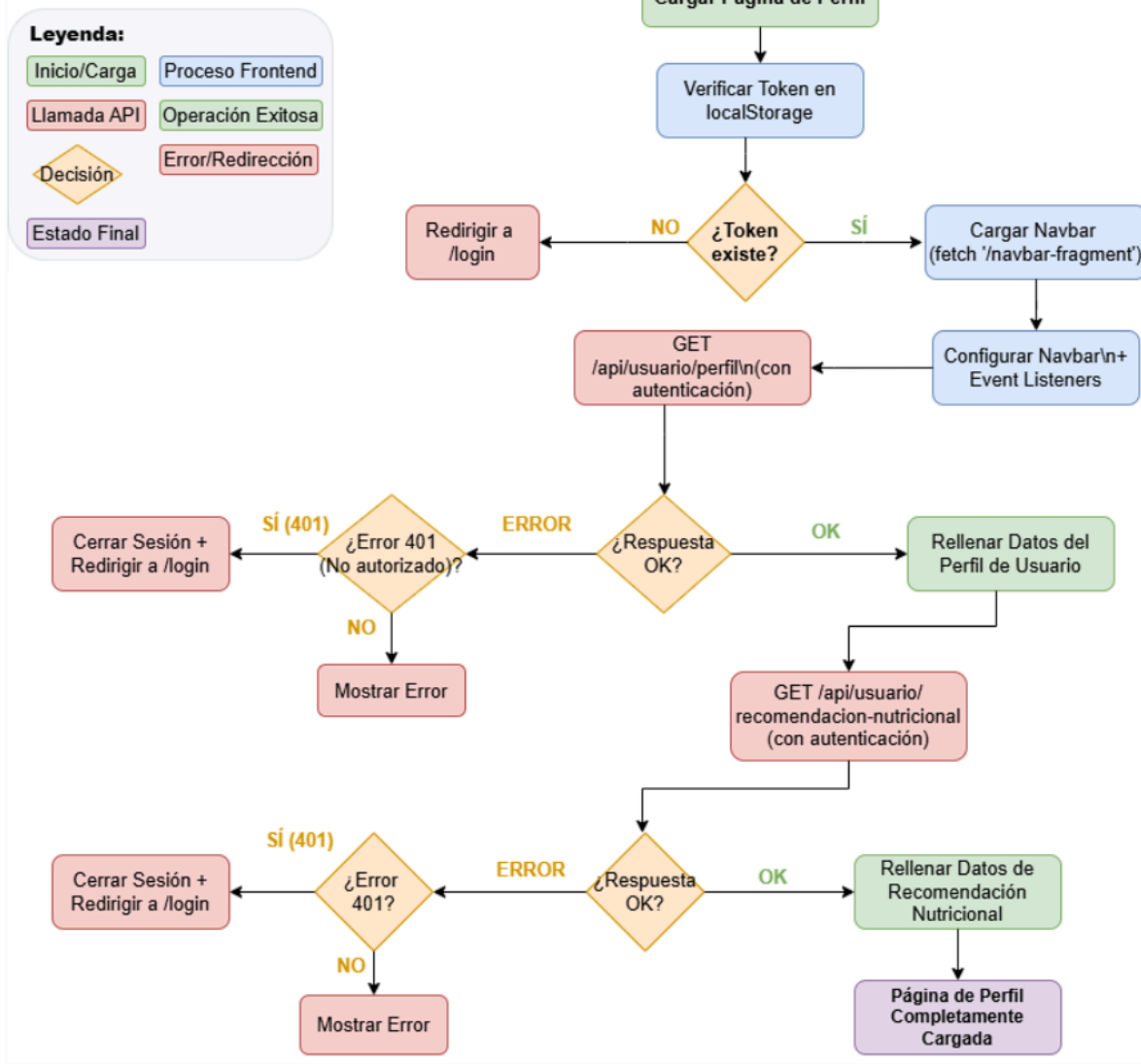
Los datos válidos de la API externa se almacenan en caché para futuras búsquedas.

Tras seleccionar un producto, el sistema valida si el usuario está autenticado (muestra alerta si no), permite elegir entre añadir a la despensa (con verificación de existencia) o registrar consumo y confirma la acción con feedback visual, todo mediante interacciones AJAX sin recargar la página.



Perfil

Flujo de Carga de Página de Perfil Autenticación, Carga de Datos y Manejo de Errores



El sistema primero valida la presencia del token JWT en **LocalStorage**. Si no existe, redirige inmediatamente a /login para autenticación. Con token válido, carga dinámicamente el **navbar** y configura los listeners de eventos.

Primera llamada API: GET /api/usuario/perfil para información básica.

Segunda llamada API: GET /api/usuario/recomendacion-nutricional. Se ejecuta tras éxito en la primera llamada. Visualización de Información.

Ejemplo de flujo completo: Usuario válido → carga navbar → obtiene perfil → obtiene recomendación → muestra todos los datos

Editar perfil

El sistema valida el token JWT almacenado (redirige a login si es inválido)

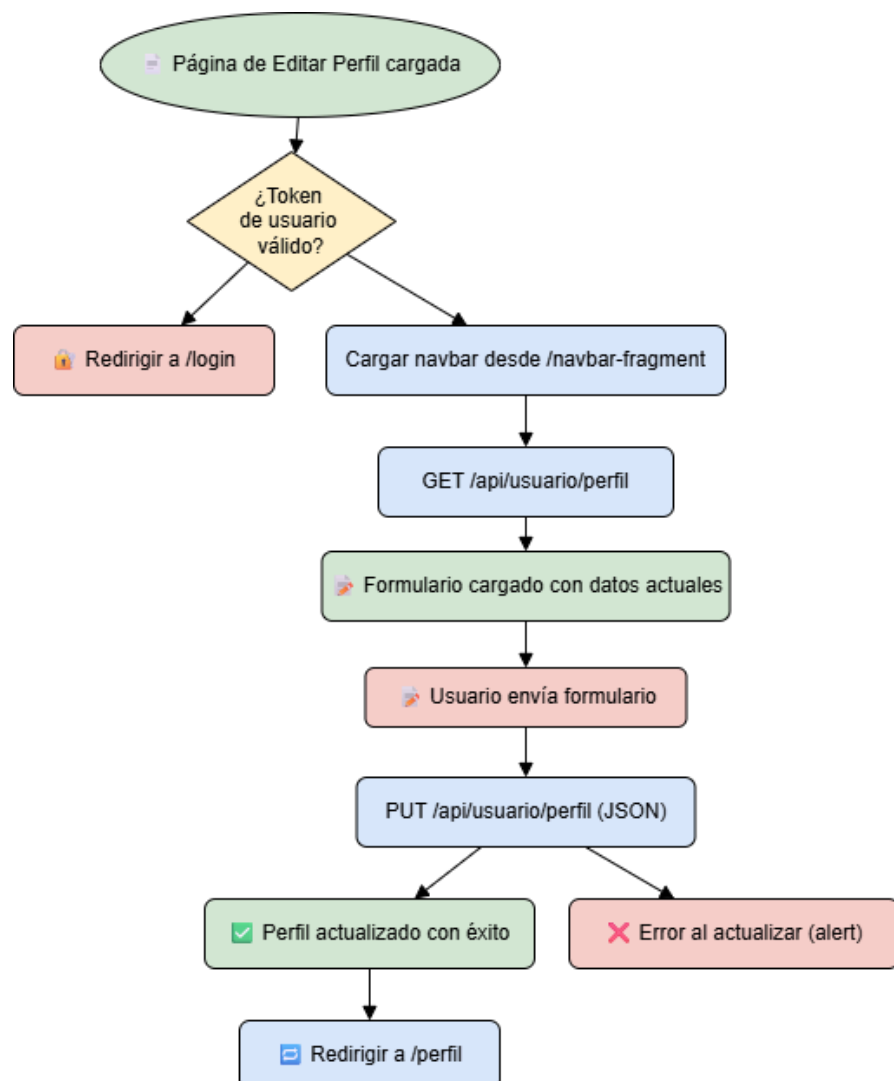
Carga dinámicamente el navbar actualizado mediante `fetch('/navbar-fragment')` y refleja el estado de autenticación.

Obtiene los datos actuales del perfil mediante `GET /api/usuario/perfil`

Autocompleta el formulario de edición con la información existente.

Al enviar el formulario se validan los campos en frontend (contraseñas coincidentes, formatos correctos), se envía la actualización mediante `PUT /api/usuario/perfil` con los datos en JSON. El manejo de errores muestra alertas específicas para problemas de validación. Con la actualización exitosa muestra confirmación visual y redirige a perfil para ver cambios. El flujo tiene validación en dos capas (frontend + backend) y persistencia de token JWT para sesión continua. El contenido se actualiza sin recargar página.

Ejemplo: Usuario modifica su peso y nivel de actividad → sistema recalcula automáticamente las recomendaciones nutricionales

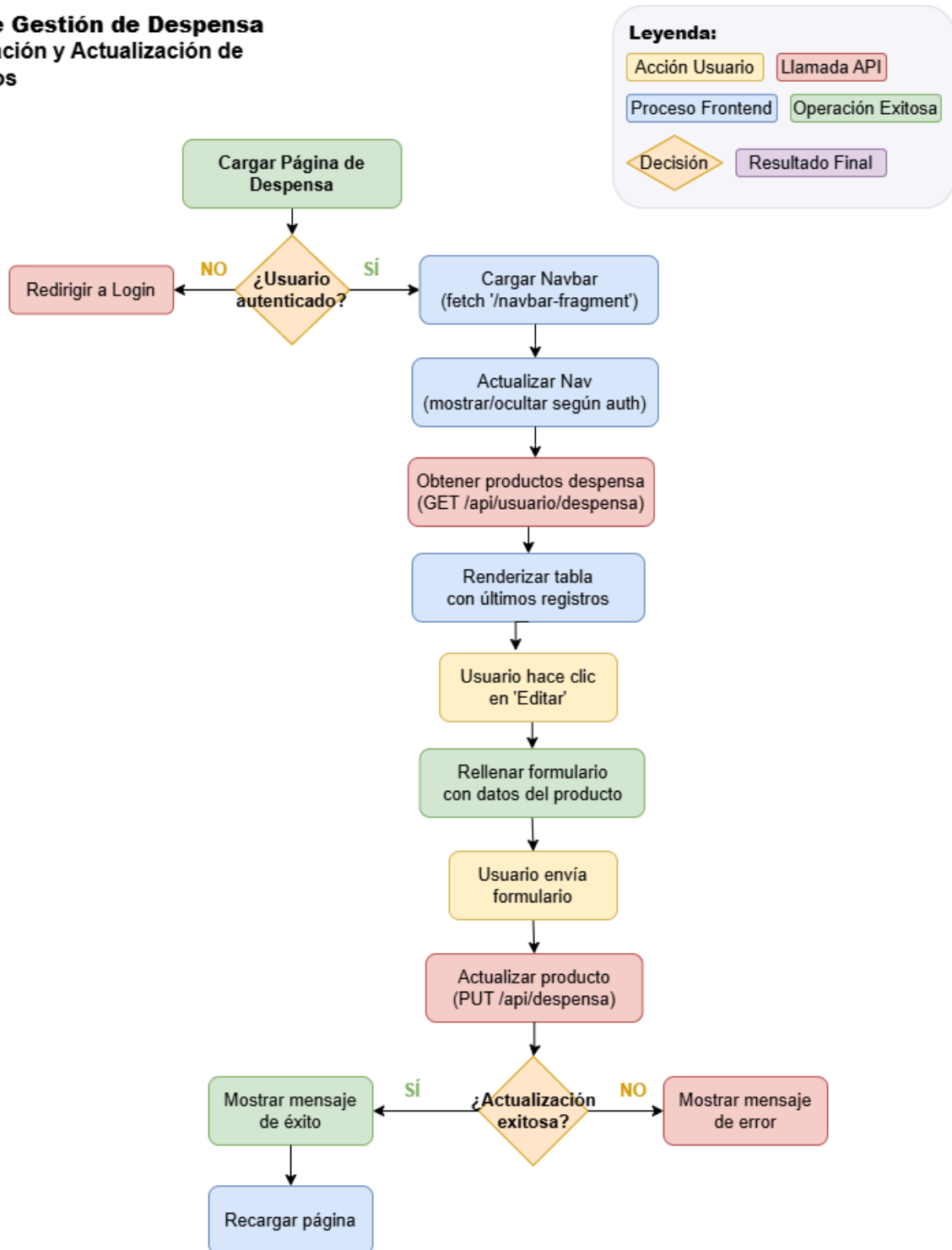


Despensa

Verifica autenticación, carga dinámicamente el navbar y obtiene los productos de despensa mediante GET /api/usuario/despensa. Renderiza una tabla interactiva con los últimos productos añadidos.

Al hacer clic en "Editar": Autocompleta un formulario con los datos actuales del producto valida campos (cantidad, fecha de compra) antes del envío mediante PUT /api/despensa con manejo de errores específicos. El éxito muestra confirmación y recarga los datos y el error notifica al usuario manteniendo los datos ingresados. Integra cálculos nutricionales basados en las cantidades actualizadas.

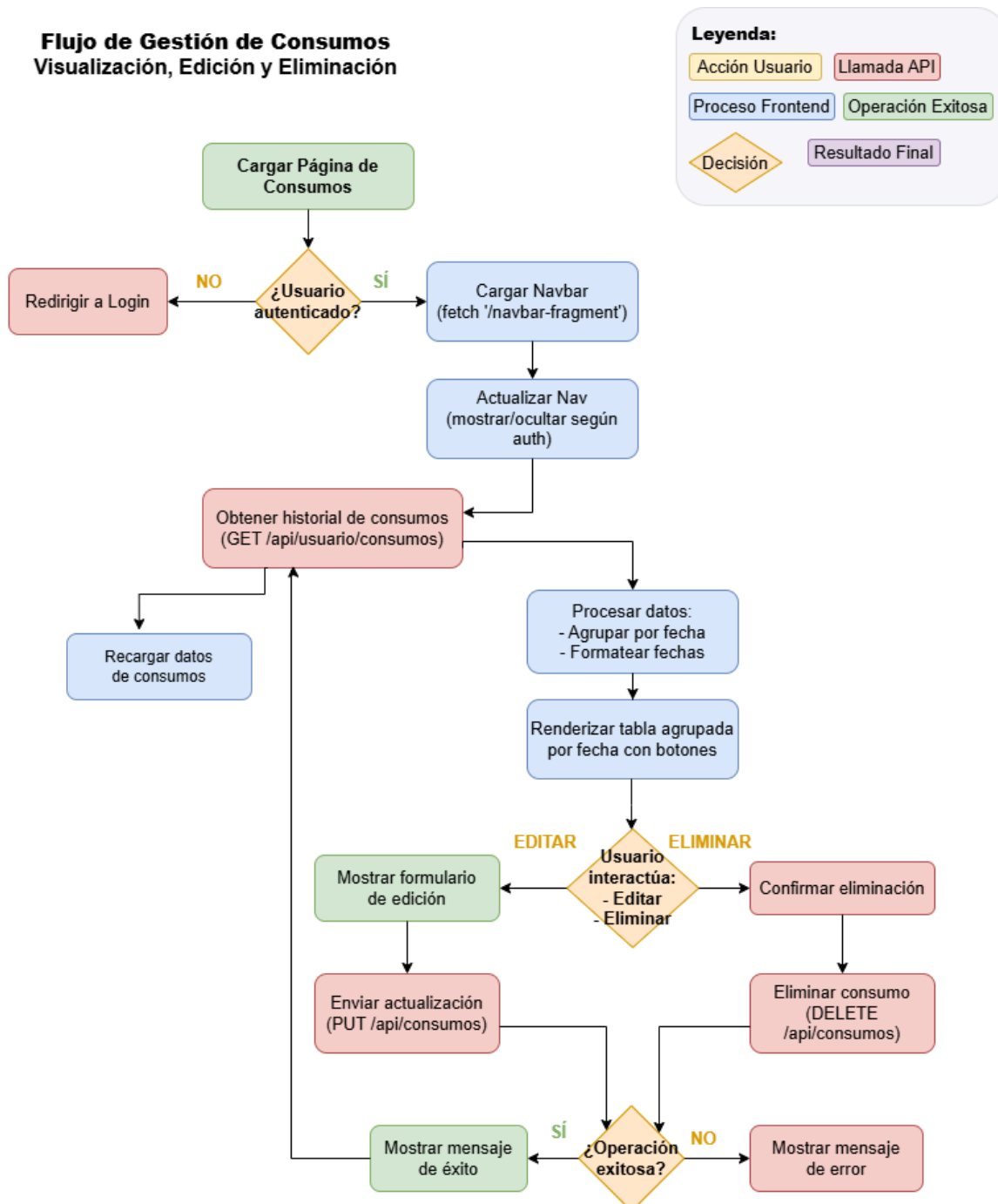
Flujo de Gestión de Despensa Visualización y Actualización de Productos



Consumos

Carga la página y verifica autenticación (JWT). Si no está autenticado redirige a login. Obtiene el historial de consumos (GET /api/usuario/consumos), procesa y agrupa datos por fecha. Renderiza tabla interactiva con opciones de edición/eliminación que se implementarán en próxima versión.

Flujo de Gestión de Consumos Visualización, Edición y Eliminación



Codificación

Entorno de Desarrollo y Tecnologías del Proyecto

Frontend Dinámico con Thymeleaf → Usa **HTML5**, **CSS3** (**Bootstrap**) y **JavaScript** para una interfaz responsive para renderizado dinámico en el servidor, con llamadas asíncronas a APIs mediante **Fetch API**.

Backend en Java con Spring → Desarrollado en **Spring Boot** para gestión eficiente de peticiones, seguridad con **JWT** y conexión a **MySQL** mediante **JPA/Hibernate**. Persistencia con **Spring Data JPA**.

API REST Propia → Endpoints bajo **/api** con autenticación **JWT**, validación de datos y respuestas estructuradas en **JSON** (éxito/error estandarizados).

Integración con API Externa (Open Food Facts) → Consume servicios externos mediante **fetch** para obtener datos nutricionales, procesando respuestas **JSON** y manejando errores con retroalimentación clara al usuario procesando la respuestas **JSON** y mostrando información detallada.

IDE: Visual Studio Code (VSC)

JWT para sesiones seguras sin estado.

MySQL Workbench para diseño/modelado de la base de datos.

Postman para probar endpoints del API REST.

Tecnologías Clave: Thymeleaf (estructura base) + JavaScript (interactividad) + REST API (comunicación con backend) + JSON para transferencia de datos.

Aspectos relevantes de la implementación

La autenticación **JWT** se integra mediante un flujo seguro: desde la generación de tokens firmados en el backend (con **JwtTokenProvider**) hasta su validación en cada petición mediante el filtro **JwtAuthenticationFilter**, garantizando acceso stateless y protección contra manipulaciones."

El sistema emplea **JWT** para gestionar sesiones sin estado, delegando la seguridad a tokens firmados con clave secreta (**HS512**), lo que elimina la necesidad de almacenar sesiones en el servidor y optimiza la escalabilidad."

La arquitectura combina **Spring Security** con **JWT**: **UserPrincipalAdapter** adapta la entidad usuario a un formato compatible con Spring, mientras que **JwtAuthEntryPoint** centraliza el manejo de errores (como tokens expirados o inválidos), devolviendo respuestas estandarizadas en **JSON**."

Para asegurar endpoints REST, el **SecurityConfig** deshabilita **CSRF** y configura rutas protegidas, donde sólo solicitudes con **JWT** válido en el header **Authorization** son procesadas, integrando además roles y permisos mediante claims del token.

Estas frases resaltan la cohesión entre tecnologías (**Spring Security + JWT**), la seguridad (firma, stateless) y la experiencia del usuario (errores claros). Puedes usarlas en tu sección de Codificación o Análisis y Diseño.

Manual de usuario.

Manual de SmartDieT

1. Página de Inicio

Objetivo: Presentación de la aplicación.

Pasos:

- Explora las secciones principales.
- Haz clic en "**Entrar**" para acceder al login.

Nutrición inteligente, decisiones saludables

Productos frescos Productos API [→ Entrar](#)

Nutrición inteligente, decide y avanza

Prestar atención a tu cuerpo, reducir el consumo ineficiente y mejorar tu rendimiento

[Entrar](#)

Smart DieT
IoT Mat Machine Learning System

[Discord](#) [LinkedIn](#) [GitHub](#) [YouTube](#) [Instagram](#)

Opciones de uso

Uso ágil

Consulta datos de productos

Aprovéchalo

Registra tu despensa y tus consumos.

Recibe feedback

Interpretación con insights o puntos fuertes.

Datos alimenticios proporcionados por OpenFoodFacts

Esta aplicación utiliza la API de OpenFoodFacts, una base de datos abierta y colaborativa de información nutricional.

Agradecemos a esta increíble comunidad por hacer posible acceder a datos alimenticios de manera libre y transparente.

[Visitar OpenFoodFacts](#)

Lo que dicen nuestros usuarios



Carlos Rodríguez

"La aplicación me encanta para registrar rápidamente lo que como, sin complicaciones. Me gustaría que la búsqueda de productos fuera más intuitiva, pero en general cumple con lo que necesito."



Ana Martínez

"La función de recomendaciones según mis metas de proteínas y calorías es increíblemente útil. Sin embargo, a veces tarda en cargar los datos de la API externa, lo que ralentiza mi rutina"



David Fernández

"Me ayuda a entender mejor lo que como y a organizar mi despensa sin abrumarme. Aunque al principio me costó entender algunas opciones, ahora la uso a diario"

Planes y Precios

Básico

Gratis

- ✓ Consulta productos
- ✓ Despensa
- ✓ Soporte por correo

Empezar

Pro

49€/año

- ✓ Consulta productos
- ✓ Despensa
- ✓ Consumos
- ✓ Soporte por correo
- ✓ Seguridad
- ✓ Insights

Pack duo

99€/año

- ✓ Consulta productos
- ✓ Despensa
- ✓ Consumos
- ✓ Soporte por correo
- ✓ Seguridad
- ✓ Insights

¿Listo para impulsarte?

Regístrate y prueba tu cuenta gratuita.

→ Vamos allá

Contáctanos

¿Tienes preguntas?

Nuestro equipo está aquí para ayudarte.

✉ Escribemos

soporte@smartdietnutricioninteligente.com

Nombre

Correo Electrónico

Asunto

Mensaje

Enviar Mensaje

Smart diet, nutrición inteligente

La solución que te ahorra tiempo y esfuerzo.

Mejora tu calidad de vida.



2. Inicio de sesión

Objetivo: Acceder a tu cuenta.

Pasos:

- Ingresa tu **email** y **contraseña**.
- Haz clic en "**Iniciar Sesión**".

Iniciar sesión

Usuario

eetxab@gmail.com

Contraseña

.....

Iniciar sesión

Registrarse

3. Registro de Usuario

Objetivo: Crear una cuenta nueva.

Pasos:

- Completa todos los campos.
- Asegúrate de que las contraseñas coincidan.
- Haz clic en "**Registrarse**".

Nutrición inteligente, decisiones saludables

Productos frescosProductos API

Entrar

Registro de Usuario

Nombre	Correo electrónico	Edad
Estibalz	eetxab@gmail.com	35
Contraseña	Confirmar contraseña	Peso (kg)
*****	*****	50
Altura (cm)	Nivel de actividad	Meta de salud
160	Ligera	Mejorar salud general
Tipo de dieta	Tipo de usuario:	Sexo
Omnívora	Basic	Femenino


Registrarse

4. Perfil de Usuario

Objetivo: Ver y gestionar tu información.

Pasos:

- Visualiza tus datos personales.
- Navega a editar perfil con el botón "**Editar Perfil**".
- Navega a otras secciones usando la barra superior.



Estibaliz
Cuenta Basic

Información Personal

Correo electrónico: eetxab@gmail.com

Edad: 35 años

Medidas Corporales

Peso: 80 kg

Altura: 160 m

Preferencias de Salud

Nivel de actividad: Sedentaria

Meta de salud: Perder peso

Tipo de dieta: Omnívora

Recomendación Nutricional

Calorías recomendadas: 1837.78 kcal

Proteínas: 551.334%

Grasas: 459.445%

Carbohidratos: 827.001%

Editar Perfil

Nutrición inteligente, decisiones saludables

Productos frescos | Productos API | Perfil | Despensa | Consumo

Salir

© 2025 Nutrición inteligente, decisiones saludables. Todos los derechos reservados.

5. Editar Perfil

Objetivo: Actualizar datos personales.

Pasos:

- Modifica los campos.
- Haz clic en "**Actualizar Perfil**" para guardar cambios.

Nutrición inteligente, decisiones saludables

Productos frescosProductos APIPerfilDespensaConsumoSalir


Editar

Información Personal

Nombre:

Correo electrónico:

Edad:

Estibaliz

eetxab@gmail.com

35

Medidas Corporales

Peso:

Altura:

80

160

Preferencias de Salud

Nivel de actividad:

Meta de salud:

Tipo de dieta:

Sedentaria

Perder peso

Omnívora

Actualizar Perfil

6. Productos (Local)

Objetivo: Gestionar productos.

Pasos:

- Busca productos y consulta sus datos.
- Si selecciona producto sin estar logueado alertará de que necesita entrar antes.
- Selecciona producto para añadir datos a despensa o consumos.
- Usa la barra superior para navegar a otras secciones.

tomate

Buscar

Formulario del Producto

Nombre:

Tomate

Energía:

18.0

Grasas (g):

0.2

Carbohidratos (g):

3.9

Proteínas (g):

0.9

Azúcares (g):

2.6

Fibra (g):

1.2

Tomate

Grasas: 0.2 g

Carbohidratos: 3.9 g

Proteínas: 0.9 g

Azúcares: 2.6 g

Fibra: 1.2 g

Vitamina C: 13.7 mg

Potasio: 237.0 mg

Calcio: 18.0 mg

Magnesio: 11.0 mg

Hierro: 0.5 mg

Seleccionar

Sopa de tomate

Grasas: 2.0 g

Carbohidratos: 14.0 g

Proteínas: 2.0 g

Azúcares: 5.0 g

Fibra: 1.5 g

Vitamina C: 19.0 mg

Potasio: 392.0 mg

Calcio: 28.0 mg

Magnesio: 15.0 mg

Hierro: 1.3 mg

Seleccionar

sauce tomate cuisinée

Grasas: 0.4 g

Carbohidratos: 7.6 g

Proteínas: 1.6 g

Azúcares: 0 g

Fibra: 0 g

Vitamina C: 0 mg

Potasio: 0 mg

Calcio: 0 mg

Magnesio: 0 mg

Hierro: 0 mg

Seleccionar

Vitamina C (mg):

13.7

Potasio (mg):

237.0

Calcio (mg):

18.0

Magnesio (mg):

11.0

Hierro (mg):

0.5

Cantidad:

1

Fecha de compra:

23/05/2025 14:10

Despensa

Consumo

7. Productos (API Open Food Facts)

Objetivo: Buscar en la base de datos de OpenFoodFacts.


Pasos:

- Busca productos y consulta sus datos.
- Si selecciona producto sin estar logueado alertará de que necesita entrar antes.
- Selecciona producto para añadir datos a despensa o consumos.
- Usa la barra superior para navegar a otras secciones.

Nutrición inteligente, decisiones saludables

Productos frescosProductos APIPerfilDespensaConsumo

Salir



tomate

Buscar

Formulario del Producto

Nombre:

Energía:

Grasas (g):

Carbohidratos (g):

Proteínas (g):

Azúcares (g):

Fibra (g):


Cantidad:

Fecha de compra:

Despensa

Consumo

SAUCE TOMATE CUISINÉE



Marca: solis

100g: 68kcal

Grasas: 0.4g

Carbohidratos: 7.6g


Proteínas: 1.6g

Azúcares: 4.4g

Fibra: 2g

Seleccionar

TOMATO KETCHUP



Marca: Heinz

100g: 102kcal

Grasas: 0.1g

Carbohidratos: 23.2g


Proteínas: 1.2g

Azúcares: 22.8g

Fibra:

Seleccionar

CONCENTRÉ DE TOMATES AICHA



Marca: Aïcha

100g: 31.6kcal

Grasas: 0.108g

Carbohidratos: 6.41g


Proteínas: 1.24g

Azúcares: 2.43g

Fibra: 0.811g

Seleccionar

DELICIA DOUBLE CONCENTRE DE...



Marca: Delicia

100g: 91kcal

Grasas: 0.09g

Carbohidratos: 18.35g

Proteínas: 4.13g

Azúcares:

Fibra:

Seleccionar


46

8. Despensa Personal

Objetivo: Gestionar tus alimentos almacenados.

Pasos:

- Revisa la lista de productos.
- Usa **"Editar"** para cambiar cantidad y fecha.
- **"Guardar edición"** para guardar la edición del registro.

 **Nutrición inteligente, decisiones saludables**

[Productos frescos](#) [Productos API](#) [Perfil](#) [Despensa](#) [Consumo](#) [Salir](#)


Lista de Productos en la Despensa

Nombre Producto	Cantidad	Fecha de Compra	
sauce tomate cuisinée	1	2025-05-23 14:16	Editar
Tomate	3	2025-05-23 14:13	Editar

Actualizar Producto

Nombre del Producto

Cantidad

Fecha de Compra
 

[Guardar edición](#)


© 2025 Nutrición inteligente, decisiones saludables. Todos los derechos reservados.

9. Registro de Consumos


Objetivo: Llevar un historial de consumos o ingestas.

Pasos:

- Consultar lista de consumos
- Editar y eliminar se implementará en la siguiente versión.

 **Nutrición inteligente, decisiones saludables**

Productos frescos Productos API Perfil Despensa Consumo

 Salir

Historial de Consumos

Producto	Cantidad	Fecha de Consumo	Calorías	Acciones
2025-05-21				
Tomate	1	2025-05-21 19:17	0.18	<button>Editar</button> <button>Eliminar</button>
2025-05-23				
Tomate	1	2025-05-23 14:10	0.18	<button>Editar</button> <button>Eliminar</button>
Total Calorías:			0.36	

Requisitos e instalación

Aplicación web desarrollada con Spring Boot 3.4.4 implementando principios de arquitectura hexagonal en el backend para separación de responsabilidades y patrón MVC en el frontend. Utiliza autenticación JWT para la gestión segura de datos.

Entorno de Desarrollo

Visual Studio Code

MySQL para gestión de la base de datos.

Spring Boot 3.4.4: Framework principal de la aplicación.

Java 17+ (JDK): Requerido por Spring Boot 3

Maven 3.6+: Para gestión de dependencias.

MySQL 8.0+: Base de datos (configurada en `application.properties`).

Configuración de Base de Datos

Crear una base de datos llamada **nutricion** en **MySQL**.

Cargar el script preparado con la base de datos y asegurarse de que está bien configurada en `application.properties`, tanto la ruta como los permisos.

Variables de Entorno y Seguridad

JWT Secret: Usar clave segura (mínimo 64 caracteres).

Credenciales de Spring Security: Usuario temporal (o implementar autenticación personalizada).

Ejecución del Proyecto

Desde el origen del proyecto en terminal: `mvn spring-boot:run`

Acceder: `http://localhost:8080` `http://localhost:8080` (puerto defecto).

Pruebas

Postman para probar endpoints del API REST

Resumen de Tecnologías Clave

Área	Tecnologías
Frontend	Thymeleaf, HTML5, CSS3 (Bootstrap 5), JavaScript (Fetch API)
Backend	Spring Boot 3, Spring Security (JWT), Spring Data JPA (MySQL), Validación Jakarta
APIs	API REST propia (/api/**), Open Food Facts (externa)
Herramientas	VSC, Maven, MySQL Workbench, Postman

Conclusiones

Conclusiones sobre el trabajo realizado

El desarrollo de esta aplicación web para la gestión nutricional ha permitido crear una herramienta integral que combina una base de datos interna con una API externa (Open Food Facts), ofreciendo a los usuarios una forma eficiente de consultar información detallada de productos y registrar su consumo diario. La implementación de una arquitectura hexagonal ha facilitado la modularidad y el mantenimiento del código, mientras que el uso de Spring Security con JWT ha asegurado un sistema de autenticación robusto y sin estado (stateless), protegiendo los datos de los usuarios.

La aplicación cumple con su objetivo principal: brindar una solución accesible y segura para quienes buscan mejorar sus hábitos alimenticios, permitiendo el registro de despensas personales y el seguimiento nutricional. La integración de tecnologías como Thymeleaf (frontend dinámico), Spring Boot (backend) y MySQL (persistencia) ha resultado en un sistema funcional, escalable y preparado para futuras ampliaciones.

Posibles ampliaciones y mejoras

Aunque el proyecto cumple con los requisitos iniciales, se identifican áreas de mejora y expansión:

1. Ampliación de roles de usuario: Introducir perfiles avanzados (nutricionistas, entrenadores) con acceso a estadísticas y recomendaciones personalizadas basadas en metas de salud.
2. Análisis de patrones de consumo: Implementar algoritmos que analizan los datos registrados para sugerir ajustes en la dieta según objetivos (ej. pérdida de peso, control de alergias).
3. Versión móvil (Android): Desarrollar una app nativa para mejorar la accesibilidad y permitir un uso más flexible fuera del entorno web.
4. Recetas personalizadas: Permitir a los usuarios guardar combinaciones de alimentos frecuentes y calcular su aporte nutricional automáticamente.

En resumen, este proyecto sienta las bases para una plataforma en crecimiento, con un enfoque claro en la usabilidad, seguridad y escalabilidad, preparada para adaptarse a las necesidades emergentes en el ámbito de la nutrición y salud digital.

Bibliografía

A continuación, se detallan las fuentes de información y referencias utilizadas en el desarrollo del proyecto:

1. Open Food Facts

Open Food Facts. (s. f.). API Documentation.

Recuperado de <https://world.openfoodfacts.org/data>

Fuente principal para la obtención de datos nutricionales de productos alimenticios. Se utilizó su API pública para integrar información detallada en la aplicación.

2. hdeleon.net

León, H. (s. f.). Blog personal y recursos de programación.

Recuperado de <https://hdeleon.net>

Referencia para conceptos avanzados en diseño y arquitectura software..

3. ChatGPT (OpenAI)

OpenAI. (2023). ChatGPT [Modelo de lenguaje].

Recuperado de <https://chat.openai.com>

Se empleó como herramienta de apoyo en la generación de ideas, estructuración de código y resolución de dudas técnicas durante el desarrollo.

4. DeepSeek

DeepSeek. (2024). Documentación y asistencia en desarrollo de software.

Plataforma utilizada para consultar dudas técnicas. y asistir en la redacción.

5. Claude (Anthropic)

Anthropic. (2023). Claude AI [Modelo de lenguaje].

Se ha utilizado como ayuda en el desarrollo de xml para crear diagramas y otras dudas técnicas.