



Instituto Politécnico Nacional
Escuela Superior de Cómputo



Proyecto final

Sistema de asistencia nutricional

Materia: Ingeniería De Software

Profesor: Gabriel Hurtado Avilés

Grupo: 6CV3

Integrantes: -Almogabar Nolasco Jaime Brayan | 2022630476

-Díaz Hernández Braulio | 2022630489

-García Quiroz Gustavo Ivan | 2022630278

- Morales Torres Alejandro | 2021630480

-Rodriguez Rivera Claudia Patricia | 2022630334

-Tellez Partida Mario lahveh | 2022630535

Índice

1	Introducción	1
1.1	Alcance del Sistema	1
2	Descripción del Sistema.....	2
3	Objetivos del Sistema.....	2
3.1	Objetivos específicos	2
4	Características Clave.....	3
4.1	Definiciones y Acrónimos.....	4
4.2	Referencias Técnicas	4
4.2.1	Documentación Oficial	4
4.2.2	Estándares y Normativas	4
5	Arquitectura del Sistema.....	5
5.1	Visión general	5
	Arquitectura del Sistema.....	5
	Backend	5
	Frontend.....	5
	Base de Datos.....	6
	Infraestructura	6
6	Especificaciones Técnicas.....	7
6.1	Backend	7
6.2	Frontend.....	7
6.3	Base de Datos	7
6.3.1	Infraestructura.....	8
6.3.2	Dependencias Principales	8
6.3.3	Integraciones y Herramientas	8
6.4	Tecnologías Utilizadas	9
6.5	Requisitos del Sistema	9

6.6	Entorno de Desarrollo	9
6.6.1	Instalación del Sistema Operativo y Prerequisitos	9
6.6.2	Instalación de Windows 10/11	9
6.6.3	Instalación de Java Development Kit (JDK) 21	11
6.6.4	Instalación de Maven	12
6.6.5	Instalación de Neo4j	13
6.6.6	Instalación de Visual Studio Code	15
7	Desarrollo	18
7.1	Documento de Requerimientos.....	18
7.1.1	Requerimientos Funcionales	18
7.1.2	Requerimientos No Funcionales.....	19
7.2	Funcionalidades F (Marco FURPS)	21
7.2.1	Funcionalidades (Functionality)	21
7.2.2	Usabilidad (Usability)	22
7.2.3	Confiabilidad (Reliability)	23
7.2.4	Rendimiento (Performance).....	23
7.2.5	Soporte (Support)	23
7.2.6	Ontología o Base de Datos Orientada a Grafos para Mejorar el Sistema 24	
7.2.7	Uso de Ontología.....	24
7.2.8	Base de Datos Orientada a Grafos.....	24
7.2.9	UI/UX Específico para la Aplicación	24
7.3	Diagrama de Casos de Uso	26
7.3.1	Registro de Usuarios	26
7.3.2	Gestión de Pacientes.....	27
7.3.3	Generación de Planes de Dieta	28
7.3.4	Seguimiento y Monitoreo	29

7.3.5	Sistema de Interacción	30
7.4	Prototipos de Interfaces Gráficas.....	33
7.4.1	Prototipo de Interfaz –Login de usuario	33
7.4.2	Prototipo de Interfaz –Registro de usuario.....	33
7.4.3	Prototipo de Interfaz – Dashboard de paciente.....	35
7.4.4	Prototipo de Interfaz – Dashboard de nutricionalista	35
7.4.5	Prototipo de Interfaz - Historial de citas	36
7.4.6	Prototipo de Interfaz - Lista de Planes Nutricionales	37
7.4.7	Prototipo de Interfaz - Chat.....	38
7.4.8	Prototipo de Interfaz – Historial de tickets.....	39
7.4.9	Prototipo de Interfaz - Perfil de Usuario	40
8	Pruebas de API REST con Postman/Retrofit	42
8.1	Pacientes	42
8.2	Consultas:	42
8.3	Tickets	43
9	Conclusiones.....	44
10	BIBLIOGRAFÍA APA.....	45

1 Introducción

Este manual técnico tiene como objetivo proporcionar una guía detallada sobre la arquitectura, implementación y mantenimiento del Sistema de Asistencia Nutricional. El documento está dirigido a desarrolladores, administradores de sistemas y personal técnico involucrado en la implementación, mantenimiento y soporte del sistema.

1.1 Alcance del Sistema

El Sistema de Asistencia Nutricional es una aplicación web desarrollada con Spring Boot que permite la gestión integral de pacientes y sus planes nutricionales. El sistema abarca las siguientes funcionalidades principales:

Gestión de Usuarios

- Registro y autenticación de nutricionistas y pacientes
- Gestión de perfiles y roles
- Control de acceso basado en roles (RBAC)

Gestión de Pacientes

- Registro del historial nutricional y médico
- Seguimiento de medidas antropométricas
- Registro de preferencias alimenticias y restricciones

Planes Nutricionales

- Creación de planes personalizados
- Cálculo de requerimientos calóricos
- Seguimiento de progreso
- Ajuste dinámico de planes

Sistema de Comunicación

- Chat en tiempo real entre nutricionista y paciente
- Sistema de tickets para consultas
- Notificaciones automáticas

2 Descripción del Sistema

El Sistema de Asistencia Nutricional es una solución moderna diseñada para facilitar la gestión de servicios nutricionales mediante un enfoque basado en microservicios. Utilizando tecnologías de vanguardia, el sistema permite una experiencia de usuario eficiente y un entorno seguro, escalable y fácil de mantener. A través de esta plataforma, se busca automatizar y optimizar los procesos asociados a la gestión de alimentos y asistencia nutricional, brindando una herramienta poderosa tanto para usuarios finales como para administradores.

3 Objetivos del Sistema

El objetivo principal del Sistema de Asistencia Nutricional es mejorar la eficiencia y efectividad de los servicios nutricionales mediante la automatización de tareas y la optimización de procesos.

3.1 Objetivos específicos

Algunos de los objetivos específicos incluyen:

- Facilitar el registro y la gestión de pacientes y nutricionistas.
- Proporcionar herramientas para la creación y seguimiento de planes de dieta personalizados.
- Permitir la comunicación en tiempo real entre nutricionistas y pacientes.
- Proporcionar un sistema de notificaciones y recordatorios para mejorar la adherencia a los planes de dieta.
- Asegurar la seguridad y privacidad de los datos mediante la implementación de mecanismos de autenticación y autorización robustos.

4 Características Clave

1. **Registro de Usuarios:** Permitir el registro de nutricionistas y pacientes mediante un formulario detallado que incluye información personal, médica y preferencias alimenticias.
2. **Gestión de Pacientes:** Registrar y almacenar el historial nutricional y médico de los pacientes, con la capacidad de consultar y editar estos datos.
3. **Generación de Planes de Dieta:** Crear planes personalizados basados en la información médica y preferencias del paciente, así como en requerimientos calóricos y hábitos alimenticios.
4. **Seguimiento y Monitoreo:** Registrar la ingesta diaria de alimentos y actividad física del paciente, mostrando un historial de progreso.
5. **Notificaciones:** Enviar recordatorios automáticos sobre comidas, actividad física y citas.
6. **Interacción:** Facilitar la comunicación en tiempo real entre nutricionistas y pacientes mediante un chat integrado y un sistema de tickets para gestionar consultas y ajustes.

4.1 Definiciones y Acrónimos

JDK: Java Development Kit

IDE: Integrated Development Environment

API: Application Programming Interface

REST: Representational State Transfer

RBAC: Role-Based Access Control

JWT: JSON Web Token

MVC: Model-View-Controller

IMC: Índice de Masa Corporal

TMB: Tasa Metabólica Basal

GET: Gasto Energético Total

BDD: Base de Datos orientada a grafos (Neo4j)

4.2 Referencias Técnicas

4.2.1 Documentación Oficial

- Spring Boot Documentation (v3.4.1)
- Neo4j Documentation (v4.3.6)
- Java Documentation (v21)
- Maven Documentation

4.2.2 Estándares y Normativas

- NOM-004-SSA3-2012 (Expediente Clínico)
- ISO/IEC 25010:2011 (Calidad del Software)
- OWASP Security Standards

5 Arquitectura del Sistema

5.1 Visión general

El **Sistema de Asistencia Nutricional** es una solución moderna diseñada para facilitar la gestión de servicios nutricionales mediante un enfoque basado en **microservicios**. Utilizando tecnologías de vanguardia, el sistema permite una experiencia de usuario eficiente y un entorno seguro, escalable y fácil de mantener. A través de esta plataforma, se busca automatizar y optimizar los procesos asociados a la gestión de alimentos y asistencia nutricional, brindando una herramienta poderosa tanto para usuarios finales como para administradores.

Arquitectura del Sistema

El sistema está estructurado en una arquitectura de **microservicios**, que asegura flexibilidad y escalabilidad. A continuación se detallan los principales componentes y tecnologías utilizadas en la arquitectura:

Backend

El backend está construido sobre **Spring Boot** y **Java 21**, con un enfoque en el patrón de diseño **MVC** para la separación de responsabilidades. El sistema expone servicios a través de **APIs RESTful**, lo que permite una integración ágil con otros sistemas y con el frontend.

- **Seguridad:** Implementación de **JWT** para la autenticación y autorización, junto con un control de acceso basado en roles (RBAC).
- **Gestión de dependencias:** Se utiliza **Maven** como herramienta para la gestión de dependencias y construcción.

Frontend

El frontend es responsivo, desarrollado con **HTML**, **CSS**, y **JavaScript**, utilizando **Angular** o **React** dependiendo de la implementación. Se comunica con el backend a través de **APIs REST** y emplea **Axios** o **HttpClient** para el consumo de datos.

Base de Datos

El sistema utiliza **Neo4j**, una base de datos orientada a grafos, para modelar relaciones complejas entre entidades, y **PostgreSQL** para datos transaccionales y persistencia estructurada.

Infraestructura

El sistema está diseñado para ser desplegado en **contenedores Docker** tanto en entornos locales como en la nube. Se utiliza **NGINX** para balanceo de carga y **Kubernetes** para la orquestación y escalabilidad del sistema.

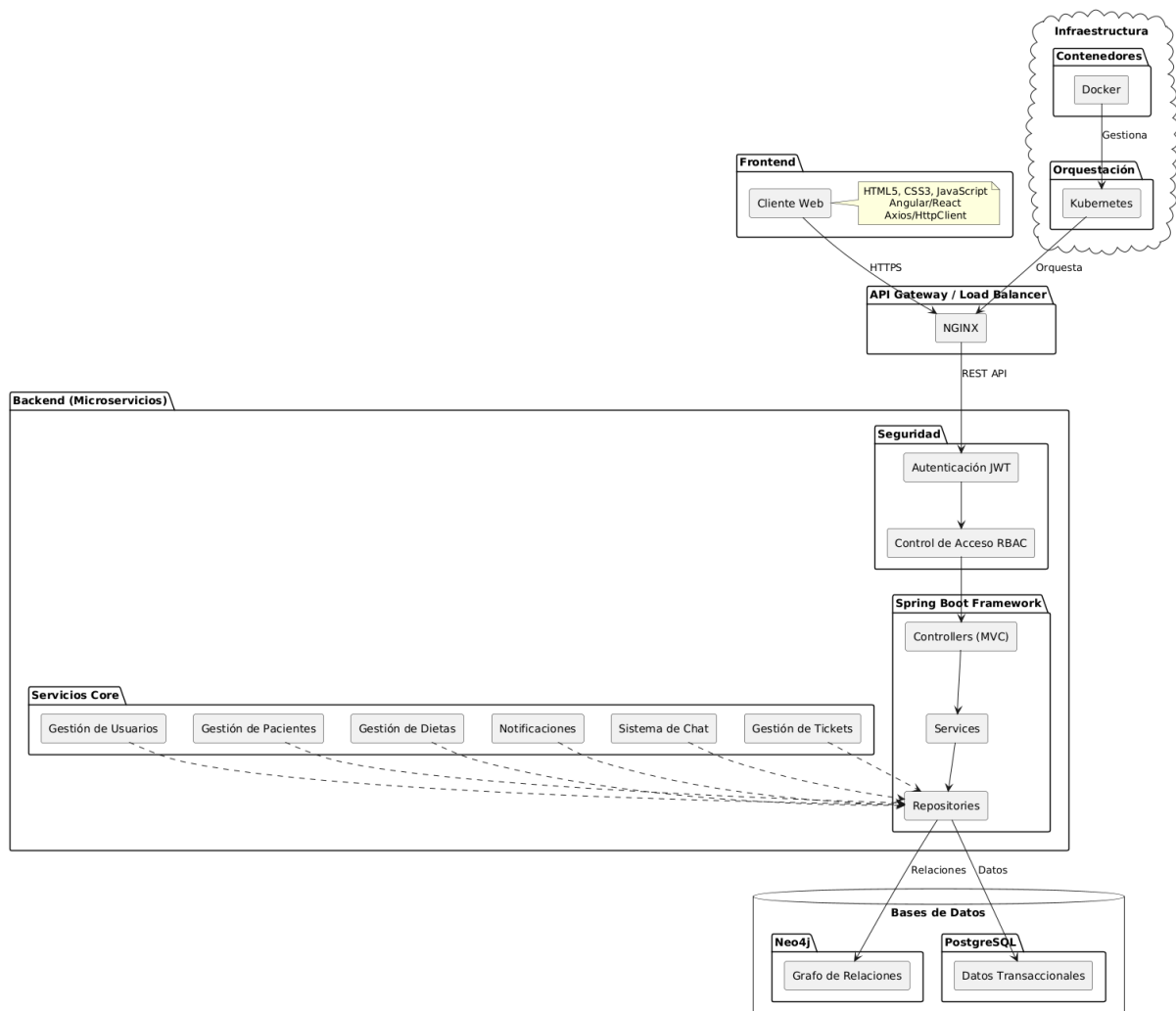


Figura 1 Arquitectura del Sistema

6 Especificaciones Técnicas

El **Sistema de Asistencia Nutricional** está desarrollado utilizando tecnologías modernas y un enfoque basado en microservicios. A continuación, se detallan las especificaciones técnicas clave:

6.1 Backend

Framework principal: Spring Boot (v3.4.1)

Lenguaje de programación: Java (JDK v21)

Arquitectura: Modelo **MVC** (Model-View-Controller)

Seguridad:

Autenticación y autorización con **JWT**.

Control de acceso basado en roles (**RBAC**).

Cumplimiento de estándares de seguridad **OWASP**.

Gestión de dependencias: Maven.

APIs:

Exposición de servicios RESTful para comunicación entre cliente y servidor.

Documentación de endpoints generada con **Swagger**.

6.2 Frontend

Tecnologías utilizadas: HTML, CSS, JavaScript.

Framework: Angular o React (según el caso de implementación).

Comunicación: Consumo de APIs REST utilizando **Axios** o **HttpClient**.

Estilo visual: Diseño responsivo con Bootstrap o Tailwind CSS.

6.3 Base de Datos

Sistema de gestión de bases de datos:

Neo4j (Base de Datos orientada a grafos) para modelar relaciones complejas entre entidades.

PostgreSQL para datos transaccionales y persistencia estructurada.

Conexión: Integración con Spring Data y consultas personalizadas.

Optimización: Uso de índices y queries optimizados.

6.3.1 Infraestructura

Servidor: Tomcat embebido en Spring Boot.

Plataforma de despliegue: Contenedores Docker para implementación en entornos locales y en la nube.

Balanceo de carga: Configurado con NGINX.

Escalabilidad: Orquestación con Kubernetes.

6.3.2 Dependencias Principales

Lombok: Para reducir código repetitivo.

Spring Security: Para políticas de seguridad avanzadas.

Spring Data Neo4j: Para integración con la base de datos de grafos.

Thymeleaf: (Opcional) Motor de plantillas para renderizar vistas en el servidor.

6.3.3 Integraciones y Herramientas

Sistema de Notificaciones: Integración con Firebase Cloud Messaging (FCM) para notificaciones en tiempo real.

Chat en tiempo real: Implementación con WebSocket.

Gestión de colas: RabbitMQ para procesamiento de tareas asincrónicas.

Estas especificaciones técnicas garantizan que el sistema sea robusto, seguro y escalable, cumpliendo con los estándares de calidad y normativas aplicables.

6.4 Tecnologías Utilizadas

- Java Version: 21
- Spring Boot Version: 3.4.1
- Neo4j Version: 4.3.6
- Maven (Build Tool)

6.5 Requisitos del Sistema

Hardware Recomendado

- Procesador: 2 núcleos o superior
- RAM: 4GB mínimo, 8GB recomendado
- Almacenamiento: 20GB disponibles
- Software Requerido
- JDK 21
- Maven 3.6+
- Neo4j 4.3.6
- IDE compatible con Spring Boot (recomendado: Visual Studio Code)
- Git para control de versiones

6.6 Entorno de Desarrollo

6.6.1 Instalación del Sistema Operativo y Prerequisitos

Requisitos Mínimos del Sistema

- Procesador: Intel Core i5 o superior
- RAM: 8GB mínimo
- Almacenamiento: 256GB SSD
- Sistema Operativo: Windows 10/11 Pro 64-bit

6.6.2 Instalación de Windows 10/11

1. Crear medio de instalación USB
 - Descargar Media Creation Tool de: <https://www.microsoft.com/software-download/windows10>

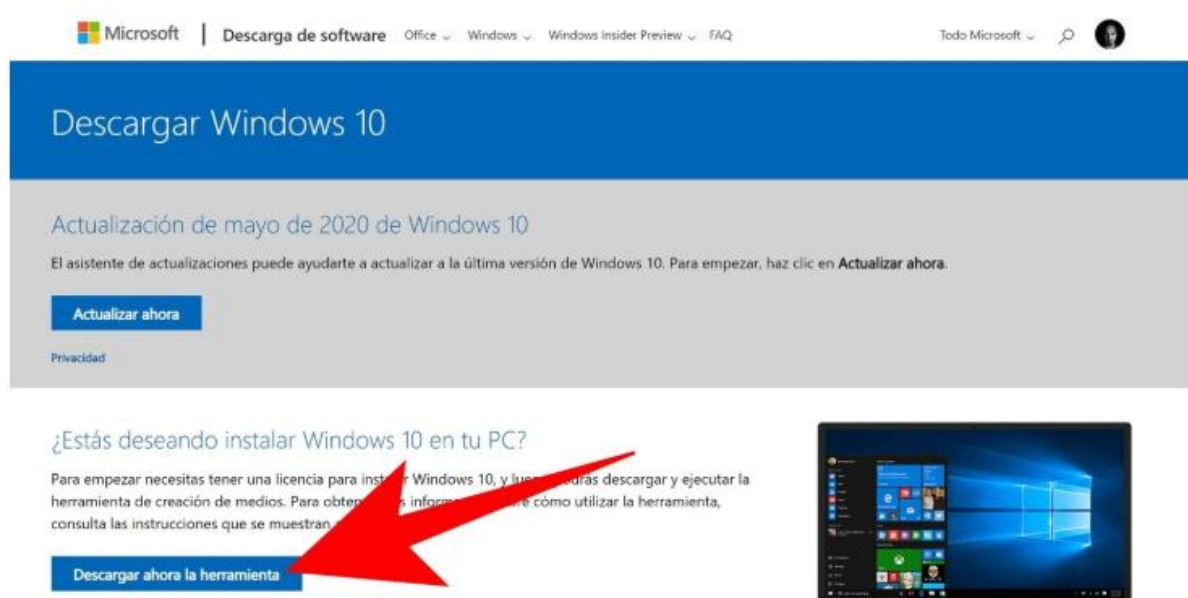


Figura 2 Crear medio de instalación USB.

- Ejecutar la herramienta y seleccionar "Crear medio de instalación"

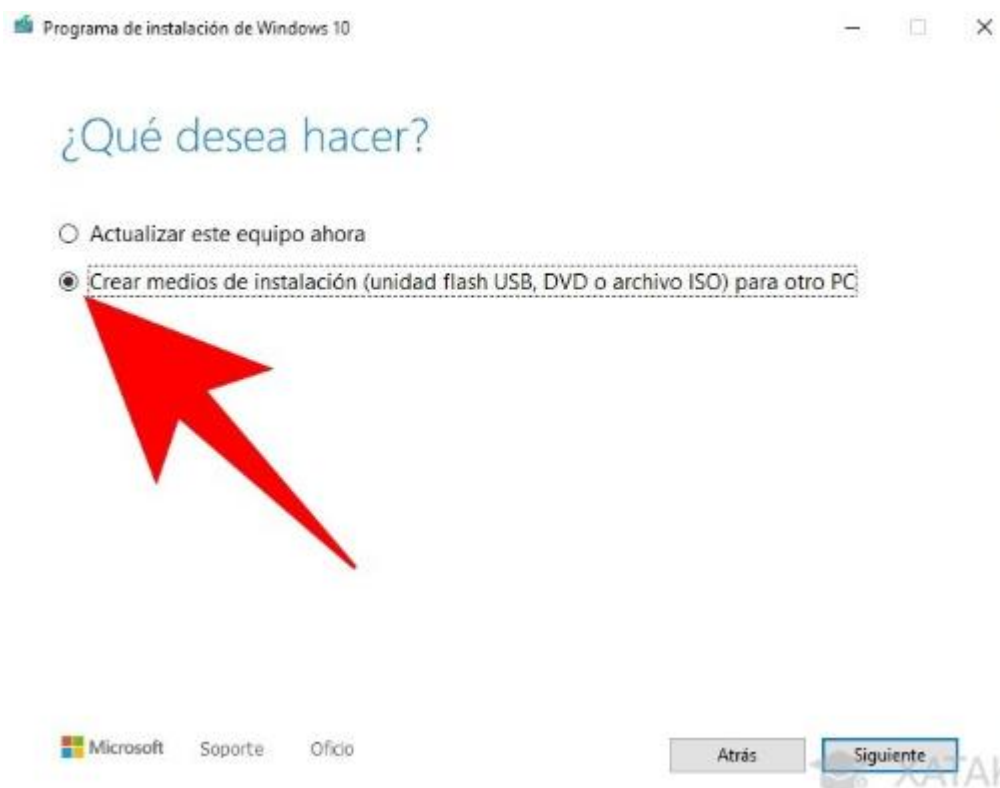


Figura 3 Crear medio de instalación

2. Proceso de instalación de Windows

- Iniciar desde USB
- Seleccionar idioma y región
- Seleccionar "Instalar ahora"
- Seguir las instrucciones.



Figura 4 Proceso de instalación de Windows

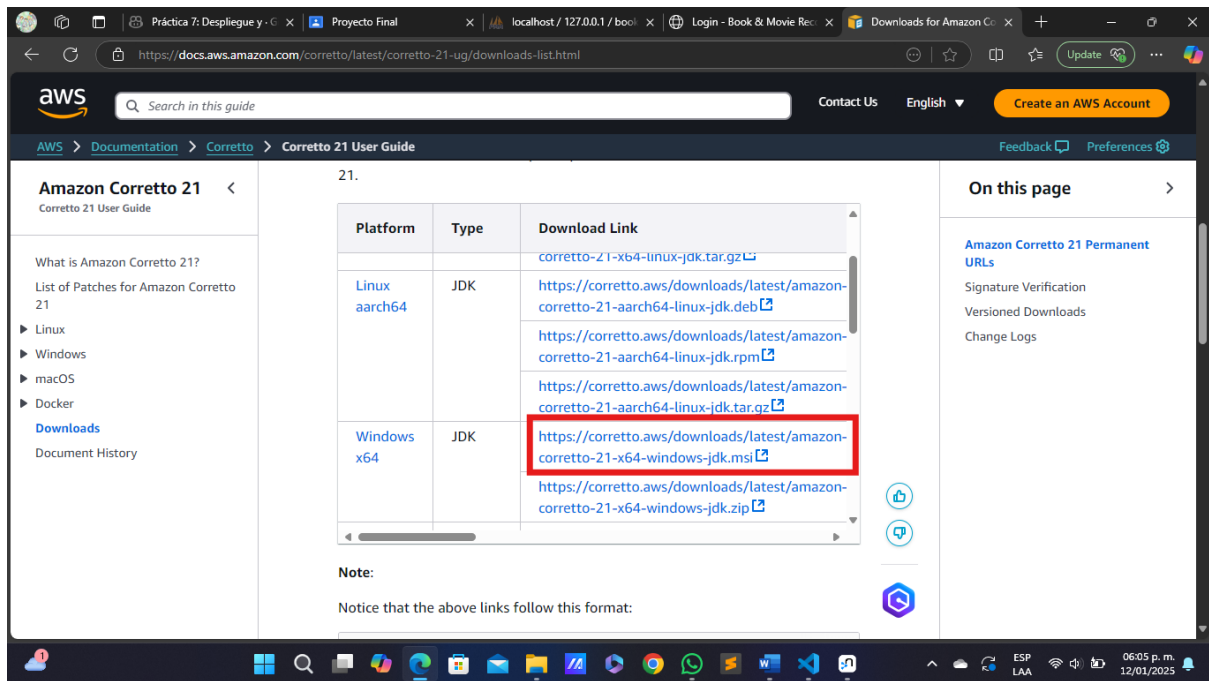
3. Configuración inicial

- Ingresar clave de producto
- Seleccionar Windows Pro
- Crear cuenta de administrador

6.6.3 Instalación de Java Development Kit (JDK) 21

1. Descargar JDK

- Ir a Amazon Corretto 21 en:
<https://docs.aws.amazon.com/corretto/latest/corretto-21-ug/downloads-list.html>
- Seleccionar JDK 21 para Windows x64



2. Instalar JDK

- Ejecutar el instalador
- Seguir el asistente de instalación
- Verificar instalación en terminal:

```
java -version
javac -version
```

Figura 5 Verificar instalación en terminal

6.6.4 Instalación de Maven

1. Descargar Maven

- Ir a Apache Maven: <https://maven.apache.org/download.cgi>
- Descargar Binary zip archive

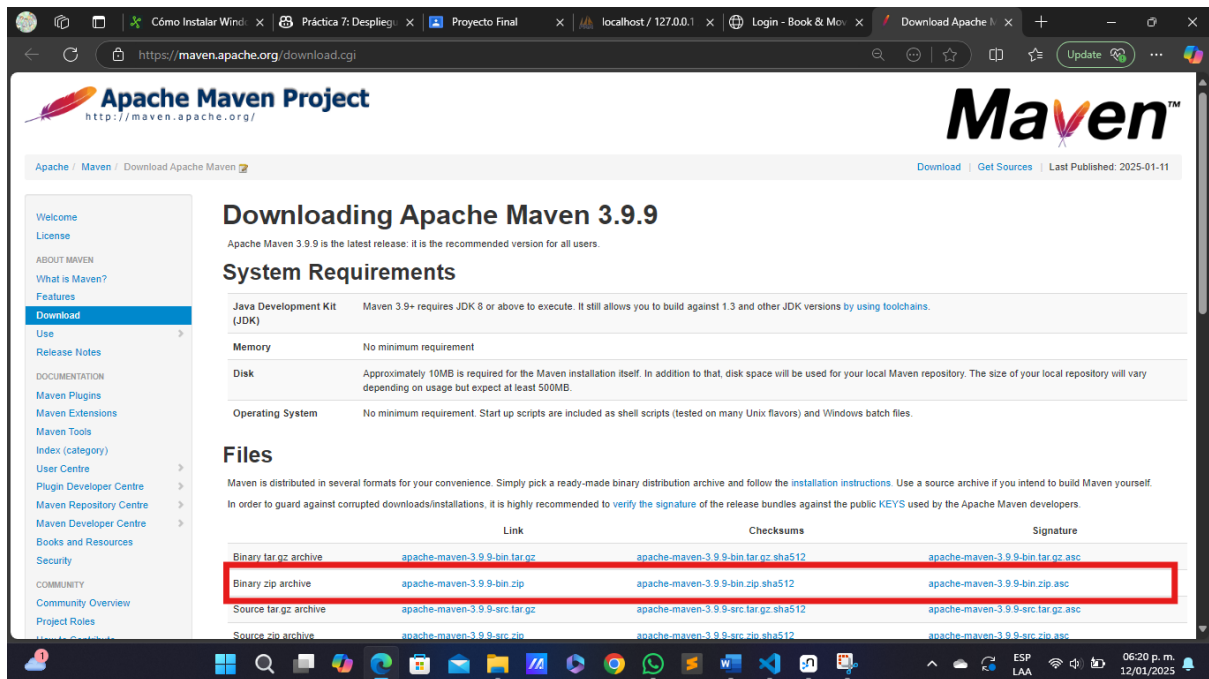


Figura 6 Instalación de Maven

2. Verificar instalación de Maven

```
mvn -version
```

Figura 7 Verificar instalación de Maven

6.6.5 Instalación de Neo4j

1. Descargar Neo4j Desktop

- Ir a [Neo4j Download](https://neo4j.com/download/): <https://neo4j.com/download/>
- Descargar Neo4j Desktop

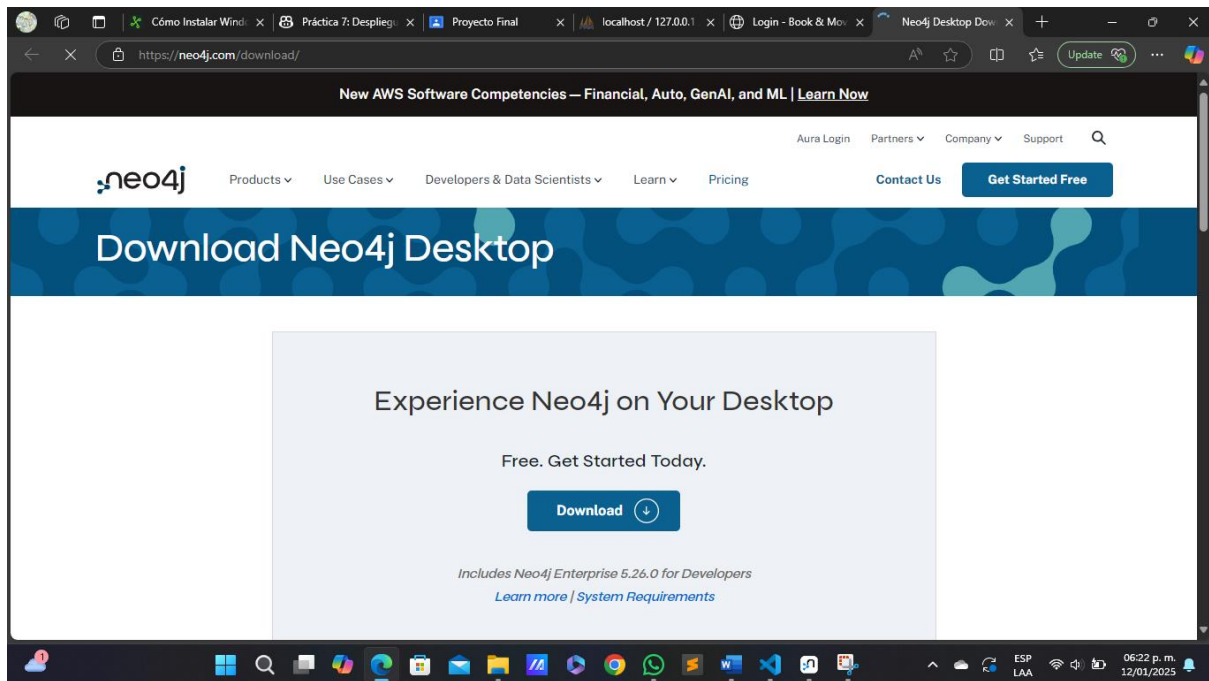


Figura 8 Instalación de Neo4j

2. Instalar Neo4j

- Ejecutar instalador
- Crear base de datos nueva
- Configurar usuario y contraseña

3. Verificar instalación

- Iniciar servidor Neo4j
- Acceder a <http://localhost:7474>
- Probar conexión

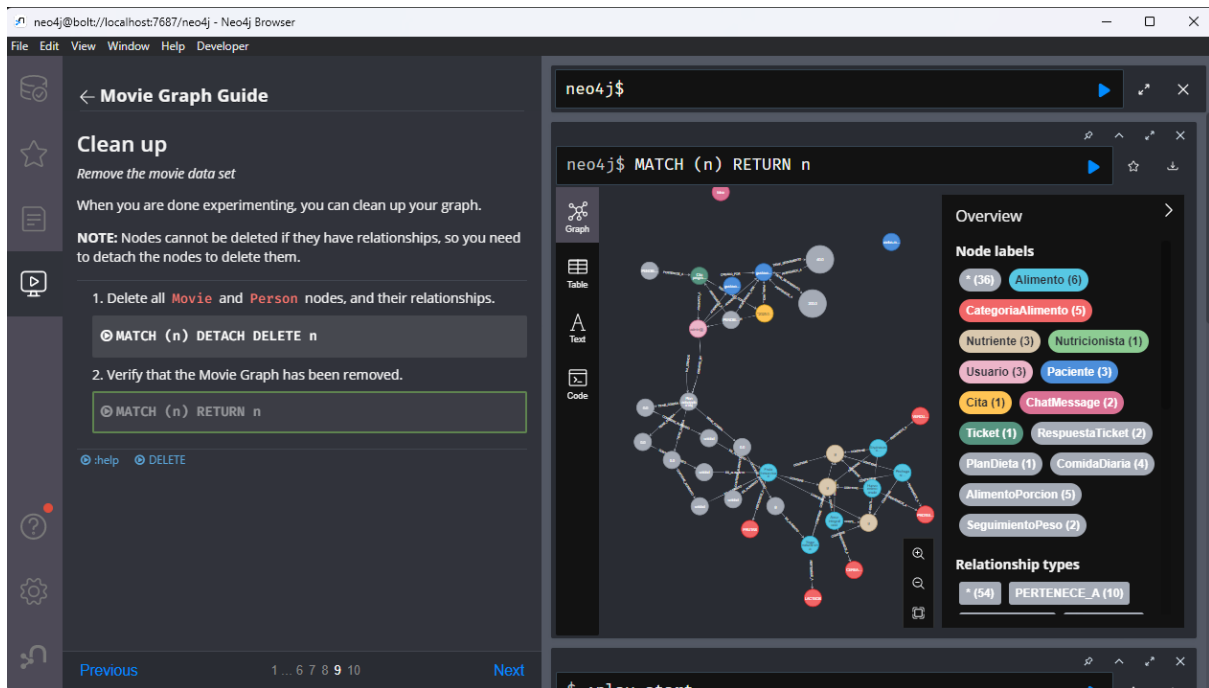


Figura 9 neo4j-browser.

6.6.6 Instalación de Visual Studio Code

1. Descargar Visual Studio Code

- Ir a [Visual Studio Code](https://code.visualstudio.com/): <https://code.visualstudio.com/>
- Descargar la versión para Windows x64

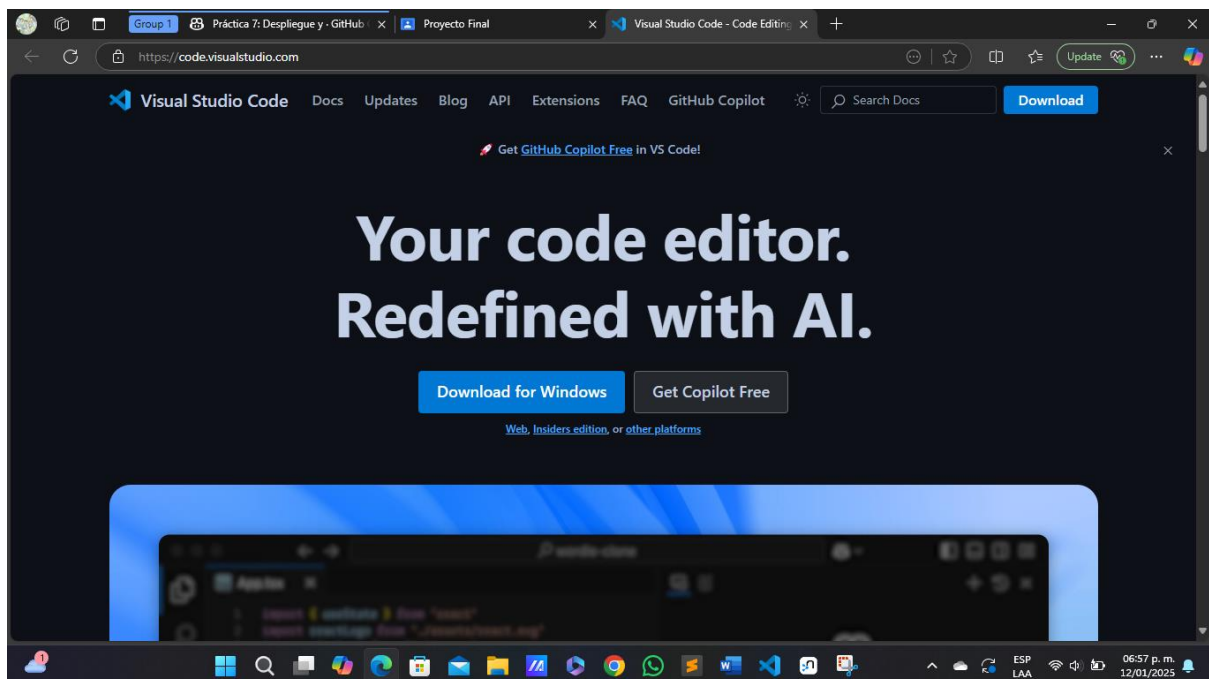


Figura 10 Descargar Visual Studio Code.

2. Instalar Visual Studio Code

- Ejecutar instalador
- Marcar opción "Add to PATH"
- Seleccionar "Create a desktop icon"

3. Instalar Extensiones Esenciales

- Extension Pack for Java
 - Incluye: Language Support for Java, Debugger for Java, Test Runner for Java
 - ID: vscjava.vscode-java-pack

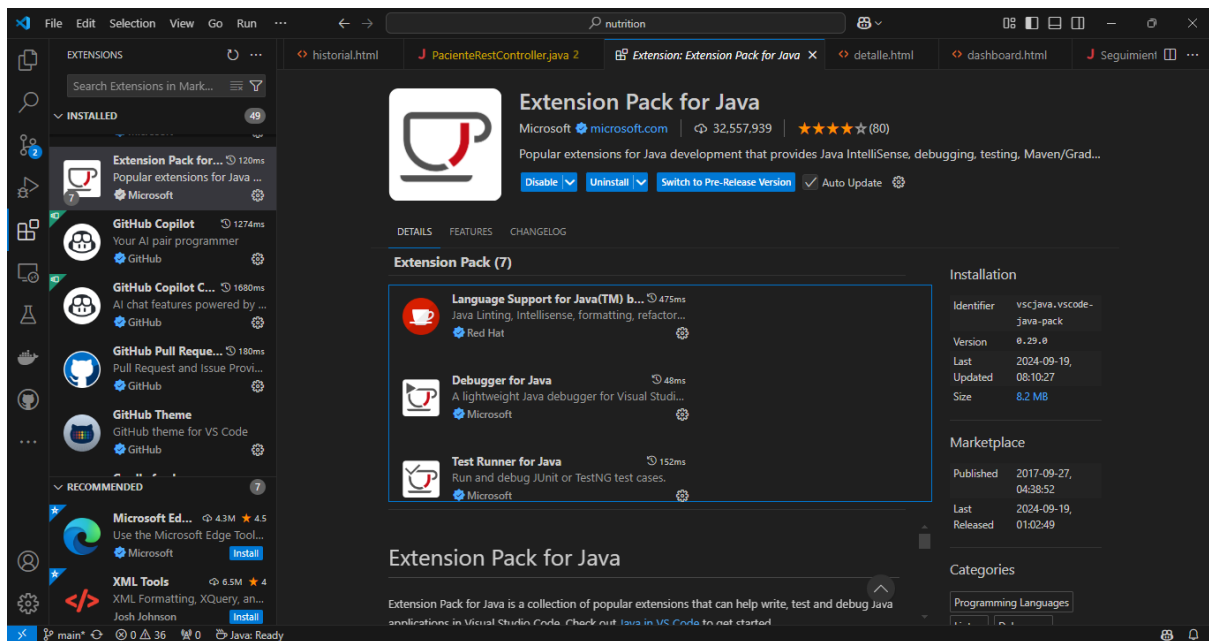


Figura 11 Extension Pack for Java.

- Spring Boot Extension Pack
 - Spring Boot Tools
 - Spring Initializr
 - Spring Boot Dashboard
 - ID: Pivotal.vscode-boot-dev-pack

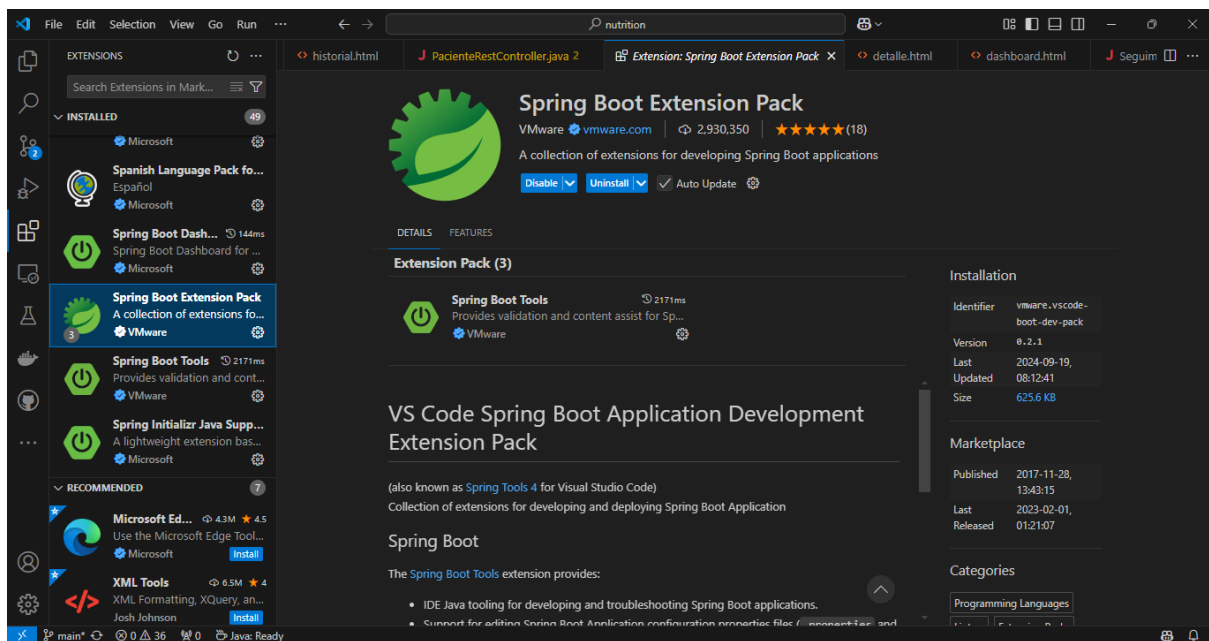


Figura 12 Spring Boot Extension Pack

7 Desarrollo

7.1 Documento de Requerimientos

7.1.1 Requerimientos Funcionales

1. Registro de usuarios

- Permitir el registro de nutricionistas y pacientes mediante un formulario que incluya:
 - Nombre, Apellido, Edad, Género, Estatura, Peso.
 - Correo electrónico y Contraseña.
 - Enfermedades crónicas (diabetes, hipertensión, etc.).
 - Preferencias alimenticias (restricciones dietéticas, alergias).

2. Gestión de pacientes

- Registrar y almacenar el historial nutricional y médico de los pacientes.
- Consultar y editar los datos de los pacientes por parte del nutricionista.

3. Generación de planes de dieta

- Crear planes personalizados basados en:
 - Información médica y preferencias alimenticias del paciente.
 - Requerimientos calóricos y macronutrientes.
 - Cultura alimenticia, lugar de residencia y hábitos.

4. Seguimiento y monitoreo

- Registrar la ingesta diaria de alimentos y actividad física del paciente.
- Mostrar un historial de progreso en términos de peso, IMC y cumplimiento del plan.

5. Notificaciones

- Enviar recordatorios automáticos para el paciente sobre comidas, actividad física y citas.

6. Interacción

- Proveer un chat integrado para la comunicación en tiempo real entre nutricionista y paciente.
- Implementar un sistema de tickets para:
 - Crear, asignar y gestionar solicitudes (como consultas específicas, problemas técnicos, o ajustes en el plan de dieta).

- Seguimiento de estados: abierto, en progreso, resuelto.
- Respuestas automáticas para tickets comunes con base en plantillas.

7.1.2 Requerimientos No Funcionales

1. Seguridad

- Implementar autenticación y autorización mediante JWT.
- Control de acceso basado en roles (RBAC).

2. Usabilidad

- Interfaz de usuario intuitiva y fácil de usar.
- Diseño responsivo para dispositivos móviles y de escritorio.

3. Rendimiento

- Respuesta rápida de la aplicación en todas las operaciones.
- Capacidad de manejar múltiples usuarios concurrentes sin degradación del rendimiento.

4. Escalabilidad

- Utilizar microservicios para permitir el escalamiento horizontal de componentes individuales.
- Desplegar contenedores Docker y orquestar con Kubernetes para facilitar el escalamiento y la gestión.

5. Mantenibilidad

- Código bien documentado y estructurado.
- Uso de Spring Boot y MVC para la separación de responsabilidades y facilitar el mantenimiento y las actualizaciones.

6. Disponibilidad

- Alta disponibilidad mediante la configuración de clústeres y balanceo de carga con NGINX.
- Despliegue en entornos de nube para asegurar redundancia y recuperación ante desastres.

7. Compatibilidad

- APIs RESTful para asegurar la integración con otros sistemas y aplicaciones.
- Soporte para múltiples navegadores y versiones.

8. Persistencia de Datos

- Utilizar Neo4j para modelar relaciones complejas entre entidades.
- Utilizar PostgreSQL para datos transaccionales y persistencia estructurada.

7.2 Funcionalidades F (Marco FURPS)

7.2.1 Funcionalidades (Functionality)

1. Registro de usuarios:

- Permitir el registro de nutricionistas y pacientes mediante un formulario que incluya:
 - Nombre, Apellido, Edad, Género, Estatura, Peso.
 - Correo electrónico y Contraseña.
 - Enfermedades crónicas (diabetes, hipertensión, etc.).
 - Preferencias alimenticias (restricciones dietéticas, alergias).

2. Gestión de pacientes:

- Registrar y almacenar el historial nutricional y médico de los pacientes.
- Consultar y editar los datos de los pacientes por parte del nutricionista.

3. Generación de planes de dieta:

- Crear planes personalizados basados en:
 - Información médica y preferencias alimenticias del paciente.
 - Requerimientos calóricos y macronutrientes.
 - Cultura alimenticia, lugar de residencia y hábitos.

4. Seguimiento y monitoreo:

- Registrar la ingesta diaria de alimentos y actividad física del paciente.
- Mostrar un historial de progreso en términos de peso, IMC y cumplimiento del plan.

5. Notificaciones:

- Recordatorios automáticos para el paciente sobre comidas, actividad física y citas.

6. Interacción:

- Chat integrado: Comunicación en tiempo real entre nutricionista y paciente.
- Sistema de tickets:
 - Crear, asignar y gestionar solicitudes (como consultas específicas, problemas técnicos, o ajustes en el plan de dieta).
 - Seguimiento de estados: abierto, en progreso, resuelto.
 - Respuestas automáticas para tickets comunes con base en plantillas.

7. Cumplimiento normativo:

- Incorporar lineamientos de normas aplicables (por ejemplo, NOM-004- SSA3-2012 para expedientes clínicos).
 - Generar reportes estándar para auditorías.
8. Endpoints REST: Proveer un conjunto de endpoints REST para interactuar con el sistema, como:
- Pacientes:
 - POST /api/patients: Registrar un nuevo paciente.
 - GET /api/patients: Consultar la lista de pacientes.
 - GET /api/patients/{id}: Obtener información detallada de un paciente.
 - PUT /api/patients/{id}: Editar información de un paciente.
 - DELETE /api/patients/{id}: Eliminar un paciente.
 - Consultas:
 - POST /api/consultations: Registrar una nueva consulta médica.
 - GET /api/consultations: Consultar todas las consultas.
 - GET /api/consultations/{id}: Obtener detalles de una consulta específica.
 - Tickets:
 - POST /api/tickets: Crear un nuevo ticket.
 - GET /api/tickets: Listar tickets generados.
 - GET /api/tickets/{id}: Consultar el estado y detalles de un ticket.
 - PUT /api/tickets/{id}: Actualizar el estado de un ticket.

7.2.2 Usabilidad (Usability)

1. Interfaz intuitiva:

o Diseño centrado en la experiencia del usuario (UI/UX) con principios claros:

- Uso de colores que indiquen estados (verde para éxito, rojo para advertencias).
- Botones descriptivos y formularios claros.
- Uso mínimo de texto para evitar saturación visual.

2. Navegación simplificada:

- Diseño jerárquico con accesos rápidos a funciones clave.
- Menú fijo y accesible desde cualquier sección de la aplicación.

3. Acceso multiplataforma:

- Diseño responsivo para dispositivos móviles y desktop.

- Aplicación móvil desarrollada para iOS y Android.
- 4. **Soporte de idiomas:**
 - Disponible en varios idiomas, incluyendo español e inglés.
- 5. **Manual de usuario:**
 - Guías detalladas con capturas de pantalla y videos explicativos para todas las funciones.

7.2.3 Confiabilidad (Reliability)

1. Respaldo de datos:
 - Copias de seguridad automáticas en la nube cada 24 horas.
 - Recuperación inmediata ante fallos.
2. Autenticación segura:
 - Contraseñas cifradas y autenticación de dos factores.
3. Disponibilidad:
 - Uptime garantizado al 99.9%.
 - Minimizar el consumo de recursos en procesos en segundo plano

7.2.4 Rendimiento (Performance)

1. Tiempo de respuesta:
 - Consultas críticas deben ejecutarse en menos de 3 segundos.
2. Escalabilidad:
 - Soportar hasta 10,000 usuarios concurrentes.
3. Optimización:
 - Consultas y transacciones deben ser eficientes y rápidas.

7.2.5 Soporte (Support)

1. Compatibilidad:
 - Funcionalidad asegurada en navegadores modernos y sistemas operativos actualizados.
2. Mantenimiento:
 - Actualizaciones de seguridad regulares.
3. Configurabilidad:

- Ajustes rápidos para adaptarse a nuevas normativas o necesidades.
4. Documentación:
- Manuales técnicos y de usuario detallados.

7.2.6 Ontología o Base de Datos Orientada a Grafos para Mejorar el Sistema

7.2.7 Uso de Ontología

Se hará uso de una ontología para estructurar el conocimiento y realizar inferencias avanzadas:

- **Clases principales:** Paciente, Nutricionista, Alimento, Plan de dieta.
- **Propiedades:** Relaciones entre clases como:
 - Paciente → Plan de dieta.
 - Paciente → Preferencias alimenticias.
 - Alimentos → Nutrientes.
- **Herramientas recomendadas:**
 - **Protégé:** Modelado.
 - **OWL:** Lenguaje de definición.
 - **Razonadores como Pellet o HermiT:** Inferencias.

7.2.8 Base de Datos Orientada a Grafos

Utilizar Neo4j para modelar relaciones entre pacientes, alimentos y planes:

- **Nodos:** Paciente, Nutricionista, Alimento, Plan de dieta.
- **Relaciones:** (Paciente)-[RESTRINGE]->(Alimento).
- **Consultas:** Realizar inferencias dinámicas mediante Cypher.

7.2.9 UI/UX Específico para la Aplicación

1. **Diseño visual:**
 - Minimalismo: Pantallas limpias y con elementos necesarios.
 - Colores suaves y contrastes que no cansen la vista.

2. **Accesibilidad:**

- Texto con tamaño ajustable.
- Inclusión de etiquetas accesibles para lectores de pantalla.

3. **Flujo de usuario:**

o Proceso de registro e inicio de sesión claro y rápido.

- Indicadores visuales para pasos completados o errores.

4. **Sistema de tickets:**

- Diseño intuitivo para la creación de solicitudes.
- Muestra de un tablero con el estado de cada ticket.
- Notificaciones para cambios en los estados del ticket.

5. **Prototipado y pruebas:**

- Probar con usuarios reales antes de implementación final.

7.3 Diagrama de Casos de Uso

7.3.1 Registro de Usuarios

Este diagrama representa el proceso de registro en el sistema, donde el Usuario No Registrado puede crear una cuenta proporcionando sus datos personales (nombre, apellido, correo, contraseña), datos antropométricos (edad, género, estatura, peso), información sobre enfermedades crónicas y preferencias alimenticias. El sistema se encarga de validar los datos ingresados, mientras que el Administrador tiene la capacidad de asignar roles específicos (Nutricionista o Paciente) a los usuarios registrados.

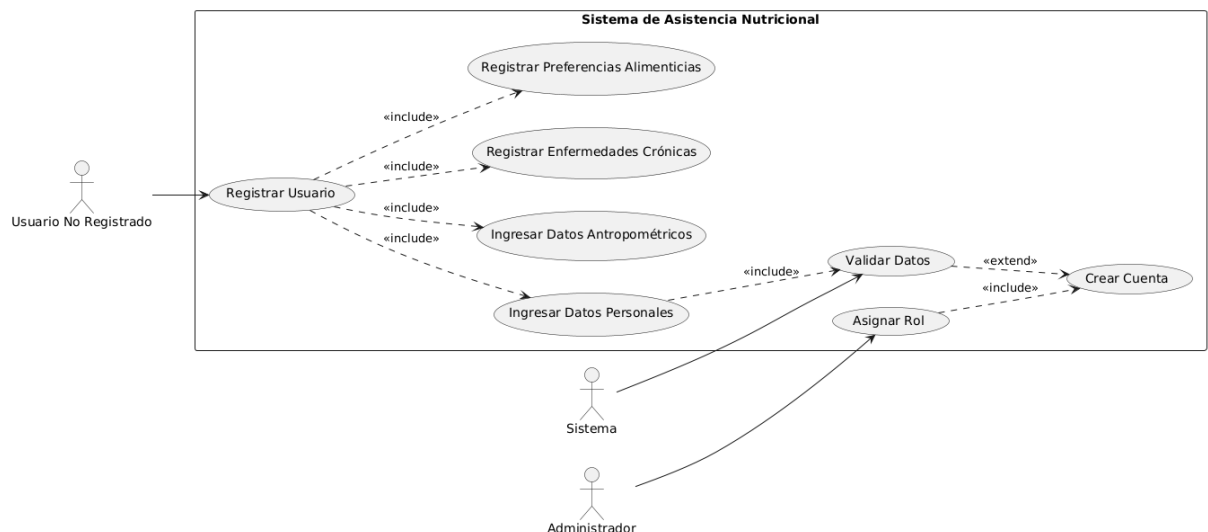


Figura 13

Figura 14 Registro de Usuarios

Actores:

- Usuario No Registrado (actor principal)
- Administrador (para asignación de roles)
- Sistema (para validaciones)

Casos de uso principales:

- Registrar Usuario (caso de uso principal)
- Ingresar Datos Personales
- Ingresar Datos Antropométricos
- Registrar Enfermedades Crónicas
- Registrar Preferencias Alimenticias
- Validar Datos

- Asignar Rol
- Crear Cuenta

Relaciones:

- Include: para procesos obligatorios
- Extend: para procesos opcionales o condicionales

7.3.2 Gestión de Pacientes

Este diagrama ilustra cómo el Nutricionista puede administrar la información de sus pacientes, incluyendo la consulta y edición del historial nutricional (peso, IMC, medidas corporales) y médico (enfermedades, alergias, medicamentos). El sistema permite registrar nuevas consultas y actualizar el historial, mientras que el Paciente puede ver su propio progreso y, opcionalmente, editar ciertos datos personales, todo sujeto a validación del sistema.

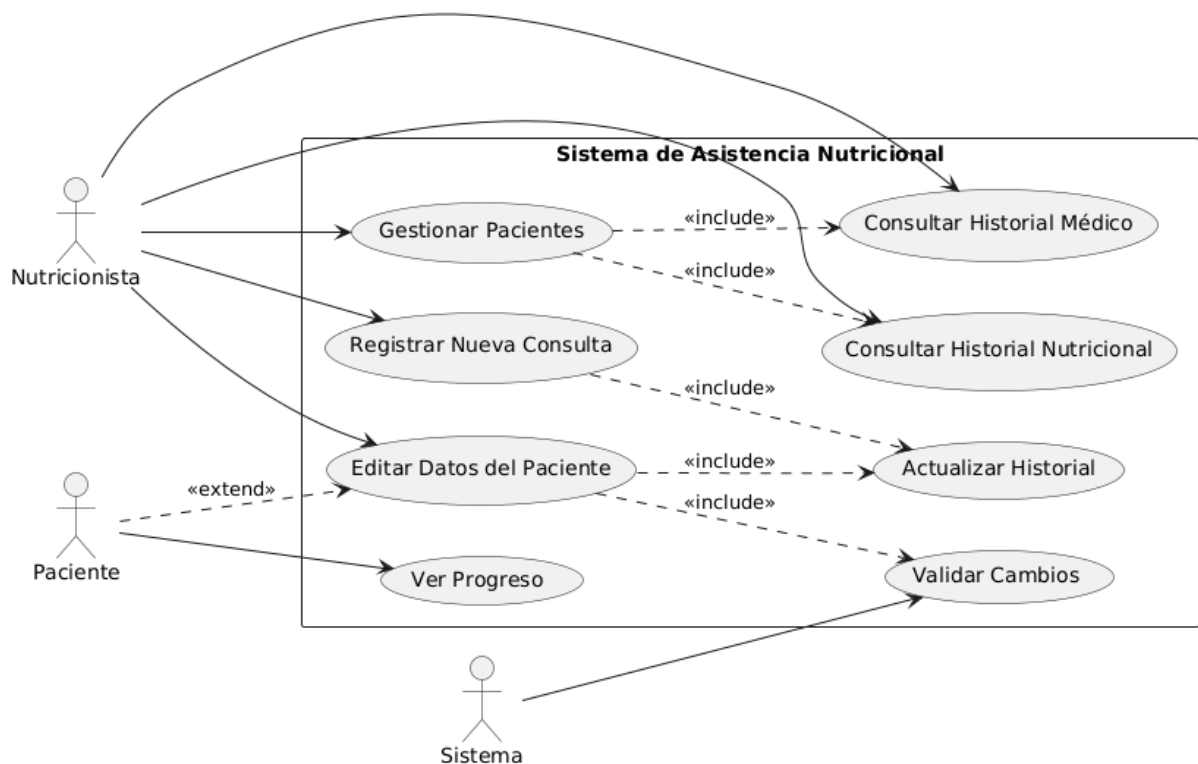


Figura 15 Gestión de Pacientes

Actores:

- Nutricionista (actor principal)
- Paciente (actor secundario)
- Sistema (para validaciones)

Casos de uso principales:

- Gestionar Pacientes (caso de uso principal)
- Consultar Historial Nutricional
- Consultar Historial Médico
- Editar Datos del Paciente
- Actualizar Historial
- Registrar Nueva Consulta
- Ver Progreso
- Validar Cambios

7.3.3 Generación de Planes de Dieta

Este diagrama muestra el proceso completo de creación de planes de dieta personalizados, donde el Nutricionista es el actor principal que consulta la información médica del paciente, verifica preferencias alimenticias, y utiliza el sistema para calcular requerimientos calóricos y definir macronutrientes. El proceso considera factores culturales y locales para la selección de alimentos y la creación de horarios de comidas, requiriendo validación del sistema y aprobación final del nutricionista antes de que el paciente pueda visualizarlo.

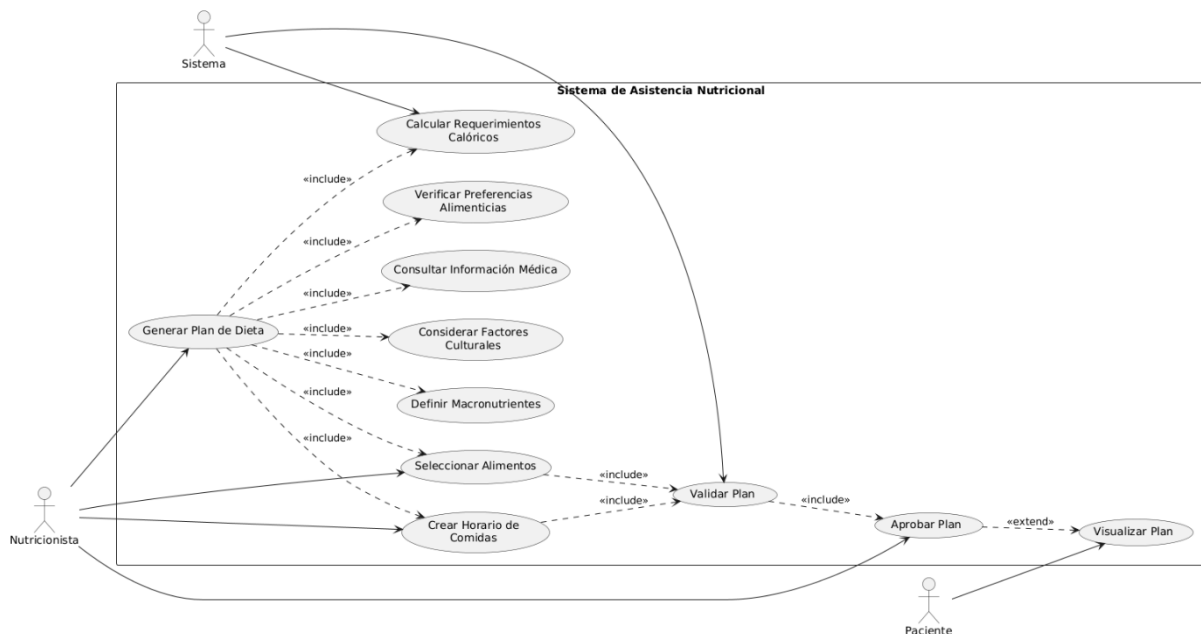


Figura 16 Generación de Planes de Dieta

Actores:

- Nutricionista (actor principal)

- Sistema (para cálculos y validaciones)
- Paciente (para visualización)

Casos de uso principales:

- Generar Plan de Dieta (caso de uso principal)
- Consultar Información Médica
- Verificar Preferencias Alimenticias
- Calcular Requerimientos Calóricos
- Definir Macronutrientes
- Considerar Factores Culturales
- Seleccionar Alimentos
- Crear Horario de Comidas
- Validar Plan
- Aprobar Plan
- Visualizar Plan

7.3.4 Seguimiento y Monitoreo

Este diagrama representa cómo el Paciente puede registrar su actividad diaria, incluyendo la ingesta de alimentos y actividad física, así como actualizar sus medidas corporales. El Sistema calcula automáticamente el IMC y valida los datos ingresados, mientras que el Nutricionista puede evaluar el cumplimiento del plan, visualizar estadísticas y generar reportes de progreso, permitiendo un seguimiento detallado del avance del paciente.

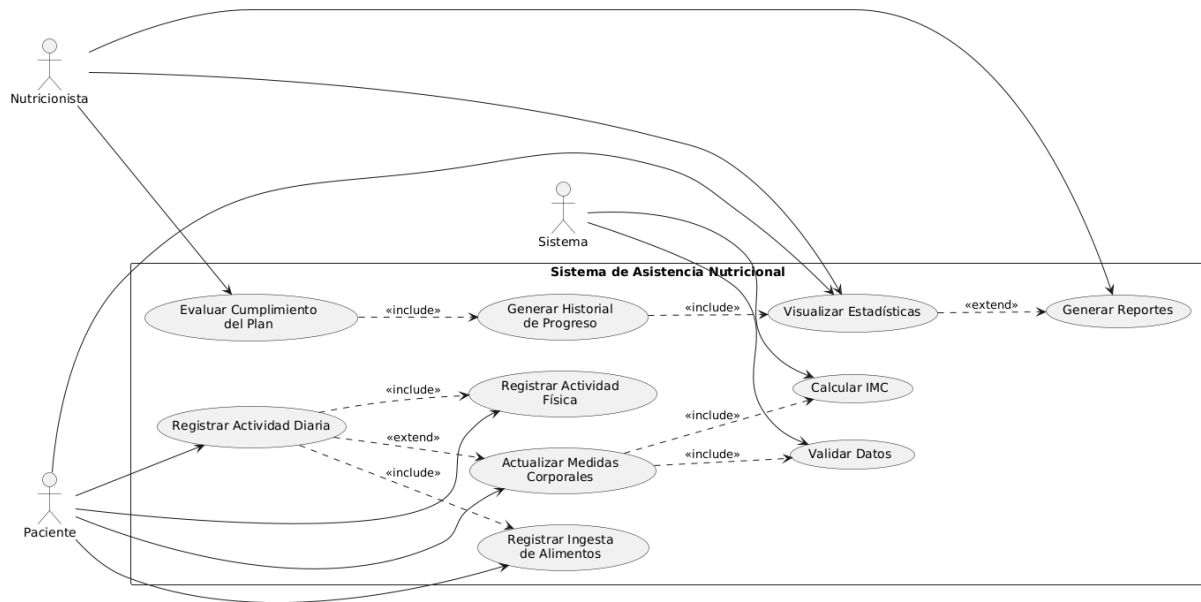


Figura 17 Seguimiento y Monitoreo

Actores:

- Paciente (actor principal)
- Nutricionista (para evaluación y reportes)
- Sistema (para cálculos y validaciones)

Casos de uso principales:

- Registrar Actividad Diaria
- Registrar Ingesta de Alimentos
- Registrar Actividad Física
- Actualizar Medidas Corporales
- Calcular IMC
- Generar Historial de Progreso
- Evaluar Cumplimiento del Plan
- Visualizar Estadísticas
- Generar Reportes
- Validar Datos

7.3.5 Sistema de Interacción

Este diagrama abarca dos componentes principales: el chat en tiempo real y el sistema de tickets. El chat permite la comunicación directa entre Paciente y Nutricionista, con capacidad de adjuntar archivos y ver el historial de conversaciones. El sistema de tickets permite a los pacientes crear solicitudes específicas que pueden

ser asignadas, gestionadas y respondidas por los nutricionistas, con la capacidad del sistema de generar respuestas automáticas para consultas comunes y mantener un seguimiento del estado de cada ticket.



Figura 18 Sistema de Interacción

Actores:

- Paciente (usuario del sistema)
- Nutricionista (gestor principal)
- Sistema (para respuestas automáticas)

Casos de uso principales divididos en dos grupos: Chat:

- Gestionar Chat
- Iniciar Conversación
- Enviar Mensaje
- Ver Historial de Chat
- Adjuntar Archivos

Sistema de Tickets:

- Gestionar Tickets
- Crear Ticket
- Asignar Ticket
- Actualizar Estado
- Responder Ticket
- Generar Respuesta Automática
- Cerrar Ticket
- Ver Historial de Tickets

7.4 Prototipos de Interfaces Gráficas

7.4.1 Prototipo de Interfaz –Login de usuario

Imagen del Prototipo:

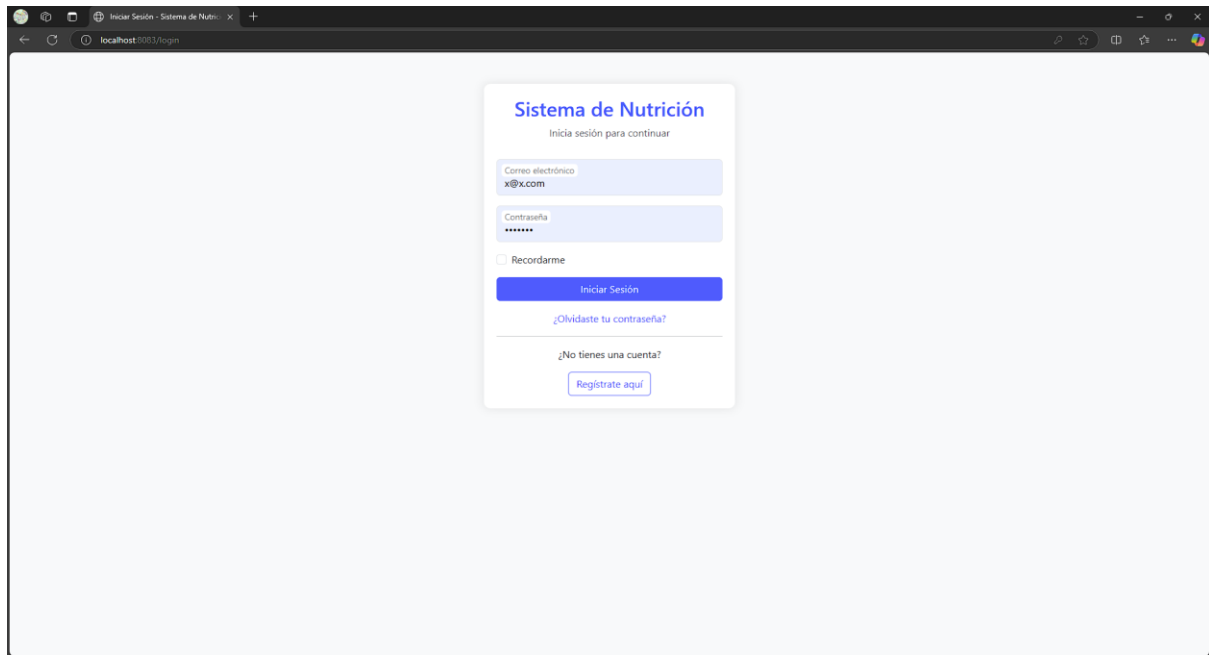


Figura 19 Login de usuario.

Descripción: Esta vista permite a los usuarios autenticarse en el sistema. Los usuarios deben ingresar su correo electrónico y contraseña para acceder a sus cuentas.

Funcionalidades:

- Formulario de ingreso de correo electrónico y contraseña.
- Validación de credenciales.
- Mensajes de error en caso de credenciales incorrectas.
- Opción para recuperar la contraseña.

7.4.2 Prototipo de Interfaz –Registro de usuario

Imagen del Prototipo:

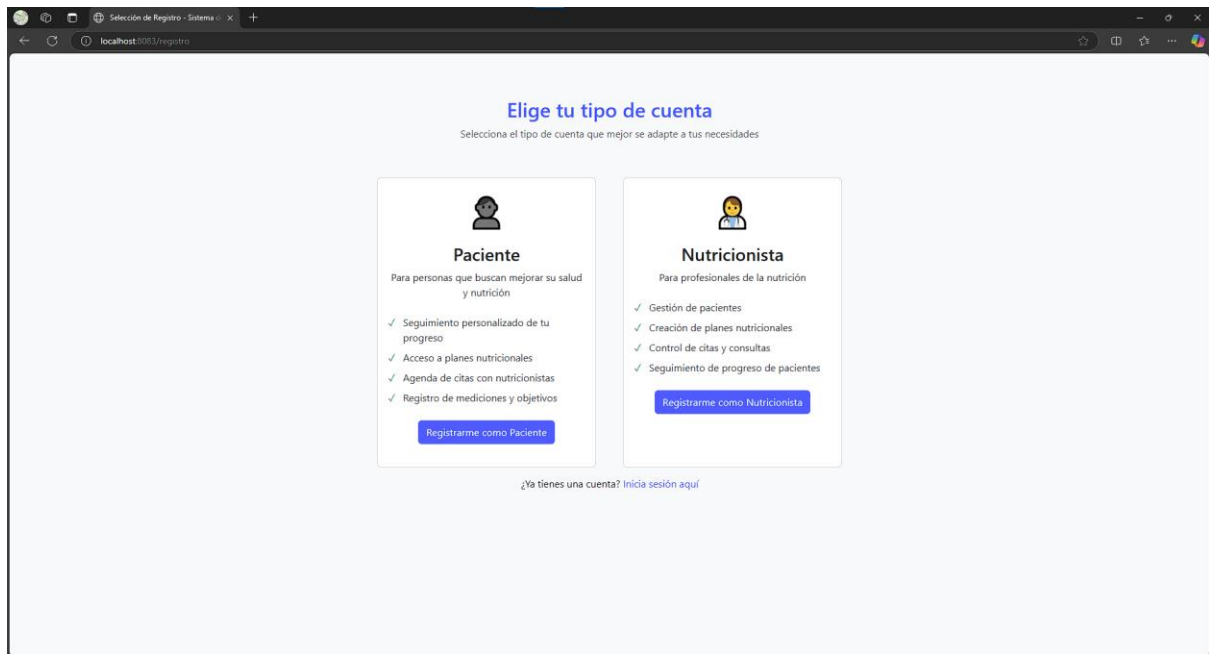


Figura 20 Registro de Usuario.

Descripción: Esta vista permite a los nuevos usuarios registrarse en el sistema, ya sea como pacientes o nutricionistas.

Funcionalidades:

- Formulario de registro que incluye nombre, apellido, edad, género, estatura, peso, correo electrónico, contraseña, enfermedades crónicas y preferencias alimenticias.
- Validación de campos obligatorios.
- Creación de cuenta y almacenamiento de datos en la base de datos.

- Mensajes de confirmación en caso de registro exitoso.

7.4.3 Prototipo de Interfaz – Dashboard de paciente

Imagen del Prototipo:

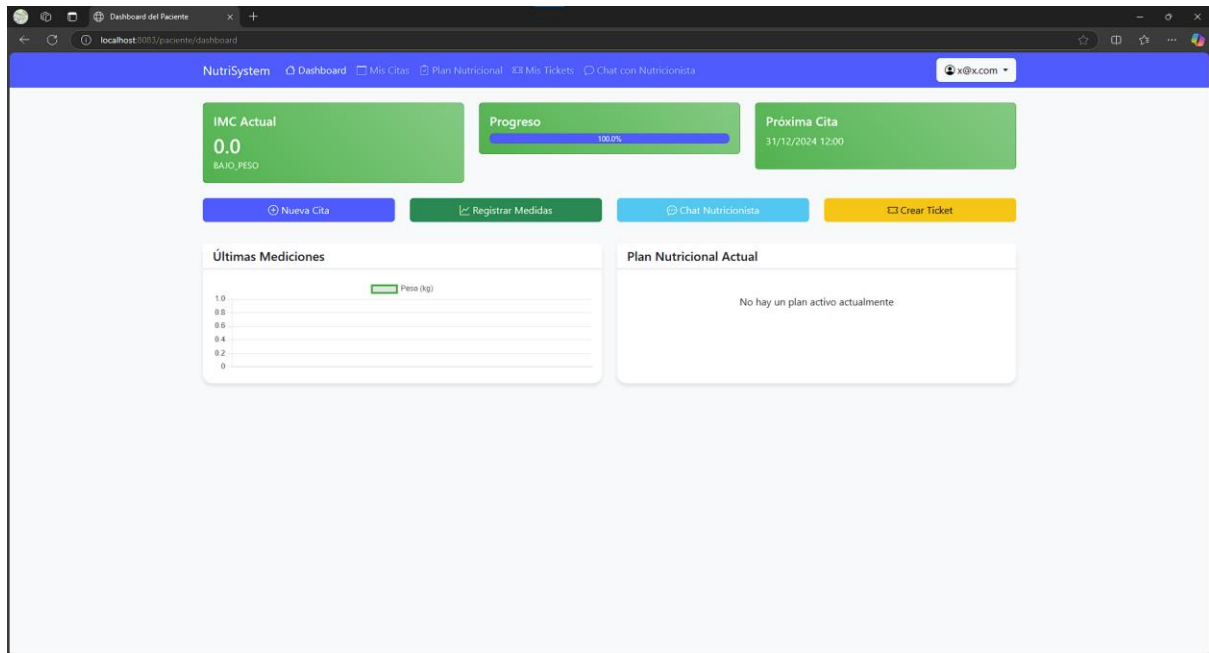


Figura 21 Dashboard de paciente.

Descripción: Esta vista es el panel principal para los pacientes, donde pueden acceder a su información y funcionalidades relacionadas con su plan nutricional.

Funcionalidades:

- Visualización del plan nutricional actual.
- Registro de ingesta diaria de alimentos.
- Monitoreo del progreso en términos de peso e IMC.
- Acceso a citas próximas y pendientes.

7.4.4 Prototipo de Interfaz – Dashboard de nutricionalista

Imagen del Prototipo:

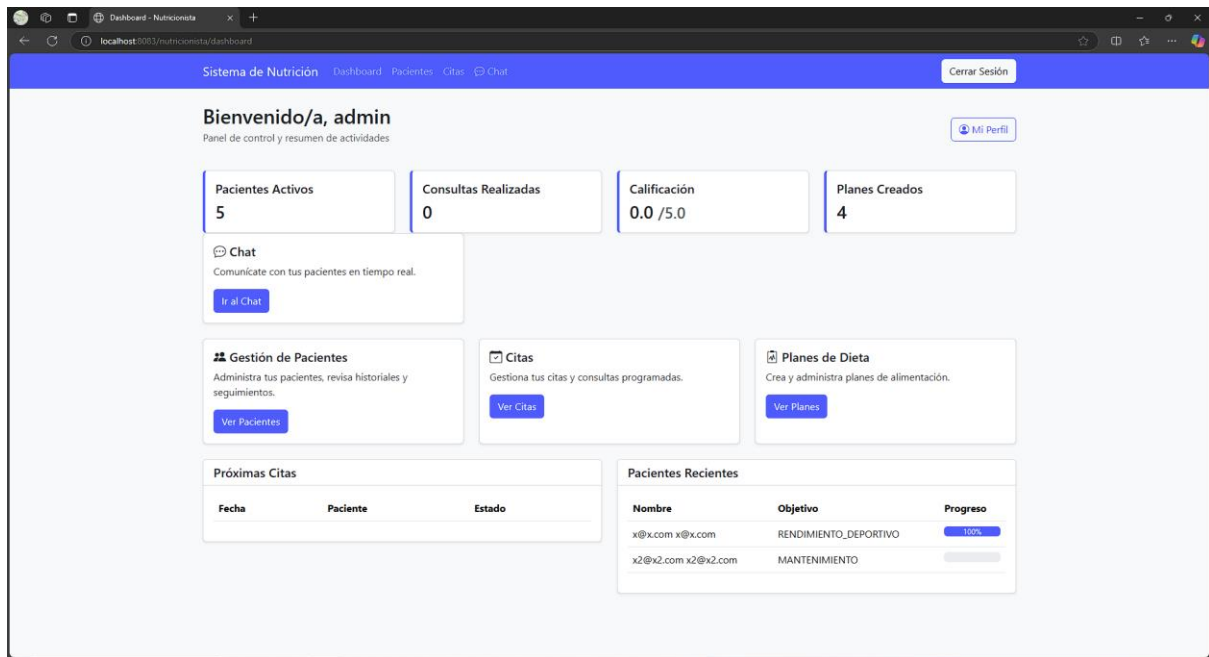


Figura 22 Dashboard de nutricionista.

Descripción: Esta vista es el panel principal para los nutricionistas, donde pueden gestionar sus pacientes y acceder a diferentes funcionalidades del sistema.

Funcionalidades:

- Visualización de la lista de pacientes.
- Acceso rápido a la creación y edición de planes nutricionales.
- Resumen de citas próximas y pendientes.
- Acceso a notificaciones y mensajes.

7.4.5 Prototipo de Interfaz - Historial de citas

Imagen del Prototipo:

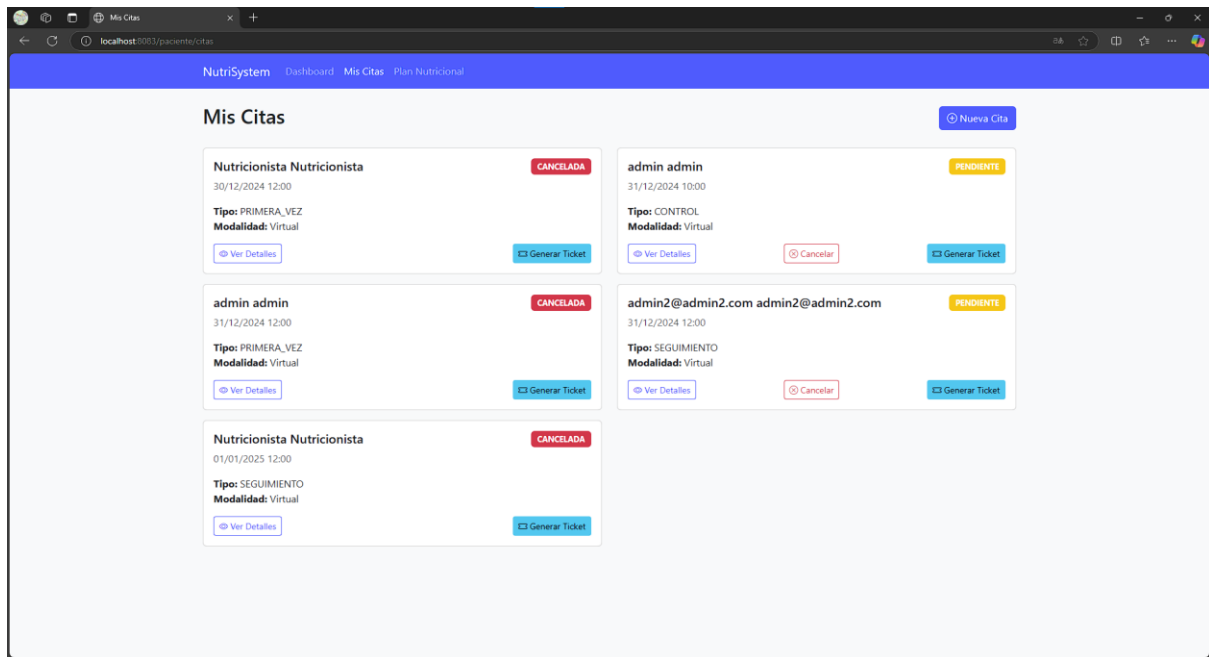


Figura 23 Historial de Citas.

Descripción: Esta vista permite a los usuarios (pacientes y nutricionistas) consultar el historial de citas pasadas y próximas.

Funcionalidades:

- Listado de citas con detalles como fecha, hora y estado.
- Filtros para buscar citas por fecha o estado.
- Opción para cancelar o reprogramar citas

7.4.6 Prototipo de Interfaz - Lista de Planes Nutricionales

Imagen del Prototipo:

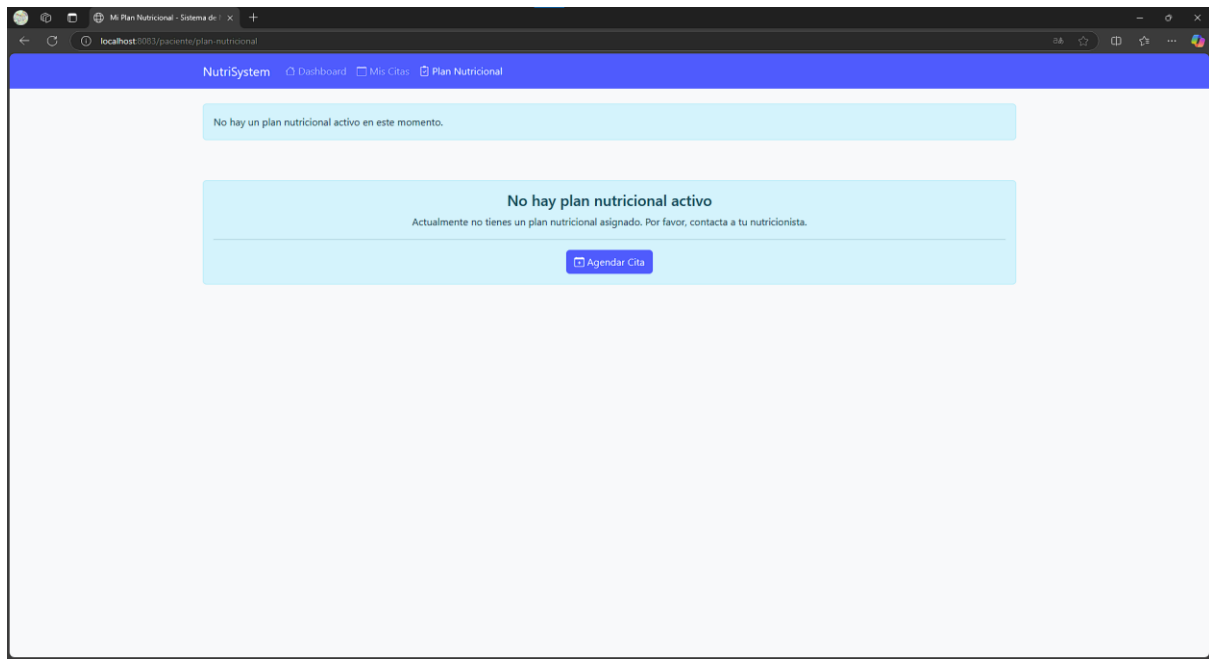


Figura 24 Lista de Planes Nutricionales.

Descripción: Esta vista permite a los nutricionistas gestionar los planes nutricionales de sus pacientes.

Funcionalidades:

- Listado de planes nutricionales creados.
- Opción para crear un nuevo plan.
- Edición y eliminación de planes existentes.
- Visualización de detalles de cada plan, incluyendo recomendaciones y restricciones dietéticas.

7.4.7 Prototipo de Interfaz - Chat

Imagen del Prototipo:

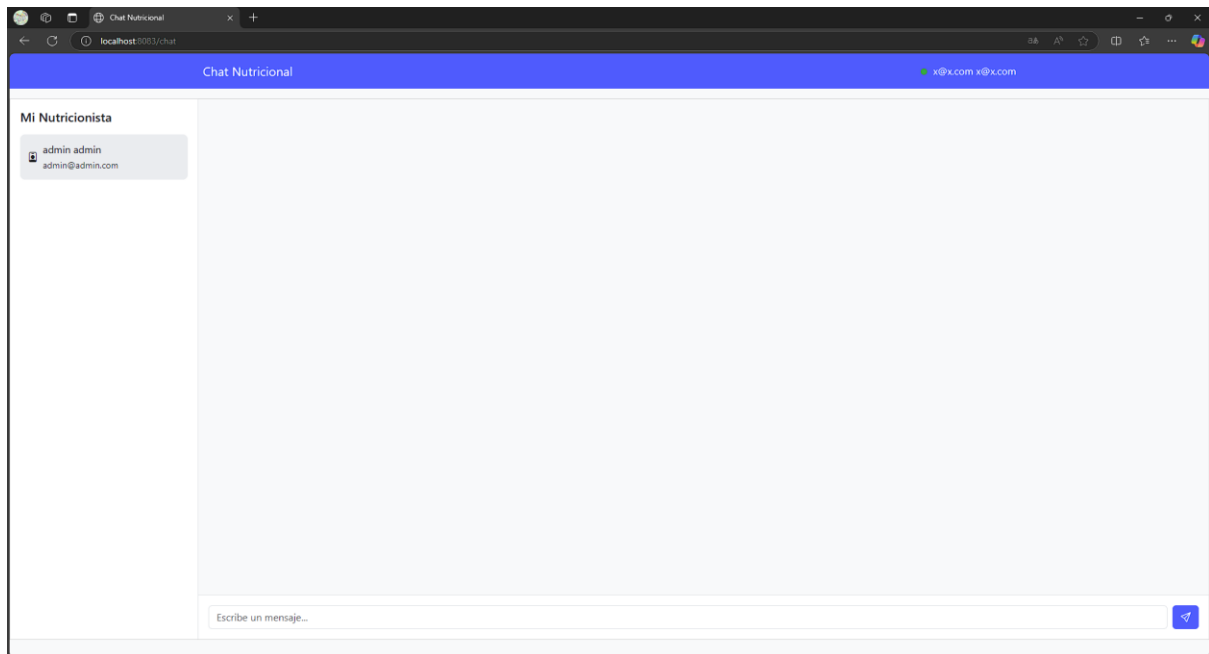


Figura 25 Chat.

Descripción: Esta vista permite la comunicación en tiempo real entre nutricionistas y pacientes.

Funcionalidades:

- Interfaz de mensajería instantánea.
- Notificaciones de nuevos mensajes.
- Historial de conversaciones.

7.4.8 Prototipo de Interfaz – Historial de tickets

Imagen del Prototipo:

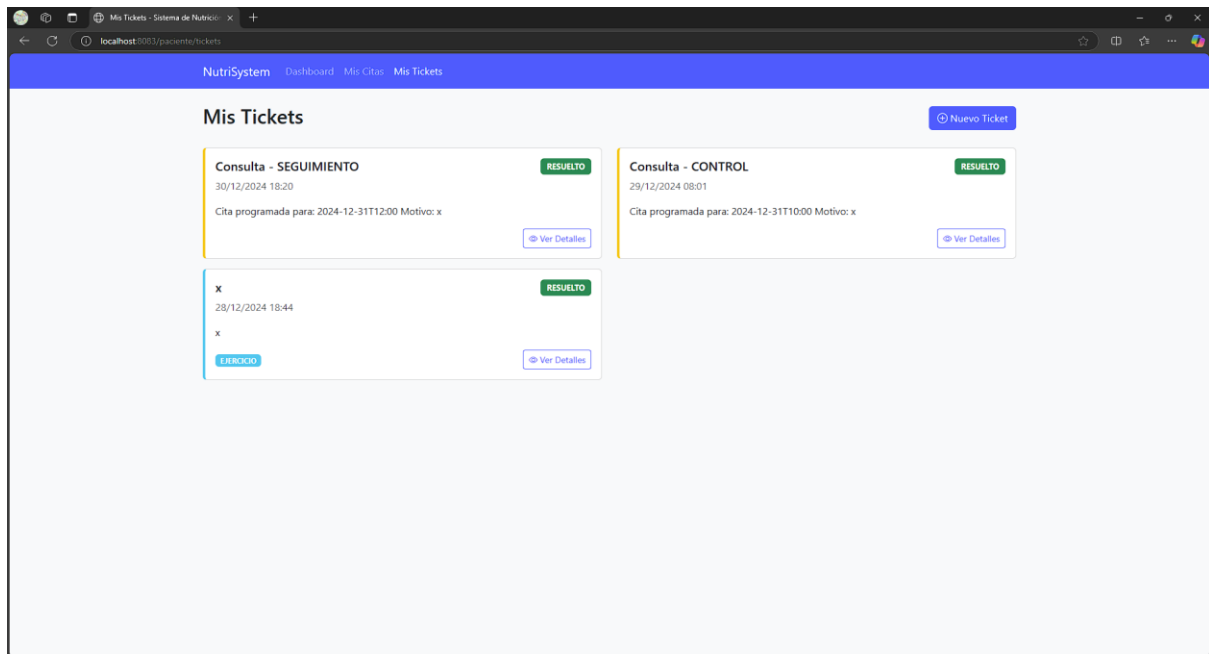


Figura 26 Historial de tickets

Descripción: Esta vista permite a los usuarios gestionar y consultar el historial de tickets de soporte o consultas.

Funcionalidades:

- Listado de tickets con detalles como fecha de creación, estado y descripción.
- Creación de nuevos tickets.
- Actualización del estado de los tickets existentes.
- Respuesta automática para tickets comunes basados en plantillas.

7.4.9 Prototipo de Interfaz - Perfil de Usuario

Imagen del Prototipo:

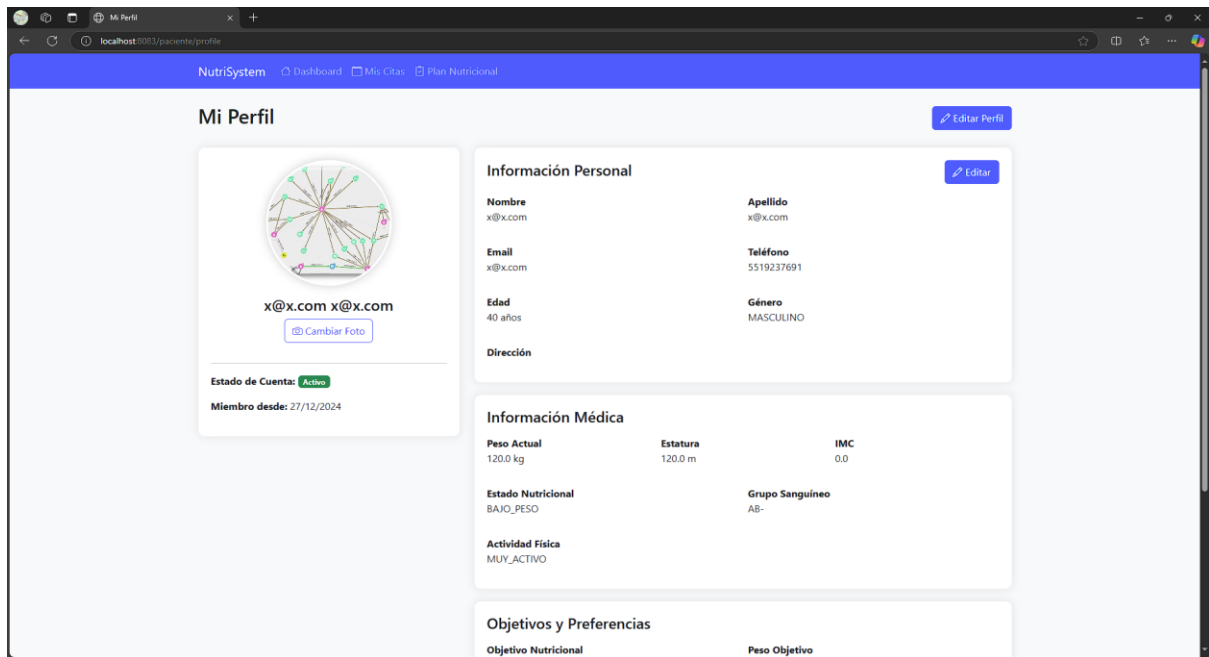


Figura 27 Perfil de Usuario.

Descripción: Esta vista permite a los usuarios (pacientes y nutricionistas) gestionar su información personal y de cuenta.

Funcionalidades:

- Visualización y edición de información personal (nombre, correo electrónico, etc.).
- Cambio de contraseña.
- Actualización de preferencias alimenticias y enfermedades crónicas.
- Opción para eliminar la cuenta (si aplica).

8 Pruebas de API REST con Postman/Retrofit

8.1 Pacientes

- GET /api/pacientes: Consultar la lista de pacientes.

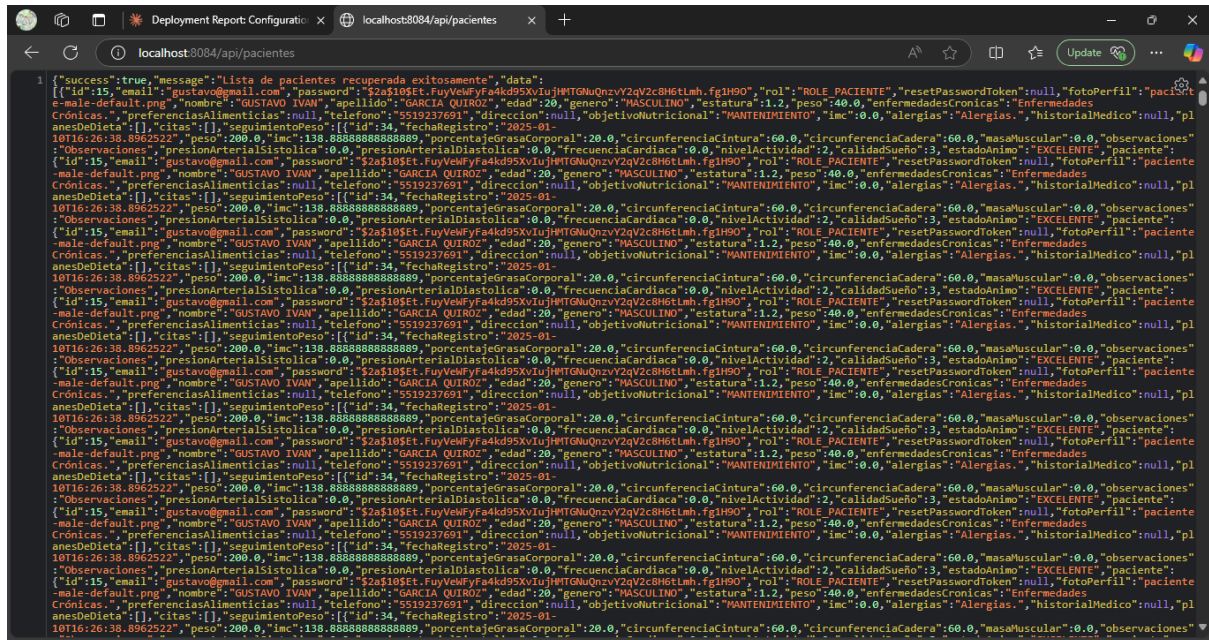


Figura 28 GET /api/pacientes

Figura 29

- GET /api/pacientes/{id}: Obtener información detallada de un paciente.

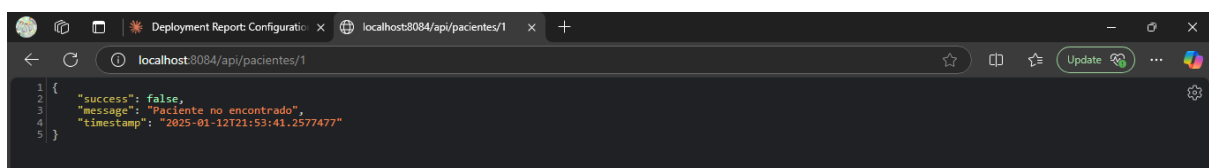
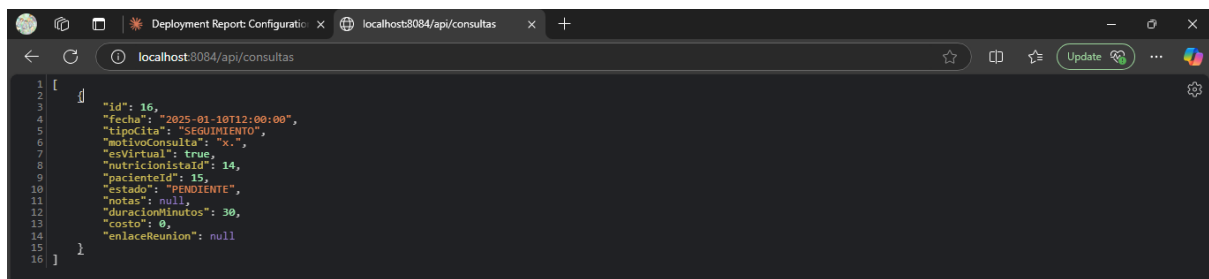


Figura 30 GET /api/pacientes/{id}

- PUT /api/pacientes/{id}: Editar información de un paciente.
- DELETE /api/pacientes/{id}: Eliminar un paciente.

8.2 Consultas:

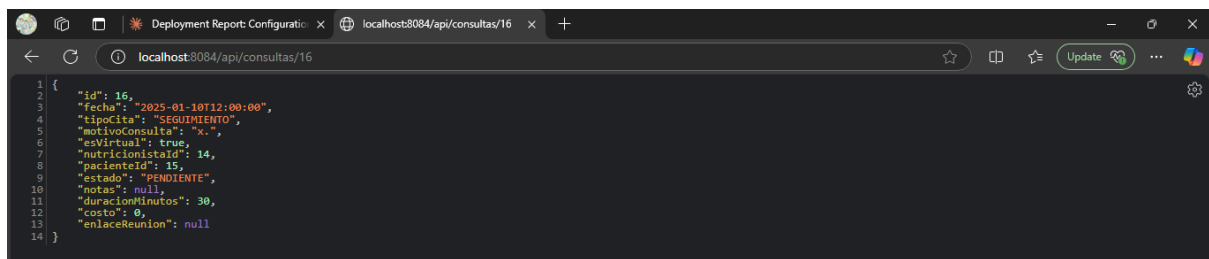
- POST /api/consultas: Registrar una nueva consulta médica.
- GET /api/consultas: Consultar todas las consultas.



```
1 [{
2   "id": 16,
3   "fecha": "2025-01-10T12:00:00",
4   "tipoCita": "SEGUIMIENTO",
5   "motivoConsulta": "x.",
6   "esVirtual": true,
7   "nutricionistaId": 14,
8   "pacienteId": 15,
9   "estado": "PENDIENTE",
10  "notas": null,
11  "duracionMinutos": 30,
12  "costo": 0,
13  "enlaceReunion": null
14 }]
```

Figura 31 GET /api/consultas

- GET /api/consultas/{id}: Obtener detalles de una consulta específica.

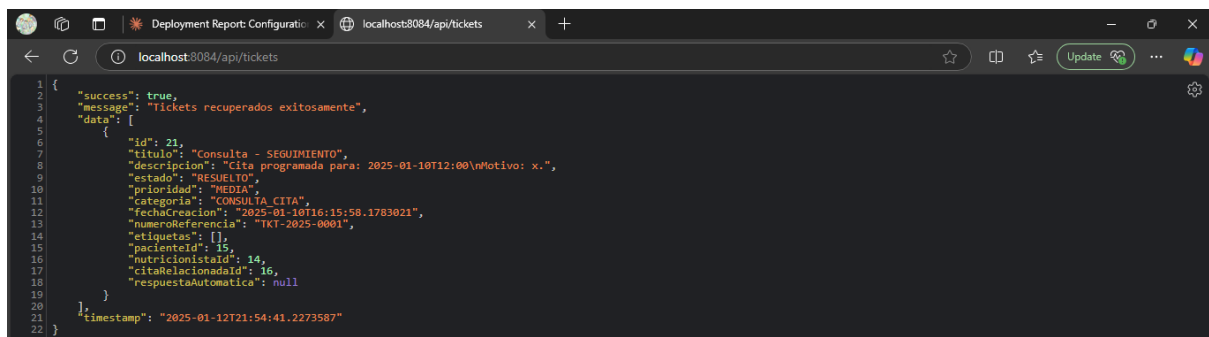


```
1 {
2   "id": 16,
3   "fecha": "2025-01-10T12:00:00",
4   "tipoCita": "SEGUIMIENTO",
5   "motivoConsulta": "x.",
6   "esVirtual": true,
7   "nutricionistaId": 14,
8   "pacienteId": 15,
9   "estado": "PENDIENTE",
10  "notas": null,
11  "duracionMinutos": 30,
12  "costo": 0,
13  "enlaceReunion": null
14 }
```

Figura 32 GET /api/consultas/{id}

8.3 Tickets

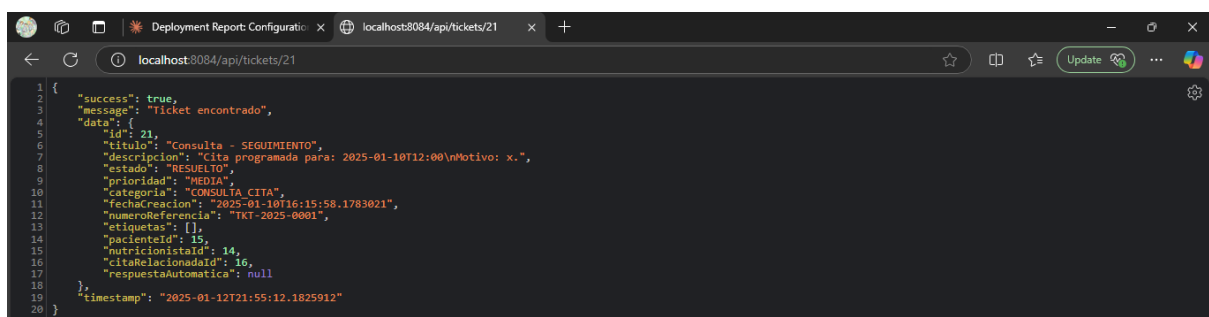
- POST /api/tickets: Crear un nuevo ticket.
- GET /api/tickets: Listar tickets generados.



```
1 {
2   "success": true,
3   "message": "Tickets recuperados exitosamente",
4   "data": [
5     {
6       "id": 21,
7       "titulo": "Consulta - SEGUIMIENTO",
8       "descripcion": "Cita programada para: 2025-01-10T12:00\nMotivo: x.",
9       "estado": "RESUELTO",
10      "prioridad": "MEDIA",
11      "categoria": "CONSULTA CITA",
12      "fechaCreacion": "2025-01-10T16:15:58.1783021",
13      "numeroReferencia": "TKT-2025-0001",
14      "etiquetas": [],
15      "pacienteId": 15,
16      "nutricionistaId": 14,
17      "citaRelacionadaId": 16,
18      "respuestaAutomatica": null
19    }
20  ],
21   "timestamp": "2025-01-12T21:54:41.2273587"
22 }
```

Figura 33 GET /api/tickets

- GET /api/tickets/{id}: Consultar el estado y detalles de un ticket.



```
1 {
2   "success": true,
3   "message": "Ticket encontrado",
4   "data": {
5     "id": 21,
6     "titulo": "Consulta - SEGUIMIENTO",
7     "descripcion": "Cita programada para: 2025-01-10T12:00\nMotivo: x.",
8     "estado": "RESUELTO",
9     "prioridad": "MEDIA",
10    "categoria": "CONSULTA CITA",
11    "fechaCreacion": "2025-01-10T16:15:58.1783021",
12    "numeroReferencia": "TKT-2025-0001",
13    "etiquetas": [],
14    "pacienteId": 15,
15    "nutricionistaId": 14,
16    "citaRelacionadaId": 16,
17    "respuestaAutomatica": null
18  },
19   "timestamp": "2025-01-12T21:55:12.1825912"
20 }
```

Figura 34 GET /api/tickets/{id}

- PUT /api/tickets/{id}: Actualizar el estado de un ticket.

9 Conclusiones

En conclusión, el **Sistema de Asistencia Nutricional** representa una solución integral y avanzada para la gestión personalizada de planes nutricionales y el seguimiento de pacientes, combinando tecnologías modernas, estándares de calidad y prácticas de desarrollo seguro. Su arquitectura basada en **Spring Boot**, con un enfoque modular y escalable, garantiza un rendimiento óptimo, mientras que la implementación de bases de datos especializadas como **Neo4j** y **PostgreSQL** permite manejar tanto relaciones complejas entre entidades como datos estructurados de manera eficiente. Además, la incorporación de herramientas de comunicación en tiempo real, como el chat y las notificaciones automáticas, mejora significativamente la interacción entre nutricionistas y pacientes, fomentando un acompañamiento más cercano y efectivo.

10 BIBLIOGRAFÍA APA

- Pressman RS. INGENIERIA DE SOFTWARE.; 2010.
- Sommerville I, Velázquez SF. Ingeniería de software.; 2011.
- Sommerville, I. (2015). Ingeniería de Software. Pearson Educación.
- Booch, G., Rumbaugh, J., & Jacobson, I. (2005). The Unified Modeling Language User Guide. Addison-Wesley.