**Introduccion**

NutriRecetas es una plataforma web diseñada para ofrecer información sobre nutrición, recetas saludables y planes alimenticios personalizados. El objetivo principal del proyecto es proporcionar a los usuarios una herramienta interactiva donde puedan acceder a contenido nutricional confiable, registrar su progreso y recibir recomendaciones según sus necesidades.

**Contexto**

En la actualidad, la mala alimentación es un factor clave en enfermedades como la obesidad y la diabetes. Muchas personas buscan información confiable y herramientas digitales para mejorar sus hábitos alimenticios, pero las opciones disponibles suelen ser complejas o costosas.

**Justificacion**

La alimentación saludable es clave para prevenir enfermedades y mejorar la calidad de vida. Sin embargo, muchas personas enfrentan dificultades para acceder a información confiable y herramientas prácticas que las ayuden en este proceso. NutriRecetas nace como una solución accesible y fácil de usar, ofreciendo recetas, planes de alimentación y seguimiento nutricional de manera intuitiva.

Desarrollado con Spring Framework y Java, este proyecto garantiza seguridad y escalabilidad, mientras que Thymeleaf y HTML permiten una experiencia de usuario dinámica y sencilla. Con esta plataforma, se busca no solo brindar información, sino también educar y motivar a las personas a adoptar hábitos saludables de forma práctica y efectiva.

**Objetivo Principal**

Desarrollar una plataforma web interactiva que brinde información confiable sobre nutrición, recetas saludables y planes de alimentación personalizados, utilizando Spring Framework, Java y Thymeleaf, con el fin de fomentar hábitos alimenticios saludables y mejorar la calidad de vida de los usuarios.

Este proyecto busca ofrecer una solución accesible, intuitiva y escalable que permita a los usuarios adoptar una alimentación equilibrada de manera sencilla. Entre sus principales beneficios se encuentran la facilidad de acceso a contenido verificado, la personalización de planes nutricionales y una interfaz amigable que mejora la experiencia del usuario. Además, al utilizar tecnologías modernas y seguras, la plataforma garantiza un rendimiento eficiente y la posibilidad de expandirse con nuevas funcionalidades en el futuro.

**Requerimientos Funcionales**

RF-01: El sistema debe permitir el registro y autenticación de usuarios o nutricionistas.

RF-02: Los usuarios podrán visualizar recetas saludables con detalles nutricionales.

RF-03: El sistema debe permitir la generación de planes de alimentación.

RF-04: Los usuarios podrán registrar su información nutricional (peso, altura, objetivos).

RF-05: El sistema debe calcular el consumo calórico recomendado según el perfil del usuario.

RF-06: Los usuarios podrán agregar recetas.

RF-07: La plataforma debe permitir la edición y eliminación de datos del perfil del usuario.

RF-08: Se debe implementar un panel de administración para la gestión de contenido.

Requerimientos no funcionales

RNF-01: La aplicación debe estar desarrollada con Spring Framework y Java en el backend.

RNF-02: La interfaz debe utilizar Thymeleaf y HTML para una navegación intuitiva.

RNF-03: Los datos del usuario deben almacenarse en una base de datos relacional segura (Ej. MySQL o PostgreSQL).

RNF-04: El sistema debe garantizar la seguridad de los datos mediante autenticación y autorización.

RNF-05: La aplicación debe ser escalable para futuras mejoras y nuevas funcionalidades.

RNF-06: Los tiempos de respuesta deben ser óptimos, asegurando una buena experiencia de usuario.

RNF-08: Se debe implementar control de versiones para mantener el código organizado y actualizado.

RNF-09: La documentación del código y del sistema debe mantenerse actualizada para facilitar su mantenimiento.

**Requerimientos del sistema**

Requerimientos de Hardware

Servidor:

Procesador: Intel Core i5 o superior.

Memoria RAM: Mínimo 8 GB (recomendado 16 GB para entornos productivos).

Almacenamiento: SSD de al menos 500 GB.

Conectividad: Red estable con ancho de banda adecuado para soportar múltiples usuarios simultáneos.

Cliente (Usuario final):

Procesador: Intel Core i3 o equivalente.

Memoria RAM: Mínimo 4 GB.

Almacenamiento: 20 GB disponibles.

Navegador web actualizado (Chrome, Firefox, Edge o Safari).

Requerimientos de Software

Backend:

Lenguaje: Java 17 o superior.

Framework: Spring Boot con módulos Spring MVC, Spring Validecion y Spring Data JPA.

Base de Datos: MySQL, PostgreSQL o H2 (para pruebas).

Servidor de Aplicaciones: Spring Boot.

Control de Versiones: Git/GitHub.

Frontend:

Motor de plantillas: Thymeleaf.

Tecnologías: HTML5, CSS3, JavaScript (con Bootstrap para estilos responsivos).

Entorno de Desarrollo:

IDE recomendado: IntelliJ IDEA

Sistema operativo compatible: Windows, Linux o macOS.

Herramienta de construcción: Maven.

Requerimientos de Seguridad

Autenticación y Autorización:

Implementación de Spring Autetication para gestionar usuarios y roles.

Arquitectura de la Aplicación: MVC (Modelo-Vista-Controlador)

La aplicación sigue la arquitectura MVC (Modelo-Vista-Controlador), que permite separar la lógica de negocio, la interfaz de usuario y el acceso a datos, facilitando la escalabilidad y el mantenimiento del sistema.

1. Capa de Presentación (Vista) - Frontend

Tecnologías utilizadas:

HTML5, CSS3, Bootstrap: Para el diseño de la interfaz de usuario.

Thymeleaf: Motor de plantillas para renderizar contenido dinámico desde el backend.

JavaScript (opcional): Para mejorar la interactividad de la aplicación.

Funcionalidad:

Muestra las páginas web con los datos procesados por el backend.

Permite la interacción del usuario con la aplicación a través de formularios y botones.

Recibe datos de los controladores y los presenta de manera estructurada.

2. Capa de Negocio (Controlador y Servicios) - Backend

Tecnologías utilizadas:

Spring Boot: Framework principal para la gestión del backend.

Spring MVC: Para la gestión de controladores y enrutamiento de solicitudes.

Spring Security: Para la autenticación y autorización de usuarios.

Componentes principales:

Controladores (Controllers): Manejan las solicitudes HTTP y coordinan la comunicación entre la vista y los servicios.

Servicios (Services): Implementan la lógica de negocio y actúan como intermediarios entre los controladores y la capa de persistencia.

DTOs (Data Transfer Objects): Se usan para transferir datos entre capas de manera segura y eficiente.

3. Capa de Persistencia (Modelo) - Base de Datos

Tecnologías utilizadas:

Spring Data JPA: Para la gestión eficiente de la base de datos.

Hibernate: Implementación de JPA para el mapeo objeto-relacional (ORM).

MySQL / PostgreSQL: Base de datos para almacenar la información.

Componentes principales:

Entidades (Entities): Representan las tablas de la base de datos y están mapeadas con anotaciones JPA.

Repositorios (Repositories): Interfaces que permiten realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre la base de datos.

Consultas personalizadas: Métodos que permiten obtener información específica a partir de la base de datos.

Flujo de Datos en la Arquitectura MVC

El usuario realiza una solicitud HTTP (ej. consulta de recetas).

El controlador recibe la solicitud y la redirige al servicio correspondiente.

El servicio ejecuta la lógica de negocio y consulta la base de datos a través del repositorio.

El repositorio obtiene los datos y los envía de vuelta al servicio.

El servicio procesa los datos y los devuelve al controlador.

El controlador envía los datos a la vista (HTML + Thymeleaf) para mostrarlos al usuario.

Beneficios de la Arquitectura MVC en la Aplicación

Separación de responsabilidades: Facilita el mantenimiento y escalabilidad del sistema.

Reutilización de código: Cada capa puede ser mejorada o modificada sin afectar a las demás.

Mejor experiencia de usuario: Gracias a la integración de Thymeleaf con HTML.

Seguridad y eficiencia: Uso de Spring Security y JPA para proteger datos y optimizar el acceso a la base de datos.

Implementación: Registro de Usuario o Nutricionista, Subida de Dietas y Recetas

1. Registro de Usuario o Nutricionista

Requisitos

El sistema debe permitir que tanto los usuarios generales como los nutricionistas se registren en la plataforma. Los nutricionistas tendrán privilegios adicionales para crear y gestionar dietas y recetas.

Implementación

Capa de Presentación (Vista):

Formulario de Registro: Se diseñó un formulario de registro en HTML con Thymeleaf donde los usuarios ingresan su nombre, correo electrónico, contraseña y tipo de cuenta (usuario o nutricionista).

Validaciones del formulario (nombre, correo, contraseña segura) tanto del lado del cliente (JavaScript) como del servidor (Spring Validation).

El tipo de cuenta se maneja mediante un campo de selección (dropdown) para elegir entre Usuario y Nutricionista.

Capa de Negocio (Controlador y Servicio):

Controlador de Registro:

El controlador recibe los datos del formulario y valida que el correo no esté ya registrado en la base de datos.

Si la validación es exitosa, los datos se pasan al servicio, donde se encripta la contraseña utilizando BCrypt y se guarda en la base de datos.

Si el usuario es un nutricionista, el sistema asigna un rol específico que le permitirá acceder a funcionalidades adicionales (como crear dietas y recetas).

Servicio de Registro:

El servicio se encarga de crear el nuevo usuario o nutricionista, asignando roles y guardando los datos en la base de datos.

Utiliza BCrypt para encriptar la contraseña antes de guardarla de manera segura.

Capa de Persistencia (Modelo):

Entidad Usuario y Nutricionista:

Se creó una clase Usuario que contiene atributos comunes como nombre, correo y contraseña.

La clase Nutricionista hereda de Usuario y agrega atributos específicos como especialidad o certificación.

Ambas entidades están mapeadas a la base de datos mediante JPA.

Repositorio de Usuario:

El repositorio permite las operaciones CRUD para las entidades Usuario y Nutricionista, como guardar y buscar por correo electrónico.

Tecnologías utilizadas:

Spring Security: Para gestionar la autenticación y autorización.

BCrypt: Para encriptar las contraseñas.

Thymeleaf y HTML: Para la creación de formularios de registro.

2. Subir Dietas y Recetas (Solo Nutricionistas)

Requisitos

Los nutricionistas deben poder crear y gestionar dietas y recetas, las cuales estarán disponibles para los usuarios que sigan sus planes.

Implementación

Capa de Presentación (Vista):

Formulario de Dieta y Receta:

Se diseñaron formularios en HTML con Thymeleaf donde los nutricionistas pueden ingresar el nombre de la dieta, la descripción, los alimentos y las recetas asociadas.

Para la receta, se permiten campos como el nombre de la receta, ingredientes, instrucciones y valor nutricional.

Lista de Dietas y Recetas:

Los nutricionistas pueden ver y gestionar las dietas y recetas que han creado, con opciones para editarlas o eliminarlas.

Capa de Negocio (Controlador y Servicio):

Controlador de Dietas y Recetas:

Los controladores reciben las solicitudes de creación, edición y eliminación de dietas y recetas.

La lógica de negocio asegura que solo los nutricionistas puedan acceder a estas funcionalidades, utilizando roles asignados durante el registro.

Servicio de Dietas y Recetas:

El servicio procesa los datos recibidos de los formularios y guarda las dietas y recetas en la base de datos.

También proporciona métodos para obtener todas las dietas y recetas de un nutricionista específico.

Capa de Persistencia (Modelo):

Entidad Dieta y Receta:

La clase Dieta tiene atributos como nombre, descripción y lista de recetas asociadas.

La clase Receta contiene atributos como nombre de receta, ingredientes, instrucciones y valores nutricionales.

Ambas entidades están relacionadas con el nutricionista a través de una clave foránea (relación de uno a muchos).

Repositorio de Dietas y Recetas:

Utilizamos Spring Data JPA para gestionar las operaciones CRUD relacionadas con dietas y recetas.

El repositorio proporciona métodos como save(), findByNutricionista(), deleteById().

Tecnologías utilizadas:

Spring Data JPA: Para la gestión de las dietas y recetas en la base de datos.

Thymeleaf y HTML: Para la creación de formularios para subir dietas y recetas.

Flujo de Datos:

Registro de Usuario o Nutricionista:

El usuario o nutricionista completa el formulario de registro y lo envía al controlador.

El controlador valida los datos y el servicio guarda la información del usuario o nutricionista en la base de datos.

Subida de Dietas o Recetas (Solo Nutricionistas):

El nutricionista accede a la interfaz de administración de dietas y recetas.

Completa el formulario de creación o edición y lo envía al controlador.

El servicio guarda la dieta o receta en la base de datos y la asocia con el nutricionista correspondiente.

Visualización de Dietas y Recetas:

Los usuarios pueden ver las dietas y recetas subidas por nutricionistas desde su interfaz, mientras que los nutricionistas gestionan el contenido desde su panel de administración.

Resumen de Tecnologías

Spring Boot y Spring MVC: Para gestionar las rutas y lógica de negocio.

Spring Security: Para autenticar y autorizar usuarios y nutricionistas.

Spring Data JPA: Para interactuar con la base de datos y gestionar dietas y recetas.

Thymeleaf y HTML: Para la creación de formularios dinámicos de registro y gestión de dietas y recetas.