Nutrition Management System

Juan Sebastian Molina Escobar-28801
Jaider Fernando Aldana Calderon-22157

Introducción

El presente documento otorga a los usuarios un breve conocimiento y entendimiento sobre el sistema para la gestión nutricional. También presenta el estudio realizado para poder desarrollar el sistema deseado, de igual manera dar a conocer los métodos para almacenar la información.

Alcance del proyecto

Este sistema busca ser una herramienta que le permita a los nutricionistas llevar un registro tanto de sus pacientes, dietas y planes alimenticios adaptables. De igual manera permitirle a estos mismo realizar modificaciones a medida que pasa el tiempo y sea necesario.

Funcionalidad del sistema

Este código Java constituye una interfaz de consola para gestionar información relacionada con pacientes, dietistas, planes de dieta y comidas. Utiliza operaciones CRUD para interactuar con archivos CSV y permite realizar acciones como registrar pacientes, dietistas, crear planes de dieta y comidas, visualizar información y eliminar registros. La aplicación almacena y manipula los datos a través de servicios y manipuladores de archivos CSV.

Métodos implementados

1. Método `main`: Inicializa la aplicación, lee datos desde archivos CSV, muestra el menú principal y solicita acciones al usuario. Guarda los datos al salir.

2. Método `displayMenu`: Muestra el menú principal con diversas opciones para la interacción del usuario. Utiliza una declaración `switch` para ejecutar acciones correspondientes según la elección del usuario.

3. Método `registerPatient`: Captura la entrada del usuario para registrar un nuevo paciente, incluyendo detalles como nombre, edad, peso, altura y condiciones preexistentes. Crea un nuevo objeto `Patient` y lo añade al servicio de pacientes.

4. Método `registerDietitian`: Recoge la entrada del usuario para registrar un nuevo dietista, incluyendo nombre y especialidad. Crea un nuevo objeto `Dietitian` y lo añade al servicio de dietistas.

5. Método `createDietPlan`: Guía al usuario a través de la creación de un nuevo plan de dieta seleccionando un paciente y un dietista. Captura detalles como calorías diarias, distribución de macronutrientes y recomendaciones específicas. Crea un nuevo objeto `DietPlan` y lo añade al servicio de planes de dieta.

6. Método `createMeal`: Toma la entrada del usuario para crear una nueva comida, incluyendo detalles como nombre, macronutrientes, calorías y hora del día. Crea un nuevo objeto `Meal` y lo añade al servicio de comidas.

7. Método `displayPatients`: Obtiene y muestra información sobre todos los pacientes registrados.

8. Método `displayDietitians`: Obtiene y muestra información sobre todos los dietistas registrados.

9. Método `displayDietPlans`: Obtiene y muestra información sobre todos los planes de dieta creados.

10. Método `displayMeals`: Obtiene y muestra información sobre todas las comidas creadas.

11. Método `saveData`: Escribe el estado actual de pacientes, dietistas, planes de dieta y comidas en archivos CSV, garantizando la persistencia de datos.

12. Método `deletePatient`: Toma la entrada del usuario para un ID de paciente y elimina al paciente correspondiente del sistema.

13. Método `deleteDietitian`: Toma la entrada del usuario para un ID de dietista y elimina al dietista correspondiente del sistema.

14. Método `deleteDietPlan`: Toma la entrada del usuario para un ID de plan de dieta y elimina el plan de dieta correspondiente del sistema.

15. Método `deleteMeal`: Toma la entrada del usuario para el nombre de una comida y elimina la comida correspondiente del sistema.

Problemas encontrados y soluciones

Problemas Posibles:

1. Entrada de Usuario:
   - Descripción Breve: Errores en la entrada del usuario, formato incorrecto o datos inesperados.
   - Solución: Implementar validación de entrada para garantizar datos correctos.

2. Manejo de Excepciones:
   - Descripción Breve: Problemas con excepciones, como divisiones por cero o lectura/escritura de archivos.
   - Solución: Implementar un manejo adecuado de excepciones para evitar fallas inesperadas.

3. Persistencia de Datos:
   - Descripción Breve: Riesgos de pérdida o corrupción de datos al leer o escribir en archivos CSV.
   - Solución: Implementar una persistencia cuidadosa y validar el formato de datos.

4. Sincronización de Datos:
   - Descripción Breve: Problemas con la sincronización de datos entre múltiples usuarios.
   - Solución: Implementar control de concurrencia para evitar conflictos de datos.

5. Memoria y Rendimiento:
   - Descripción Breve: Problemas de rendimiento o uso excesivo de memoria.
   - Solución: Optimizar estructuras de datos y algoritmos según sea necesario.

6. Actualización de Dependencias:
   - Descripción Breve: Problemas con bibliotecas desactualizadas.
   - Solución: Mantener dependencias actualizadas para corregir errores y mejorar seguridad.

7. Pruebas Insuficientes:
   - Descripción Breve: Falta de pruebas exhaustivas.
   - Solución: Implementar pruebas unitarias e integrales para detectar y corregir errores.

8. Usabilidad y Experiencia del Usuario:
   - Descripción Breve: Problemas con la usabilidad y la satisfacción del usuario.
   - Solución: Realizar pruebas de usuario y obtener retroalimentación constante para mejorar.

9. Escalabilidad:
   - Descripción Breve: Problemas de rendimiento con el aumento de datos.
   - Solución: Diseñar la aplicación considerando posibles aumentos en la carga de trabajo.

10. Seguridad:
  - Descripción Breve: Riesgos de vulnerabilidades y exposición de datos sensibles.
  - Solución: Realizar evaluaciones de seguridad y aplicar prácticas de desarrollo seguro.

Soluciones Breves:

1. Validación de Entrada:
  - Utilizar funciones de validación para asegurar datos correctos.

2. Manejo de Excepciones:
  - Implementar bloques `try-catch` para gestionar excepciones.

3. Persistencia de Datos:
  - Validar formatos de datos y asegurar lectura/escritura segura.

4. Sincronización de Datos:
  - Emplear control de concurrencia para datos compartidos.

5. Memoria y Rendimiento:
  - Optimizar algoritmos y estructuras de datos.

6. Actualización de Dependencias:
  - Mantener bibliotecas actualizadas regularmente.

7. Pruebas Insuficientes:
  - Desarrollar y ejecutar pruebas unitarias e integrales.

8. Usabilidad y Experiencia del Usuario:
  - Obtener retroalimentación del usuario y ajustar la interfaz.

9. Escalabilidad:
  - Diseñar la aplicación considerando crecimiento futuro.

10. Seguridad:
    - Realizar evaluaciones de seguridad y seguir buenas prácticas de desarrollo seguro.

Futuras mejoras y actualizaciones

1. Interfaz Gráfica de Usuario (GUI): Considera desarrollar una interfaz gráfica de usuario para hacer la aplicación más amigable y accesible, especialmente para usuarios no técnicos.

2. Autenticación y Seguridad: Implementa un sistema de autenticación para proteger la información sensible. Asegúrate de que solo usuarios autorizados puedan acceder y modificar datos críticos.

3. Base de Datos Relacional: Migrar a una base de datos relacional podría mejorar la eficiencia y escalabilidad de la aplicación en comparación con el uso de archivos CSV. Esto también facilita la realización de consultas más complejas.

4. Validación de Entrada: Agrega una capa de validación para asegurarte de que los datos ingresados por el usuario sean válidos y seguros. Evitará problemas como entradas incorrectas o maliciosas.

5. Historial y Auditoría: Implementa un sistema de registro para realizar un seguimiento de las acciones realizadas por los usuarios. Esto facilitará la auditoría y la resolución de problemas en caso de errores.

6. Notificaciones y Alertas: Incorpora un sistema de notificaciones para alertar a los usuarios sobre cambios importantes, citas pendientes o recordatorios relacionados con la gestión de pacientes y dietas.

7. Manejo de Imágenes y Documentos: Si es relevante, permite a los usuarios adjuntar imágenes o documentos relacionados con pacientes, dietas o comidas para un seguimiento más detallado.

8. Soporte para Múltiples Idiomas: Si la aplicación se utiliza en entornos multiculturales, considera agregar soporte para múltiples idiomas para mejorar la accesibilidad.

9. Análisis y Estadísticas: Agrega funcionalidades para realizar análisis y estadísticas sobre los datos recopilados, lo que podría proporcionar información valiosa para los dietistas y profesionales de la salud.

10. Integración con Dispositivos Wearables: Explora la posibilidad de integrar la aplicación con dispositivos wearables para recopilar datos adicionales sobre la actividad física y la salud de los pacientes.

11. Compatibilidad con Formatos de Datos Estándar: Asegúrate de que la aplicación sea compatible con estándares de intercambio de datos en el campo de la salud, lo que facilita la integración con sistemas externos.

12. Mejora de la Experiencia del Usuario (UX): Realiza mejoras continuas en la interfaz y la experiencia del usuario con base en comentarios de los usuarios para hacer la aplicación más intuitiva y fácil de usar.

13. Actualizaciones Legales y de Cumplimiento: Mantente actualizado con las regulaciones y normativas en el ámbito de la salud para garantizar el cumplimiento legal y la privacidad de los datos del paciente.

Diagrama de clases