Instituto Tecnológico de Costa Rica

Tarea Corta: Prototipo

Curso: Bases de datos

Grupo:

GR-01

Estudiantes:

Henry Andrés Castro Moreno (2022026502)

Jose Ignacio Rivera Mora (2022227827)

Eder Jose Vega Suazo (2021067844)

Christopher David Rodríguez Cordero (2022040771)

Profesor:

Marco Rivera Meneses

Descripción de métodos.

1. **Métodos de Autenticación**

   RegistrarUsuario(string tipoUsuario, Usuario usuario)

   - Función que registra un nuevo usuario (nutricionista o cliente) en el sistema.
   - **Entrada:** Tipo de usuario (Nutricionista o Cliente) y un objeto que contiene los datos del usuario.
   - **Salida:** Confirmación del registro o un mensaje de error en caso de datos incompletos o duplicados.
   - **Procesos clave:** Encriptación de la contraseña utilizando el algoritmo MD5 antes de guardar los datos.

   IniciarSesion(string email, string password)

   - Método para autenticar a un usuario mediante su correo electrónico y contraseña.
   - **Entrada:** Correo y contraseña del usuario.
   - **Salida:** Objeto de usuario autenticado o mensaje de error si las credenciales no coinciden.
   - **Procesos clave:** Validación de contraseñas encriptadas.

2. **Métodos de Gestión de Productos**

   AgregarProducto(Producto producto)

   - Permite a los nutricionistas registrar un nuevo producto en el sistema.
   - **Entrada:** Objeto del producto con su descripción y valores nutricionales.
   - **Salida:** Confirmación de registro o mensaje indicando errores en los datos proporcionados.
   - **Procesos clave:** Validación de unicidad del código de barras y almacenamiento temporal del producto en espera de aprobación.

   ListarProductosPendientes()

   - Recupera la lista de productos registrados por nutricionistas que aún no han sido aprobados.

- **Salida:** Lista de productos pendientes de validación.

AprobarProducto(int idProducto)

- Método exclusivo para el administrador que permite aprobar un producto.
- **Entrada:** Identificador único del producto.
- **Salida:** Confirmación de aprobación.

3. **Métodos de Asociación Nutricionista-Paciente**

BuscarCliente(string criterio)

- Permite al nutricionista buscar un cliente mediante su nombre, correo o ID.
- **Entrada:** Criterio de búsqueda (cadena parcial o completa).
- **Salida:** Lista de coincidencias o mensaje indicando que no se encontraron resultados.

AsociarPaciente(int idNutricionista, int idPaciente)

- Método para asociar a un cliente como paciente de un nutricionista.
- **Entrada:** Identificadores únicos del nutricionista y del cliente.
- **Salida:** Confirmación de asociación.

4. **Métodos de Generación de Reportes**

GenerarReporteCobro()

- Genera un reporte con el monto que debe cobrarse a cada nutricionista basado en su tipo de suscripción y número de pacientes asociados.
- **Salida:** Reporte en formato visual (pantalla) y opción de exportarlo como archivo PDF.
- **Procesos clave:** Cálculo automático de descuentos según el tipo de suscripción.

5. **Métodos de Persistencia de Datos**

   GuardarEnXML(string nombreArchivo, object datos)

   - Método genérico para guardar datos en un archivo XML.
   - **Entrada:** Nombre del archivo y datos a guardar.
   - **Salida:** Archivo XML actualizado.

   LeerDesdeXML<T>(string nombreArchivo)

   - Método genérico para leer datos desde un archivo XML.
   - **Salida:** Objeto deserializado con los datos almacenados.

   GuardarEnJSON(string nombreArchivo, object datos)

   - Método genérico para guardar datos en un archivo JSON.
   - **Entrada:** Nombre del archivo y datos a guardar.
   - **Salida:** Archivo JSON actualizado.

   LeerDesdeJSON<T>(string nombreArchivo)

   - Método genérico para leer datos desde un archivo JSON.
   - **Salida:** Objeto deserializado con los datos almacenados.

6. **Métodos Auxiliares**

   ValidarDatosUsuario (Usuario usuario)

   - Verifica que todos los campos obligatorios del usuario estén completos y cumplan con los formatos requeridos.
   - **Salida:** Verdadero si los datos son válidos o una lista de errores encontrados.

   EncriptarMD5(string textoPlano)

   - Método para encriptar contraseñas usando el algoritmo MD5.
   - **Entrada:** Texto plano (contraseña).
   - **Salida:** Contraseña encriptada como cadena hexadecimal.

Descripción de estructura de datos

Autenticación (Log In):

- Método para validar el usuario y contraseña encriptados usando MD5.
- Métodos auxiliares para verificar si las credenciales existen en el sistema.

Registro:

- Métodos para registrar nutricionistas y clientes, validando datos obligatorios.
- Encriptación de contraseñas y almacenamiento de la información en formato XML.

Gestión de productos:

- Métodos para agregar productos, con validaciones de formato y almacenamiento temporal hasta la aprobación del administrador.

Asociación de pacientes:

- Métodos para buscar clientes registrados y vincularlos con un nutricionista.

Generación de reportes:

- Métodos para calcular los montos a cobrar a los nutricionistas basados en su tipo de pago y cantidad de pacientes.
- Función para exportar los reportes en formato PDF o enviarlos a imprimir.

Descripción de arquitectura desarrollada

La arquitectura del proyecto se diseñó bajo un enfoque modular y basado en capas, asegurando una estructura organizada, escalable y fácil de mantener. Cada módulo tiene una responsabilidad específica, lo que facilita la integración y la extensión del sistema. A continuación, se describe en detalle la arquitectura utilizada:

Componentes Principales

1. **Gestión de Usuarios (Nutricionistas y Clientes):**
    - Registro de usuarios con validaciones de datos requeridos.
    - Autenticación de credenciales mediante contraseñas encriptadas con MD5.

- Asociaciones entre nutricionistas y pacientes.

2. **Gestión de Productos:**
    - Almacenamiento temporal de productos registrados por los nutricionistas.
    - Validaciones de los datos del producto (por ejemplo, código de barras único).
    - Aprobación pendiente por parte del administrador antes de hacerlos visibles globalmente.
3. **Gestión de Cobros y Reportes:**
    - Cálculo de montos a cobrar a los nutricionistas, considerando su tipo de plan de pago.
    - Generación de reportes organizados por tipo de pago y exportación en formato PDF.
4. **Persistencia de Datos:**
    - Uso de **archivos XML** para almacenar listas de usuarios, pacientes y productos.
    - Uso de **archivos JSON** para manejar configuraciones y parámetros del sistema.
    - Los datos son estructurados jerárquicamente para facilitar su lectura y escritura.

Problemas encontrados

Descripción del Problema: Conexión con la API

En el proyecto, se presentó un problema intermitente relacionado con la conectividad entre el cliente y la API, lo que afecta la ejecución fluida de ciertas funcionalidades clave como el registro, la autenticación de usuarios y la consulta de datos externos.

El error se manifiesta en las siguientes circunstancias:

- Solicitudes HTTP que se envían al servidor generan un código de estado 504 (Gateway Timeout) de manera aleatoria.
- En otros casos, se reciben códigos de error 500 (Internal Server Error), lo que indica fallos en el servidor.
- Latencias altas en la respuesta de la API, que impactan la experiencia de usuario.

Conclusiones

Cumplimiento de Requisitos Funcionales:
El proyecto logró implementar la gran mayoría de funcionalidades indicadas en el enunciado, abarcando las vistas de Nutricionista, Cliente/Paciente y Administrador. Cada módulo permite la gestión de datos de manera eficiente y asegura una experiencia de usuario fluida.

Escalabilidad y Portabilidad:
El uso de archivos XML y JSON como base de datos permitió simplificar el almacenamiento y lectura de datos. Si bien esto fue suficiente para el alcance inicial del proyecto, estas soluciones pueden presentar limitaciones en sistemas con gran cantidad de datos o múltiples usuarios concurrentes.

Interfaz de Usuario:
Las interfaces desarrolladas ofrecen una experiencia intuitiva, asegurando que nutricionistas, pacientes y administradores puedan navegar y utilizar el sistema sin necesidad de capacitación técnica.

Trabajo Colaborativo:
La planificación y el seguimiento detallado de tareas a través de reuniones y bitácoras garantizó una comunicación efectiva entre los integrantes del equipo, lo cual fue clave para cumplir con los plazos establecidos.

Referencias Bibliografías

BillWagner. (s. f.). *Guía de C#: lenguaje administrado de .NET*. Microsoft Learn. https://learn.microsoft.com/es-es/dotnet/csharp/

Harness the power of PDF. (s. f.). iTextpdf. https://itextpdf.com/