Taller de Programación II

Desarrollo de una aplicación de gestión para consultorios médicos

Grupo 6

Integrantes : Leonardo Romero Cabrera
Esteban Cádiz Leyton
Sebastián Martínez Silva
Vicente Alarcón Ibarra

Docente/s : Braulio Quiero Hernández

Curso : Taller de Programación II

Fecha : 02/12/2024

Índice de Contenidos

Introducción

En la actualidad, la digitalización se ha vuelto esencial en diversos ámbitos, incluida la gestión de información en recintos de salud, como los consultorios médicos. Este proyecto tiene como objetivo desarrollar una aplicación básica que permita optimizar las operaciones de un consultorio, reemplazando el uso tradicional por una solución digital confiable y accesible.

Utilizando Avalonia y C# Como principales herramientas de desarrollo, la aplicación integra características como manejo de roles de usuarios (médico y secretaria), registro y búsqueda de pacientes, y almacenamiento de datos en formato JSON. Este enfoque no solo mejora la accesibilidad y organización de la información, sino que también sienta las bases para una futura escalabilidad del sistema.

El presente informe documenta las etapas del proyecto, desde el análisis y diseño hasta la construcción e implementación de las funcionalidades. Además, se incluyen diagramas UML y detalles técnicos relevantes para mostrar la estructura de la aplicación.

Propósito

El propósito de este documento es presentar de manera detallada el desarrollo de una aplicación que optimice la gestión de un consultorio médico. La aplicación está diseñada para digitalizar los procesos, facilitando la organización, el acceso y la actualización de la información del paciente.

Objetivos

El objetivo principal de este proyecto es crear una aplicación para la gestión de un consultorio médico, utilizando Avalonia y C# para el desarrollo de la interfaz y la lógica. La aplicación tiene como propósito digitalizar y optimizar la gestión de la información de los pacientes, citas y fichas médicas. Se implementarán funcionalidades esenciales como inicio de sesión, gestión de usuarios (Secretaria y Médico), y almacenamiento de datos mediante archivos JSON. Con esto, se busca mejorar la eficiencia en el manejo de datos, facilitando el acceso a la información en el consultorio.

Organización del documento

El documento se estructura de la siguiente manera:

- Introducción: Contextualización del proyecto y su propósito.
- Definición del problema: Descripción de la necesidad del proyecto y cómo se aborda el problema.
- Interesados: Identificación de los usuarios clave en el sistema y sus necesidades.
- Funciones del Producto: Detalles de las funciones implementadas en la aplicación, organizadas por módulos.
- Elaboración y Análisis: Descripción de la fase de desarrollo y los procesos analizados para el diseño del sistema.
- Diagramas de Caso de Usos y Clases: Diagramas UML que detallan la estructura del sistema.
- Construcción y Requisitos Futuros: Explicación del proceso de construcción, desafíos y mejoras futuras.
- Código Fuente y Conclusiones: Resumen de los logros del proyecto, dificultades encontradas y la valoración general.

Definición del problema

Hoy en día se sigue utilizando en algunos centros con atención al cliente el lápiz y papel generando una gran pérdida de espacio y tiempo si es que de búsqueda se tratase, es por eso que optan por medios más novedosos y eficientes tales como, almacenar la misma información, pero digital agilizando la búsqueda y reduciendo el espacio empleado para almacenarlos, por el mismo motivo se demanda aplicaciones hechas a medida para las empresas o clientes.

Descripción del problema

Un consultorio necesita tratar la información de sus pacientes desde los datos personales del mismo, el médico asignado y sus respectivas fichas médicas. La solución está en la digitalización de toda esa información mediante un aplicación para los usuarios, la cual tendrá una interfaz amigable para lograr agilizar la interacción, relleno y/o búsqueda de estos datos.

Interesados

Dentro de la estructura de las clínicas y/o consultorios quien siempre se está presente y son necesarios a cada momento son las secretarias ellas se encargan de asistir y derivar al paciente a un médico específico, a su vez el médico de turno necesita interactuar con la plataforma para consultar los datos del paciente y/o rellenar su ficha médica posterior atención con el médico

Nombre	Descripción	Interés
Secretaria	Personal Encargado de asistir al paciente	Reservar una hora para un paciente rellenar datos del paciente Consulta de paciente
Medico	Personal Especializado en la atención y diagnóstico del paciente	Rellenar ficha médica, utilizar elementos desechables y entregar medicamentos

Funciones del Proyecto

Módulo 1

Descripción : Funciones internas de los usuarios necesarias para ingresar al sistema

Función 1	void LogIn()
Personal Involucrado	Medico : Iniciar sesión perfil de Médico Secretaría :Iniciar sesión perfil de Secretaria
Precondiciones	El Usuario estará en la ventana de inicio de Secretaria o médico donde ingresara 2 parámetros Username y password en caso de no ingresar nada estos se tomarán como null
Entrada	String Username , string password
Flujo Básico	Mediante los parámetros de entrada y tras deserializar el archivo json que almacena los usuarios autorizados para ingresar al perfil deseado, compara los datos ingresados con los perfiles autorizados
Extensiones	Si los datos ingresados corresponden a un perfil autorizado este accede a las funciones de su perfil Si los datos son incorrectos llama a una ventana que se encarga de advertir en error
Salidas	“Función void no retorna nada”

Función 2	Void LogOff
Personal Involucrado	Secretaria : Cerrar sesion de Secretaria y guardar cambios
Precondiciones	Se asume que el secretario ingreso sesion correctamente e interactúa con los pacientes
Entrada	No tiene entradas
Flujo Básico	Al cerrar sesión serializa y guarda todos los datos recolectados que estaban a disposición del perfil “Box, sala de espera, Seguridad , Limpieza y Médicos
Extensiones	No tiene extensiones aparte del Flujo Básico
Salidas	“Función void sin salida por consola “Función regresa a la ventana de selector de perfil”

Función 3	Void RecuperarJsonLogIn
Personal Involucrado	Medico : Función para la búsqueda específica de las credenciales del médico
Precondiciones	El usuario se encuentra en la ventana de login médico y ya ingresó un nombre en la casilla
Entrada	String Username
Flujo Básico	La función rescata el archivo json alojado en el “nombre ingresado" .json para luego deserializar y almacenar los datos en variables propias de la clase, luego almacena la dirección del archivo json y llama a 2 funciones para recuperar más datos.
Extensiones	No tiene extensiones más que el flujo básico
Salidas	“Función void sin salida de datos” Solo llama otras funciones

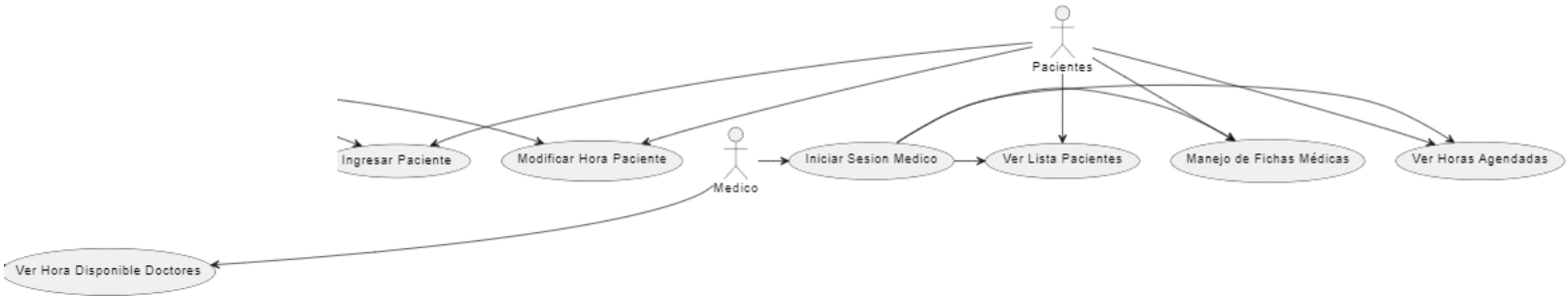
Módulo 2

Descripción: Funciones anexadas a un botón de la inferaz del usuario.

Función 1	FichaMedica
Personal Involucrado	Médico : Rellenar Ficha médica del paciente.
Precondiciones	Se ingreso el paciente y el médico está interactuando con sus datos luego de rescatarlos del json que lo contenía
Entrada	Esta función tiene como parámetro los datos ingresados en su ventana
Flujo Básico	Seleccionar Grupo Sanguíneo y almacenar en la variable de su clase cliente además almacena otros datos como alergias antecedentes y observaciones.
Extensiones	La función no tiene extensión ya que la falta de rellenar algún dato se toma como null
Salidas	Deriva al usuario a la pantalla ListaFichaWindow

Función 2	btModPaciente
Personal Involucrado	Medico : Modificar la ficha del paciente
Precondiciones	Médico hizo LogIn correctamente ademas se esta trabajando sobre un paciente ingresado previamente y seleccionado por el médico para modificar
Entrada	Datos del paciente que se desea modificar (como clase)
Flujo Básico	Se selecciona el paciente y se despliega la ventana para modificar los datos del paciente y su ficha
Extensiones	En caso de no seleccionar un paciente o no se logre aceptar el parámetro este mostrará una ventana de error
Salidas	Sin Salidas por consola funcion void Salida por ventanas de Error en caso de no seguir el flujo básico Salida mostrar ventana ModFichasWindows en caso de seguir el flujo básico son problemas

Diagramas de Caso de Usos



Elaboración

En esta sección se explica el proceso de desarrollo de la aplicación, desde la identificación de los requisitos hasta la implementación final. Se comenzó con el análisis de sistemas similares para definir las funcionalidades necesarias, como la gestión de pacientes, citas y fichas médicas. A partir de esto, se diseñaron las clases clave (Usuario, Paciente, Cita, Ficha Médica) y se creó una interfaz de usuario intuitiva, adaptada a las necesidades de secretarias y médicos.

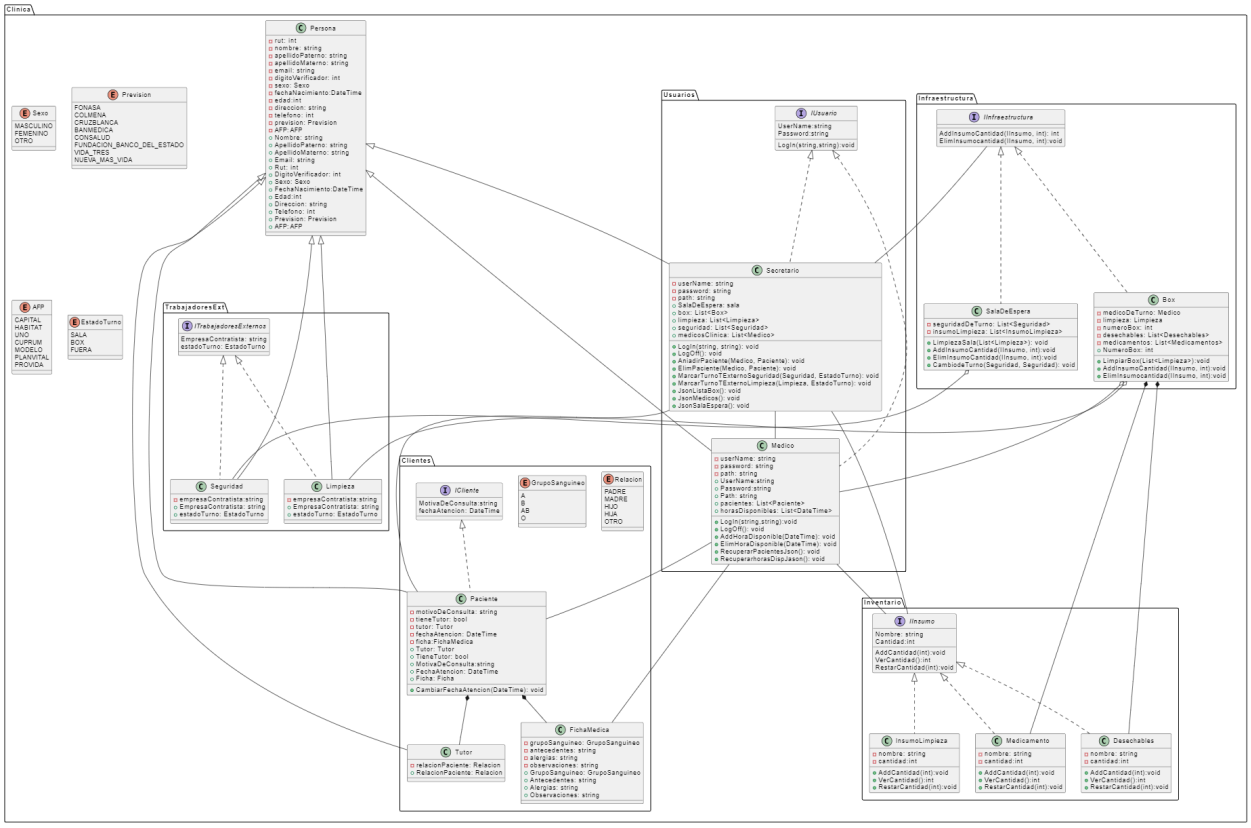
La aplicación se estructuró utilizando archivos JSON para almacenar los datos, lo que permitió una solución simple y eficiente sin la necesidad de una base de datos compleja. Se implementaron funciones esenciales como inicio de sesión, gestión de citas y acceso a fichas médicas, logrando una herramienta funcional para el consultorio.

Análisis

El análisis del proyecto se centró en identificar los requisitos clave para la gestión de un consultorio médico digitalizado. La principal necesidad era optimizar el manejo de la información de los pacientes, las citas y las fichas médicas. A partir de esta necesidad, se definieron las clases esenciales del sistema, como Usuario, Paciente, Cita y Ficha Médica, que permiten organizar la información de manera estructurada y eficiente.

Se investigaron plataformas similares para comprender las mejores prácticas y las estructuras más comunes en sistemas de gestión de consultorios médicos. Esta comparación ayudó a definir la arquitectura del sistema y las funcionalidades necesarias, como la creación de perfiles de usuario (Secretaria y Médico), la asignación de citas y el acceso a las fichas médicas.

Diagramas de Clases



Información UML

TrabajadoresExt : Trabajadores externos a la estructura funcional del sistema (Seguridad y Limpieza).

Clientes : Toda la información relacionada al paciente y su posterior diagnóstico.

Usuarios : Personal con credenciales para interactuar con la aplicación (Medicos y Secretarios).

Inventario : Agrupación de elementos utilizables dentro de las consultas clínicas (Medicamentos y desechables).

Infraestructura : Agrupación de habitaciones (Sala de Espera y Boxes)

Construcción

En esta sección se detalla el proceso de construcción de la aplicación, abordando los principales componentes y tecnologías utilizadas en su desarrollo. Se explicarán las decisiones tomadas en cuanto a la interfaz de usuario, la estructura interna del sistema y el manejo de los datos. También se incluirán las funcionalidades clave implementadas, como el inicio de sesión, la gestión de perfiles de usuario (Secretaria y Médico), y la organización de la información de los pacientes mediante archivos JSON. Además, se destacarán los desafíos encontrados durante la implementación y cómo se resolvieron para lograr una aplicación funcional.

Interfaz de usuario

La interfaz de usuario de la aplicación está diseñada para ser intuitiva y fácil de usar, adaptándose a las necesidades de secretarias y médicos. En la pantalla de inicio, los usuarios seleccionan su perfil y luego acceden a una ventana de inicio de sesión para ingresar sus credenciales. Una vez autenticados, el menú principal ofrece opciones específicas para cada rol, la secretaría puede registrar pacientes, agendar citas y gestionar información, mientras que los médicos tienen acceso a las fichas médicas de los pacientes para registrar diagnósticos y tratamientos. La interfaz presenta un diseño limpio, con botones y formularios claramente etiquetados, facilitando la navegación y la realización de tareas.

Requisitos Futuros

En el futuro, se planea agregar funcionalidades como un portal para que los pacientes puedan revisar su historial médico y agendar citas, además de un perfil administrativo para gestionar usuarios y supervisar el sistema. También se busca reemplazar el almacenamiento en JSON por una base de datos más robusta, implementar notificaciones de citas, generar reportes estadísticos del consultorio, y mejorar la seguridad con contraseñas cifradas y autenticación multifactor. Estas mejoras harán que la aplicación sea más completa, eficiente y adaptable a las necesidades reales de un consultorio médico.

Código Fuente

```
private void ConfirmarH(object? sender, Avalonia.Interactivity.RoutedEventArgs e)
{
    if (PacienteEsNuevo)
    {
        //CASO PACIENTE NUEVO
        try
        {
            Medico medicoElegido = (Medico)LbDoctores.SelectedItem;
            DateTime horaElegida = ((HoraFechaDTime)LbHdisp.SelectedItem).DatetimeS;
            pac.Fechaatencion = horaElegida;
            foreach (Medico med in medicos)
            {
                if (med.NombreAp == medicoElegido.NombreAp)
                {
                    med.ElimHoraDisponible(horaElegida);
                    med.Pacientes.Add(pac);
                }
            }
            sec.MedicosClinica = medicos;//Se guardan los cambios hechos a la lista de medicos en secretario que maneja todo eso
            var menuWindow = new MenuWindow(sec);
            menuWindow.Show(); // Muestra la ventana secundaria
            this.Close();
        }
        catch (Exception ex)
        {
            ErrorWindow err = new ErrorWindow();
            err.ErrorMsg.Text = ex.Message;
        }
    } else
    {
        //CASO PARA MODIFICAR HORA DE UN PACIENTE EXISTENTE
        Medico medicoElegido = (Medico)LbDoctores.SelectedItem;
        DateTime horaElegida = ((HoraFechaDTime)LbHdisp.SelectedItem).DatetimeS;
        DateTime horaLiberada = pac.Fechaatencion;

        //Ciclo for busca medico que tiene al paciente para liberarle la hora y remover paciente
       foreach(Medico Med in medicos)
        {
            if(Med.Pacientes.Contains(this.pac))
            {
                Med.Pacientes.Remove(this.pac);
                Med.AddHoraDisponible(horaLiberada);
            }

        }


        //Ciclo que añade paciente a medico ocupando su hora disponible
        foreach (Medico med in medicos)
        {
            if (med.NombreAp == medicoElegido.NombreAp)
            {
                med.ElimHoraDisponible(horaElegida);
                med.Pacientes.Add(pac);
            }
        }
        this.sec.MedicosClinica = medicos;//Se guardan los medicos en sec
        var menuWindow = new MenuWindow(sec);
        menuWindow.Show(); // Se abre menu de nuevo
        this.Close();
    }
}
```

Lo que destaca a esta aplicación es el uso dinámico de las clases a las cuales les corresponde cierta información dependiendo de su naturaleza. En el extracto de código arriba, se puede ver una función la cual se ejecuta al presionar un botón en Avalonia. Específicamente este es el caso del ingreso o modificación de un paciente. Esto se hace mediante la clase Secretario, ya que es la sesión iniciada y luego se ingresa a la lista que tiene guardada de los médicos de la clínica. Mediante estos se accede a los pacientes, ya que estos son asignados al médico que

los atiende. Se selecciona una hora en la interfaz gráfica y se le asigna al paciente, en el caso de que se esté modificando la hora también se libera la que ya no está usada. Mediante la variable “PacienteEsNuevo”, se sabe si este es un nuevo ingreso o modificación.

Una vez se realizan todas estas asignaciones, se devuelve a la ventana del menú principal creando una nueva instancia a la que se le pasa el secretario es un estado actual después de las modificaciones. Así, al cerrar la sesión, todo lo que se haya hecho se guarda en su respectivo JSON.

Por menores

Durante la realización del proyecto, el principal desafío fue el manejo del tiempo. Dada la complejidad de implementar todas las funcionalidades requeridas, consideramos que se habría necesitado un plazo mayor, idealmente de unas cuatro semanas.

Otro aspecto complicado fue la implementación de ciertos elementos técnicos, como el uso de DataGrid, que inicialmente estaba planificado para mejorar la visualización y gestión de los datos. Sin embargo, múltiples errores durante su implementación nos llevaron a descartar tras invertir varias horas en pruebas sin éxito.

Además, fue necesario reestructurar algunas partes del proyecto en varias ocasiones, lo que generó ajustes en el diseño inicial y nos obligó a replantear ciertas funcionalidades. Estas dificultades, aunque desafiantes, nos dieron un aprendizaje significativo sobre la importancia de la planificación y la adaptación en proyectos de programación.

Conclusiones

El desarrollo de esta aplicación permitió al equipo profundizar en el uso de herramientas modernas como Avalonia y C#, además de aprender a trabajar con archivos JSON para simular una base de datos local. A pesar de las limitaciones de tiempo y los desafíos técnicos encontrados, se logró cumplir con los objetivos principales, proporcionando una solución funcional para la gestión de información en un consultorio médico.

Aunque el proyecto presenta una base sólida, quedaron pendientes algunas funcionalidades adicionales, como la integración de roles administrativos o una interfaz más completa para el manejo de pacientes. Estos elementos representan oportunidades de mejora y expansión para versiones futuras del sistema.

En conclusión, este proyecto no solo resuelve una problemática específica, sino que también sirvió como experiencia formativa para aplicar conceptos teóricos y prácticas de programación orientada a objetos en un entorno de trabajo colaborativo.

Bibliografía y anexos

Avalonia Docs | Avalonia Docs.https://docs.avaloniaui.net/. Accedido 2 de diciembre de 2024.
BillWagner. C# Guide - .NET Managed Language.
https://learn.microsoft.com/en-us/dotnet/csharp/. Accedido 2 de diciembre de 2024.

«herramienta de código abierto que utiliza descripciones textuales simples para dibujar hermosos diagramas UML.» Plantuml.com, https://plantuml.com/es/. Accedido 2 de diciembre de 2024.