



FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA DE SISTEMAS
COMPUTACIONALES

INFORME DE PROYECTO FINAL

TÍTULO: SISTEMA DE GESTIÓN DE CITAS MÉDICAS

Autores:

Alonzo Rabanal, Jhonny Rafael (100%)
Bustamante Colunche, Josue Willy (100%)

Curso:

Estructura de Datos

Docente del Curso:

ARRIBASPLATA PALOMINO, Mark Anthony

Cajamarca – Perú

2025-2

Contenido

I.	RESUMEN.....	3
II.	INTRODUCCIÓN.....	3
II.1	Motivación del proyecto.....	3
II.2	Propuestas	3
III.	ANÁLISIS DEL PROBLEMA	4
III.1	Descripción de la empresa	4
III.2	Identificación del problema	4
III.3	Antecedentes	5
III.4	Marco Teórico.....	6
	Simple:.....	6
	Doble:	6
	Pila:.....	7
	Cola.....	7
	Árbol:.....	7
III.5	Definición de objetivos	8
IV.	HERRAMIENTAS DE INGENIERÍA.....	8
V.	GENERACIÓN DE SOLUCIONES	9
V.1	Flujograma de procesos	9
VI.	METODOLOGÍA DE DESARROLLO A UTILIZAR.....	9
VIII.1	Conclusiones.....	21
VIII.2	Recomendaciones.....	21
IX.	COMPROMISO ÉTICO EN EL EJERCICIO PROFESIONAL	21
X.	REFERENCIAS O BIBLIOGRAFÍA	22
XI.	ANEXOS	23

I. RESUMEN.

El presente proyecto tiene como finalidad el desarrollo de un Sistema de Gestión de Citas Médicas para la clínica “Centro Médico San Marcos”, ubicada en la provincia de San Marcos. Este sistema fue implementado en el lenguaje de programación C#, utilizando el entorno de desarrollo Visual Studio 2022, con el objetivo de optimizar la administración de pacientes y médicos mediante estructuras de datos como listas enlazadas, colas, pilas y árboles binarios.

La propuesta busca automatizar procesos que anteriormente se realizaban de forma manual, reduciendo errores de asignación, pérdida de información y tiempos de espera. Además, permite gestionar de manera eficiente la atención médica, mostrando las colas de pacientes por especialidad, asignando médicos automáticamente y manteniendo un historial de atenciones.

De esta forma, el sistema contribuye a mejorar la organización, productividad y calidad del servicio dentro de la clínica, brindando una herramienta confiable y de fácil uso tanto para el personal médico como administrativo.

Palabras clave: sistema de gestión, citas médicas, estructuras de datos, automatización, atención médica.

II. INTRODUCCIÓN.

II.1 Motivación del proyecto

A lo largo del tiempo los hospitales y clínicas han sido lugares en los que se maneja grandes volúmenes de información relacionados con pacientes, doctores y sus respectivas atenciones médicas. Sin embargo, en muchos de ellos se sigue registrando toda esta información de maneras manual lo que puede generar errores al momento de asignar los pacientes al médico respectivo, pérdida de información y mala gestión de los recursos. Es por esto por lo que se planteó el desarrollo de un software con la finalidad de mejorar la eficiencia administrativa del hospital y a su vez contribuir a una mejor experiencia tanto para el personal médico como para los pacientes, asegurando un manejo estructurado y accesible de la información.

II.2 Propuestas

Listar las diferentes propuestas que tuvieron antes de elegir el proyecto que fue seleccionado.

Las opciones que se consideraron antes de decidir el proyecto fueron:

- Realizar un sistema de ventas para una tienda comercial.
- Realizar un sistema de registro de estudiantes y cursos.

III. ANÁLISIS DEL PROBLEMA

III.1 Descripción de la empresa

La Clínica “Centro médico San Marcos” Se encuentran en la Provincia de San Marcos, a 4 cuadras de la Plaza de armas.

Cuenta con ambientes para las especialidades de: Medicina General, Ginecología, Traumatología, Neumología, Cardiología, Pediatría, Dermatología y Gastroenterología.

La clínica es pequeña por lo que no cuenta con un sistema para gestionar las atenciones que se tienen.

III.2 Identificación del problema

La clínica “Centro médico San Marcos” que no cuenta con un sistema de gestión automatizado enfrenta diversas deficiencias que afectan su funcionamiento operativo y la calidad del servicio que brinda.

En primer lugar, la administración manual de los registros médicos incrementa el riesgo de errores humanos, como la pérdida de información, duplicidad de datos o asignaciones incorrectas entre pacientes y doctores. Además, el acceso a la información se vuelve lento y poco eficiente, dificultando la atención oportuna y el seguimiento adecuado de los tratamientos.

Asimismo, la falta de centralización de los datos impide contar con una visión completa del historial clínico del paciente, afectando la toma de decisiones médicas.

En el ámbito administrativo, se genera una mala gestión de recursos humanos y materiales, ya que no se dispone de herramientas para controlar el ingreso de los pacientes de manera automática a su cita médica en el orden que le corresponde.

Por otro lado, la comunicación entre las diferentes áreas del establecimiento se ve limitada, lo que puede provocar retrasos en la atención, insatisfacción del paciente y una disminución en la productividad del personal. En conjunto, estas deficiencias reducen la eficiencia general de la clínica y dificultan ofrecer un servicio de salud moderno, organizado y de calidad.

III.3 Antecedentes

En los últimos años, diversas investigaciones han abordado el desarrollo de sistemas informáticos orientados a optimizar la gestión en centros de salud y clínicas privadas. Estos proyectos han tenido como finalidad mejorar la atención al paciente, la administración de citas médicas y la organización de las historias clínicas mediante el uso de tecnologías web y metodologías ágiles. Los antecedentes seleccionados a continuación guardan relación directa con el presente proyecto, ya que demuestran la importancia de la digitalización y automatización de procesos clínicos, así como los beneficios obtenidos en eficiencia, accesibilidad y calidad del servicio.

- Pérez Acharte y Ponce de la Cruz (2022) tuvieron por objetivo desarrollar e implementar un sistema web para la gestión de citas médicas y hospitalización en la clínica MEDISALUD S.A.C., mediante un marco de trabajo Scrum. Como resultado, el sistema gestionó pacientes, historias clínicas, citas y hospitalización, mejorando los procesos misionales de la empresa.
- Benites Ortiz (2022) propuso la implementación de un sistema de gestión web de historias clínicas para el Centro Médico Ocupacional Sanna en Talara, motivado por el desorden en la información de admisión. Los resultados mostraron que el 70 % del personal estaba insatisfecho con el sistema previo y el 90 % aceptó favorablemente la propuesta de implementación.
- Meléndez Gárate (2023) investigó la relación entre la implementación de un sistema informático hospitalario y la gestión de la historia clínica electrónica ambulatoria en una clínica privada de Lima, encontrando que el SIH contribuye significativamente a la optimización de procesos de atención, calidad de atención y accesibilidad de los registros clínicos.
- Ruiz Ricra (2024) desarrolló un sistema de gestión dirigido a clínicas dentales independientes, abordando tanto la administración como la gestión de pacientes e inventario, y evidenciando que los odontólogos independientes afrontan deficiencias en el control de información administrativo-clínica, para lo cual la solución propuesta aportó una mejora significativa en la gestión.

- Caicedo Saavedra (2024) tuvo por objetivo desarrollar un sistema web de historias clínicas electrónicas para pacientes ambulatorios en la Clínica San Felipe, mediante metodologías ágiles XP y Scrum, con lo que reorganizó el registro clínico y mejoró la gestión de datos de atención.

III.4 Marco Teórico

Simple:

La lista enlazada es un TDA que nos permite almacenar datos de una forma organizada. En una lista enlazada, cada elemento apunta al siguiente excepto el último que no tiene sucesor y el valor del enlace es NULL. Por ello los elementos son registros que contienen el dato a almacenar y un enlace al siguiente elemento. Los elementos de una lista, suelen recibir también el nombre de nodos de la lista. Debemos tener en cuenta que en C# el puntero es una variable del tipo nodo que “apuntará” al siguiente nodo de nuestra lista, de esta manera tendremos todos los elementos de la lista debidamente enlazados.

En una lista simple podemos realizar las siguientes operaciones:

Insertar: inserta un nodo con dato x en la lista, pudiendo realizarse esta inserción al principio o final de la lista o bien en orden.

Eliminar: inserta un nodo con dato x en la lista, pudiendo realizarse esta inserción al principio o final de la lista o bien en orden.

Buscar: busca un elemento en la lista.

Localizar: obtiene la posición del nodo en la lista.

Vaciar: borra todos los elementos de la lista

Doble:

En este tipo de lista a diferencia de la simple se cuenta con 2 enlaces uno apuntando al siguiente nodo y uno al anterior. Existen 2 tipos de lista dobles:

- *Lineales: Colección de nodos, donde cada nodo tiene dos punteros, uno de ellos apuntando a su predecesor y otro a su sucesor, siendo el predecesor del primer nodo NULL y el sucesor del ultimo NULL.*
- *Circulares: En este tipo de lista doble, el puntero izquierdo del primer nodo apunta al último nodo de la lista, y el puntero derecho del último nodo apunta al primer nodo de la lista.*

Pila:

Es una lista ordenada de elementos en la que todas las inserciones y supresiones se realizan por un mismo extremo de la lista. Se le pueden añadir y retirar nuevos nodos únicamente de su parte superior, la cima de la pila. Por esta razón, se conoce una pila como una estructura de datos LIFO por last-in, first out, es decir último en entrar primero en salir.

Cuando se dice que la pila está ordenada, se quiere decir que hay un elemento al que se puede acceder primero, otro elemento al que se puede acceder en segundo lugar, un tercero, etc. No se requiere que las entradas se puedan comparar utilizando el operador “menor que” y pueden ser de cualquier tipo. Se referencia una pila mediante un apuntador al elemento superior de la misma. El miembro de enlace en el último nodo de la pila se define a NULL, para indicar que se trata de la parte inferior de la misma.

Cola.

Es una estructura de datos lineal donde los elementos se insertan por un extremo (atrás) y se eliminan por el otro extremo (delante). Esto también se le conoce como el método FIFO First In First Out (primero en entrar, primero en salir).

Árbol:

Un árbol es una estructura de datos no lineal que permite representar relaciones jerárquicas entre elementos. Está compuesto por nodos enlazados entre sí, donde el primer nodo se denomina raíz y a partir de él se ramifican los demás nodos llamados hijos. Cada nodo puede tener uno o varios nodos descendientes, y aquellos que no poseen hijos se conocen como hojas. Esta estructura se diferencia de las listas, pilas y colas porque los datos no se organizan de manera secuencial, sino jerárquica, lo que facilita representar relaciones de dependencia o clasificación. Existen diversos tipos de árboles, siendo el más común el árbol binario, en el que cada nodo puede tener como máximo dos hijos, uno izquierdo y otro derecho. Dentro de esta categoría, el árbol binario de búsqueda organiza los datos de tal forma que los valores menores se ubican en el subárbol izquierdo y los mayores en el derecho, permitiendo una búsqueda, inserción y eliminación más eficiente. En C#, los árboles suelen implementarse mediante clases y

referencias a objetos, donde cada nodo contiene un valor y punteros a sus nodos hijos, lo que permite gestionar de manera ordenada la información y optimizar procesos de búsqueda dentro de estructuras jerárquicas.

III.5 Definición de objetivos

Objetivo General.

Desarrollar un sistema en C# que asigne pacientes a médicos según su especialidad, con el fin de optimizar la atención médica y mejorar la eficiencia en la administración de recursos clínicos, además de tener un registro de todas las personas que fueron atendidas en la clínica.

Objetivos específicos

Desarrollar un algoritmo de asignación que relacione automáticamente a cada paciente con el médico adecuado según la especialidad requerida

Implementar una interfaz gráfica de usuario (GUI) que permita el registro de pacientes y médicos.

Implementar una interfaz gráfica donde se pueda observar la cola de pacientes que hay en cada especialidad.

Implementar una página donde se guarde un historial de las atenciones que se han realizado.

IV. HERRAMIENTAS DE INGENIERÍA

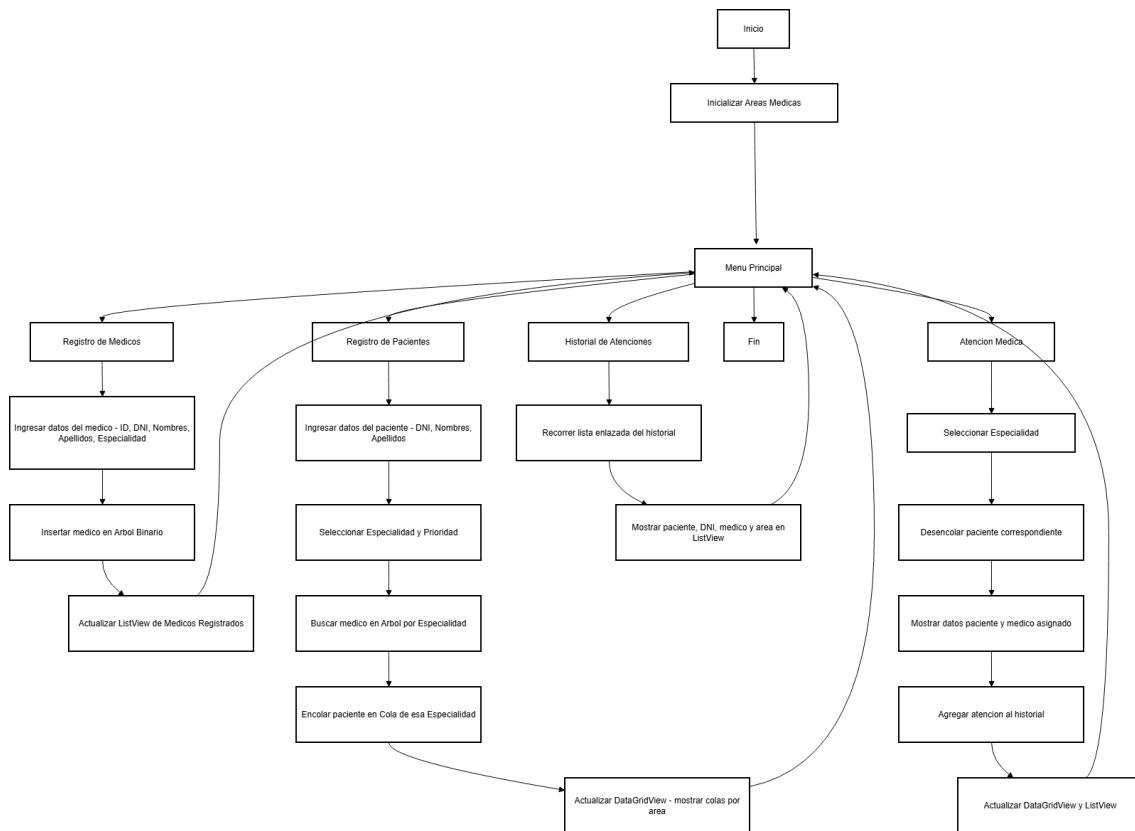
Indicar las herramientas que utilizarán:

La herramienta que se seleccionó para el desarrollo del proyecto es Visual Studio 2022, debido a que es el entorno de desarrollo integrado (IDE) utilizado en clase, por lo que es una herramienta de la cual se tiene conocimiento previo. Esta herramienta ofrece un conjunto de características que facilitan la programación, depuración y gestión de proyectos en distintos lenguajes. Siendo el C# el lenguaje que se utilizará para este proyecto.

Visual Studio 2022 destaca por su interfaz intuitiva, su compatibilidad con múltiples frameworks y su capacidad para manejar proyectos de cualquier tamaño. Permite trabajar de manera organizada. Otra de sus virtudes es la integración con Git, lo que permite a todos los miembros del grupo poder trabajar de manera colaborativa.

V. GENERACIÓN DE SOLUCIONES

V.1 Flujograma de procesos



VI. METODOLOGÍA DE DESARROLLO A UTILIZAR

Lo primero que se realizó es la creación de GUI en el cual lo primero que se agregó fue un TabControl este nos permite tener varias pestañas dentro de un mismo formulario y así tener más organizadas las acciones que se pueden realizar.

En la primera pestaña se encuentran los elementos necesarios para registrar a los médicos. Se han incorporado Label que indican los datos requeridos y TextBox donde el usuario puede ingresar la información correspondiente. Los datos que se registran son: ID del médico, DNI, Nombre, Apellido paterno, Apellido materno. Además, se incluye un ComboBox que permite seleccionar la especialidad médica del profesional, entre las áreas disponibles en el sistema. También se tiene un ListView donde se muestra los médicos que se agregan.

En la segunda pestaña al igual que la anterior se agregaron los elementos necesarios para registrar los pacientes, se utilizó Label para identificar los campos y cuadros de texto TextBox para ingresar los datos solicitados. Los datos que se registran son: DNI, Nombre, Apellido paterno, Apellido materno. En esta ventana también se incluye un ComboBox para seleccionar el área médica en la cual el

paciente será atendido y además cuenta con un CheckBox que permite indicar si el paciente tiene prioridad de atención.

En la tercera pestaña se gestiona la atención de los pacientes registrados. Hay un ComboBox en el que se puede seleccionar la especialidad de la cual se va a atender al paciente, y al presionar el botón “Atender”, el sistema desencola al siguiente paciente de dicha especialidad, a su vez el sistema muestra un MessageBox en el cual se puede ver los datos del paciente y del médico que lo atenderá. El mensaje incluye el nombre completo y el DNI del paciente, el nombre completo del médico y el área médica correspondiente donde se realizará la atención. Además, la pestaña incluye un DataGridView que muestra visualmente el estado de las colas de pacientes para cada área médica, permitiendo observar en tiempo real qué pacientes están esperando en cada especialidad.

En la cuarta pestaña se muestra el historial de todos los pacientes que han sido atendidos en la clínica y se muestra a través de un ListView.

En cuanto a la parte del código se creó una biblioteca de clases y se agregaron las clases necesarias para la correcta ejecución del GUI. El sistema cuenta con las siguientes clases:

Medico. En el cual se encuentran los datos requeridos para el registro de un nuevo médico.

```
public class Medico
{
    public int Id;
    public string DNI;
    public string Nombre;
    public string ApellidoPaterno;
    public string ApellidoMaterno;
    public string Area;
    public Cola_Pacientes Pacientes;
}
```

Nodo_Medico. Esta clase nos permite la creación del Árbol en el que se guardarán los médicos registrados.

```
public class Nodo_Medico
{
    public Medico dato;
    public Nodo_Medico Izquierdo;
    public Nodo_Medico Derecho;
}
```

Arbol. En esta clase se tiene dos métodos:

- **insertar:** este método sirve para ir agregando los médicos al árbol

```

if (raiz == null)
{
    Nodo_Medico nuevo = new Nodo_Medico();
    nuevo.dato = d;

    raiz = nuevo;
}
  
```

Esta parte sirve para la creación de la raíz en caso de que no haya ningún médico registrado aún.

```

else
{
    if (d.Id < raiz.dato.Id)
    {
        insertar(ref raiz.Izquierdo, d);
    }
    else if (d.Id > raiz.dato.Id)
    {
        insertar(ref raiz.Derecho, d);
    }
    else
    {
        Console.WriteLine("DATO DUPLICADO");
    }
}
  
```

Esta parte para agregar las ramas cuando ya existe una raíz.

- **Insertar:** este método sirve para poder hacer el llamado en GUI de manera más sencilla.

```

public void Insertar(Medico d)
{
    insertar(ref raiz_principal, d);
}
  
```

- **buscar_Area:** Recorre todo el árbol hasta encontrar y devolver al médico según la especialidad que tiene.

```

private Medico buscar_Area(Nodo_Medico raiz, string area)
{
    if (raiz == null) return null;

    if (raiz.dato.Area == area)
        return raiz.dato;

    Medico encontrado = buscar_Area(raiz.Izquierdo, area);
    if (encontrado != null) return encontrado;

    return buscar_Area(raiz.Derecho, area);
}

2 referencias
public Medico Buscar_Area(string area)
{
    return buscar_Area(raiz_principal, area);
}
  
```

- **Buscar_Area:** este método sirve para poder hacer el llamado en GUI de manera más sencilla.

```
public Medico Buscar_Area(string area)
{
    return buscar_Area(raiz_principal, area);
}
```

Paciente. En el cual se encuentran los datos requeridos para el registro de un nuevo paciente.

```
public class Paciente
{
    public string DNI;
    public string Nombre;
    public string ApellidoPaterno;
    public string ApellidoMaterno;
    public int Prioridad;
}
```

Nodo_Paciente. Esta clase nos permite la creación de la cola en el que se guardarán los pacientes registrados.

```
public class Nodo_Paciente
{
    public Paciente dato;
    public Nodo_Paciente Siguiente;
}
```

Cola_Paciente. Esta clase contiene dos métodos:

- **Encolar:** Este método inserta pacientes en una cola, asegurando que los de prioridad alta sean atendidos antes, sin perder el orden de llegada entre pacientes del mismo tipo.

```

public void Encolar(Paciente p)
{
    Nodo_Paciente nuevo = new Nodo_Paciente();
    nuevo.dato = p;

    if (frente == null)
    {
        frente = nuevo;
        final = nuevo;
    }
    else
    {
        if (p.Prioridad == 0)
        {
            final.Siguiente = nuevo;
            final = nuevo;
        }
        else
        {
            if (frente.dato.Prioridad == 0)
            {
                nuevo.Siguiente = frente;
                frente = nuevo;
            }
            else
            {
                Nodo_Paciente temp = frente;
                while (temp != null && temp.Siguiente.dato.Prioridad == 1)
                {
                    temp = temp.Siguiente;
                }
                nuevo.Siguiente = temp.Siguiente;
                temp.Siguiente = nuevo;
            }
        }
    }
}
  
```

- **Desencolar:** El método elimina y devuelve al primer paciente de la cola, avanzando el puntero frente al siguiente nodo. Si la cola está vacía, devuelve null.

```

public Paciente Desencolar()
{
    if (frente != null)
    {
        Paciente p = frente.dato;

        frente = frente.Siguiente;

        return p;
    }
    return null;
}
  
```

Nodo_Historial: Representa un nodo para una lista enlazada que guarda el historial de atenciones. Cada nodo contiene la información de una atención paciente y médico y una referencia al siguiente nodo

```

public class Nodo_Historial
{
    public string PacienteNombre;
    public string PacienteDNI;
    public string MedicoNombre;
    public string MedicoArea;
    public Nodo_Historial Siguiente;
}
  
```

Form1.

En esta clase primero se crea un objeto del tipo Arbol, que almacena todos los médicos registrados. Se declara una variable que apunta al inicio de una lista enlazada de áreas médicas. Se inicializa una variable que servirá como inicio del historial de atenciones que inicia con el valor null.

```

Arbol arbolMedicos = new Arbol();
Nodo_Area listaAreas;
Nodo_Historial historialAtenciones = null;
  
```

Contamos con un constructor Form1() que nos permite crear la interfaz gráfica, cargar las áreas médicas llamado al método InicializarAreas(), Registrar médicos por defecto llamando al método Médicos Defecto(), método que fue creado para agilizar la prueba al momento de la ejecución y poder agregar directamente los pacientes a cualquier área. y también contiene la configuración para el historial que se mostrará en el ListView.

```

public Form1()
{
    InitializeComponent();
    InicializarAreas();
    Medicos_Defecto();
    lv_Historial.View = View.Details;
    lv_Historial.FullRowSelect = true;
    lv_Historial.GridLines = true;
    lv_Historial.Columns.Add("Paciente", 180);
    lv_Historial.Columns.Add("DNI", 100);
    lv_Historial.Columns.Add("Médico", 180);
    lv_Historial.Columns.Add("Área", 150);
}
  
```

En este apartado también contamos con varias clases, las cuales son:

Nodo_Area. El cual tiene un nodo de una lista enlazada simple que guarda el nombre del área y un puntero al siguiente nodo.

```

public class Nodo_Area
{
    public string Nombre;
    public Nodo_Area Siguiente;

    8 referencias
    public Nodo_Area(string nombre)
    {
        Nombre = nombre;
        Siguiente = null;
    }
}
  
```

InicializarAreas. En este método se crea la lista enlazada con las áreas médicas y carga los nombres en los ComboBox del formulario.

```

private void InicializarAreas()
{
    cb_Especialidad.Items.Clear();
    cb_Area.Items.Clear();
    cb_Area_Desencolar.Items.Clear();

    listaAreas = new Nodo_Area("Medicina General");
    Nodo_Area actual = listaAreas;
    actual.Siguiente = new Nodo_Area("Ginecologia"); actual = actual.Siguiente;
    actual.Siguiente = new Nodo_Area("Traumatologia"); actual = actual.Siguiente;
    actual.Siguiente = new Nodo_Area("Neumologia"); actual = actual.Siguiente;
    actual.Siguiente = new Nodo_Area("Cardiologia"); actual = actual.Siguiente;
    actual.Siguiente = new Nodo_Area("Pediatría"); actual = actual.Siguiente;
    actual.Siguiente = new Nodo_Area("Dermatología"); actual = actual.Siguiente;
    actual.Siguiente = new Nodo_Area("Gastroenterología");
    Nodo_Area temp = listaAreas;
    while (temp != null)
    {
        cb_Especialidad.Items.Add(temp.Nombre);
        cb_Area.Items.Add(temp.Nombre);
        cb_Area_Desencolar.Items.Add(temp.Nombre);
        temp = temp.Siguiente;
    }
}
  
```

Médicos Defecto. Recorre la lista de áreas y crea un médico por cada una con datos predefinidos. Inserta cada médico en el árbol binario. Esta clase fue creada a fines prácticos, para agilizar la prueba al momento de probar los registros, es decir que como producto final es omisible.

```

private void Medicos_Defecto()
{
    Nodo_Area temp = ListaAreas;
    int id = 1;

    while (temp != null)
    {
        Medico nuevo = new Medico();
        nuevo.Id = id;

        switch (temp.Nombre)
        {
            case "Medicina General":
                nuevo.Nombre = "Luis";
                nuevo.ApellidoPaterno = "Herrera";
                nuevo.ApellidoMaterno = "Campos";
                nuevo.DNI = "12345678";
                break;
            case "Ginecología":
                nuevo.Nombre = "María";
                nuevo.ApellidoPaterno = "Rojas";
                nuevo.ApellidoMaterno = "Flores";
                nuevo.DNI = "23456789";
                break;
            case "Traumatología":
                nuevo.Nombre = "José";
                nuevo.ApellidoPaterno = "Ramírez";
                nuevo.ApellidoMaterno = "Torres";
                nuevo.DNI = "34567898";
                break;
            case "Neurología":
                nuevo.Nombre = "Ricardo";
                nuevo.ApellidoPaterno = "Fernández";
                nuevo.ApellidoMaterno = "Cruz";
                nuevo.DNI = "45678901";
                break;
            case "Cardiología":
                nuevo.Nombre = "Pedro";
                nuevo.ApellidoPaterno = "Salazar";
                nuevo.ApellidoMaterno = "Mendoza";
                nuevo.DNI = "56789812";
                break;
            case "Pediatría":
                nuevo.Nombre = "Ana";
                nuevo.ApellidoPaterno = "Morales";
                nuevo.ApellidoMaterno = "Gómez";
                nuevo.DNI = "67898123";
                break;
            case "Dermatología":
                nuevo.Nombre = "Javier";
                nuevo.ApellidoPaterno = "Quiroz";
                nuevo.ApellidoMaterno = "Soto";
                nuevo.DNI = "789901234";
                break;
            case "Gastroenterología":
                nuevo.Nombre = "Carla";
                nuevo.ApellidoPaterno = "Sánchez";
                nuevo.ApellidoMaterno = "Vargas";
                nuevo.DNI = "89012345";
                break;
        }

        nuevo.Area = temp.Nombre;
        nuevo.Pacientes = new Cola_Pacientes();
        arbolMedicos.Insertar(nuevo);
        id++;
        temp = temp.Siguiente;
    }

    MessageBox.Show("Se registraron médicos por defecto para todas las áreas.", "Inicio del sistema");
}
  
```

btn_Registrar_Medico_Click. Es el Método por defecto al ejecutar la acción click en el botón nombrado Registrar de la pestaña Registro Médicos. En este método se crea un nuevo médico con los datos de los TextBox y se agrega al árbol, finalmente se limpia los campos de texto.

```

private void btn_Registrar_Medico_Click(object sender, EventArgs e)
{
    Medico nuevo = new Medico();
    nuevo.Id = int.Parse(txt_ID.Text);
    nuevo.DNI = txt_DNI.Text;
    nuevo.Nombre = txt_Nombre.Text;
    nuevo.ApellidoPaterno = txt_Paterno.Text;
    nuevo.ApellidoMaterno = txt_Materno.Text;
    nuevo.Area = cb_Especialidad.Text;
    nuevo.Pacientes = new Cola_Pacientes();

    arbolMedicos.Insertar(nuevo);

    MessageBox.Show("Médico registrado correctamente.", "Registro Médico");

    txt_ID.Text = "";
    txt_DNI.Text = "";
    txt_Nombre.Text = "";
    txt_Paterno.Text = "";
    txt_Materno.Text = "";
    cb_Especialidad.SelectedIndex = -1;
}
  
```

btn_Registrar_Paciente_Click. Es el Método por defecto al ejecutar la acción click en el botón nombrado Registrar de la pestaña Registro Pacientes. En este método a través del área seleccionada en el ComboBox se busca al médico de esa especialidad y luego lo agrega a una cola para ese médico, finalmente se limpia los campos.

```

private void btn_Registrar_Paciente_Click(object sender, EventArgs e)
{
    string area = cb_Area.Text;
    Medico medico = arbolMedicos.Buscar_Area(area);

    if (medico != null)
    {
        Paciente p = new Paciente();
        p.DNI = txt_DNIPaciente.Text;
        p.Nombre = txt_NPaciente.Text;
        p.ApellidoPaterno = txt_PaternoPaciente.Text;
        p.ApellidoMaterno = txt_MaternoPaciente.Text;
        p.Prioridad = checkB_Prioridad.Checked ? 1 : 0;

        medico.Pacientes.Encolar(p);

        MessageBox.Show("Paciente registrado correctamente.", "Registro Paciente");

        txt_DNIPaciente.Text = "";
        txt_NPaciente.Text = "";
        txt_PaternoPaciente.Text = "";
        txt_MaternoPaciente.Text = "";
        cb_Area.SelectedIndex = -1;
        checkB_Prioridad.Checked = false;

        ActualizarTabla();
    }
    else
    {
        MessageBox.Show("No existe un médico registrado en el área seleccionada.", "Aviso");
    }
}
  
```

btn_Atender_Click. En este método a través de la especialidad seleccionada en el ComboBox de la pestaña se desencola al primero de dicha especialidad. Se muestra un mensaje con los datos del paciente que debe pasar para atención, los datos de quien lo

va a atender y a que área de la clínica debe dirigirse. Además agrega el paciente desencolado al historial.

```
private void btn_Atender_Click(object sender, EventArgs e)
{
    string area = cb_Area_Desencolar.Text;
    Medico medico = arbolMedicos.Buscar_Area(area);

    if (medico != null)
    {
        Paciente atendido = medico.Pacientes.Desencolar();
        if (atendido != null)
        {
            MessageBox.Show("Paciente:\n" +
                atendido.Nombre + " " + atendido.ApellidoPaterno + " " + atendido.ApellidoMaterno +
                "\nDNI: " + atendido.DNI +
                "\n\nPasar para atención con el doctor:\n" +
                "Dr. " + medico.Nombre + " " + medico.ApellidoPaterno + " " + medico.ApellidoMaterno +
                "\n\nEn el área de:\n" +
                medico.Area,
                "Atención");

            AgregarHistorial(atendido, medico);
            ActualizarHistorial();
            ActualizarTablaO();
        }
        else
        {
            MessageBox.Show("No hay pacientes en la cola para esta área.", "Cola vacía");
        }
    }
    else
    {
        MessageBox.Show("No existe un médico en el área seleccionada.", "Aviso");
    }
}
```

ActualizarTabla. Crea columnas por área y muestra los pacientes en cola por cada área a través del DataGridView.

```
private void ActualizarTabla()
{
    dgv_Vista.Rows.Clear();
    dgv_Vista.Columns.Clear();
    Nodo_Area temp = listaAreas;
    while (temp != null)
    {
        dgv_Vista.Columns.Add(temp.Nombre, temp.Nombre);
        temp = temp.Siguiente;
    }
    int maxPacientes = ObtenerMaxPacientes(arbolMedicos.raiz_principal);
    for (int i = 0; i < maxPacientes; i++)
    {
        int index = dgv_Vista.Rows.Add();
        DataGridViewRow fila = dgv_Vista.Rows[index];
        LlenarFilaPorArea(arbolMedicos.raiz_principal, fila, i);
    }
}
```

ObtenerMaxPacientes. Recorre todo el árbol de médicos y devuelve la fila con mayor longitud de cola entre todos los médicos del árbol. Se usa para saber cuántas filas debe tener el DataGridView cuando muestras las colas por área.

```
private int ObtenerMaxPacientes(Nodo_Medico raiz)
{
    if (raiz == null) return 0;

    int maxIzq = ObtenerMaxPacientes(raiz.Izquierdo);
    int maxDer = ObtenerMaxPacientes(raiz.Derecho);
    int cantidad = ContarPacientes(raiz.dato.Pacientes.frente);

    int mayor = cantidad;
    if (maxIzq > mayor) mayor = maxIzq;
    if (maxDer > mayor) mayor = maxDer;

    return mayor;
}
```

ContarPacientes. Cuenta los pacientes que hay en la cola de pacientes.

```
private int ContarPacientes(Nodo_Paciente nodo)
{
    int count = 0;
    while (nodo != null)
    {
        count++;
        nodo = nodo.Siguiente;
    }
    return count;
}
```

LlenarFilaPorArea. Este método se encarga de escribir el nombre del paciente en la casilla del DataGridView según la especialidad a la que se va a derivar.

```
private void LlenarFilaPorArea(Nodo_Medico raiz, DataGridViewRow fila, int posicion)
{
    if (raiz == null) return;

    foreach (DataGridViewColumn col in dgv_Vista.Columns)
    {
        if (col.HeaderText == raiz.dato.Area)
        {
            Nodo_Paciente paciente = ObtenerNodoEnPosicion(raiz.dato.Pacientes.frente, posicion);
            if (paciente != null)
            {
                fila.Cells[col.Index].Value = paciente.dato.Nombre;
            }
            break;
        }
    }

    LlenarFilaPorArea(raiz.Izquierdo, fila, posicion);
    LlenarFilaPorArea(raiz.Derecho, fila, posicion);
}
```

ObtenerNodoEnPosicion. Este método devuelve el nodo paciente en una posición específica de la cola.

```
private Nodo_Paciente ObtenerNodoEnPosicion(Nodo_Paciente frente, int posicion)
{
    int indice = 0;
    Nodo_Paciente temp = frente;
    while (temp != null)
    {
        if (indice == posicion)
            return temp;
        temp = temp.Siguiente;
        indice++;
    }
    return null;
}
```

AgregarHistorial. Este método crea un nodo con los datos de la atención (paciente + médico) y lo añade al final de la lista enlazada HistorialAtenciones.

```
private void AgregarHistorial(Paciente p, Medico m)
{
    Nodo_Historial nuevo = new Nodo_Historial();
    nuevo.PacienteNombre = p.Nombre + " " + p.ApellidoPaterno + " " + p.ApellidoMaterno;
    nuevo.PacienteDNI = p.DNI;
    nuevo.MedicoNombre = m.Nombre + " " + m.ApellidoPaterno + " " + m.ApellidoMaterno;
    nuevo.MedicoArea = m.Area;

    if (historialAtenciones == null)
    {
        historialAtenciones = nuevo;
    }
    else
    {
        Nodo_Historial temp = historialAtenciones;
        while (temp.Siguiente != null)
        {
            temp = temp.Siguiente;
        }
        temp.Siguiente = nuevo;
    }
}
```

ActualizarHistorial. Este método se encarga de actualizar la vista de los pacientes atendidos en el ListView.

```
private void ActualizarHistorial()
{
    lv_Historial.Items.Clear();

    Nodo_Historial temp = historialAtenciones;
    while (temp != null)
    {
        ListViewItem item = new ListViewItem(temp.PacienteNombre);
        item.SubItems.Add(temp.PacienteDNI);
        item.SubItems.Add(temp.MedicoNombre);
        item.SubItems.Add(temp.MedicoArea);
        lv_Historial.Items.Add(item);

        temp = temp.Siguiente;
    }
}
```

VIII. RECOMENDACIONES FINALES

VIII.1 Conclusiones

El desarrollo del Sistema de Gestión de Citas Médicas permitió aplicar de forma práctica los conocimientos adquiridos en el curso de Estructura de Datos, demostrando la utilidad de las colas, árboles y listas enlazadas en la resolución de problemas reales.

La implementación del sistema logró optimizar los procesos administrativos de la clínica, reduciendo los errores asociados al registro manual de información y mejorando la eficiencia en la asignación de pacientes a médicos según su especialidad.

El uso de un entorno de desarrollo moderno como Visual Studio 2022 facilitó la creación de una interfaz gráfica intuitiva, contribuyendo a que el sistema sea accesible y fácil de manejar para el personal no especializado en informática.

El proyecto evidenció la importancia de la digitalización en la gestión de servicios de salud, ya que permite centralizar la información, mejorar la comunicación interna y brindar una atención más rápida y ordenada a los pacientes.

VIII.2 Recomendaciones

- Se recomienda continuar con la ampliación del sistema, incorporando nuevas funcionalidades como la generación automática de reportes estadísticos, control de horarios médicos y gestión de citas futuras.
- Se sugiere capacitar al personal administrativo y médico en el uso del sistema, con el fin de aprovechar al máximo sus beneficios y asegurar una correcta adopción tecnológica.
- Finalmente, se recomienda mantener un mantenimiento periódico del software para corregir posibles errores, optimizar el rendimiento y adaptar la aplicación a las necesidades cambiantes de la clínica.

IX. COMPROMISO ÉTICO EN EL EJERCICIO PROFESIONAL

Yo Josue Willy Bustamante Colunche, como estudiante de la carrera de Ingeniería de sistemas Computacionales de la Universidad Privada del Norte, me comprometo a respetar los principios, valores y normativas que rigen a esta institución para el desarrollo de trabajos académicos, cumpliendo con las indicaciones para el desarrollo de estos trabajos. De tal manera respetar la información de otros autores citando correctamente, tener la autorización de uso de información de la empresa

en la cual se está trabajando el proyecto. Asimismo, me comprometo en cumplir con las normativas éticas del ejercicio profesional de un ingeniero y aportar de manera positiva tanto en el campo profesional, social y académico.

Yo Jhonny Rafael Alonzo Rabanal, como estudiante de la carrera de Ingeniería de sistemas Computacionales de la Universidad Privada del Norte, me comprometo a respetar los principios, valores y normativas que rigen a esta institución para el desarrollo de trabajos académicos, cumpliendo con las indicaciones para el desarrollo de estos trabajos. De tal manera respetar la información de otros autores citando correctamente, tener la autorización de uso de información de la empresa en la cual se está trabajando el proyecto. Asimismo, me comprometo en cumplir con las normativas éticas del ejercicio profesional de un ingeniero y aportar de manera positiva tanto en el campo profesional, social y académico.

X. REFERENCIAS O BIBLIOGRAFÍA

Pérez Acharte, R., & Ponce de la Cruz, D. (2022). *Desarrollo e implementación de un sistema web para la gestión de citas médicas y hospitalización de la clínica MEDISALUD S.A.C.* [Tesis de licenciatura, Instituto IDAT]. Repositorio Institucional IDAT. <https://hdl.handle.net/20.500.14364/78>

Benites Ortiz, A. (2022). *Propuesta de implementación de un sistema de gestión web de historias clínicas en el Centro Médico Ocupacional Sanna – Talara* [Tesis de licenciatura, Universidad Católica Los Ángeles de Chimbote]. Repositorio Institucional ULADECH Católica. <https://hdl.handle.net/20.500.13032/31111>

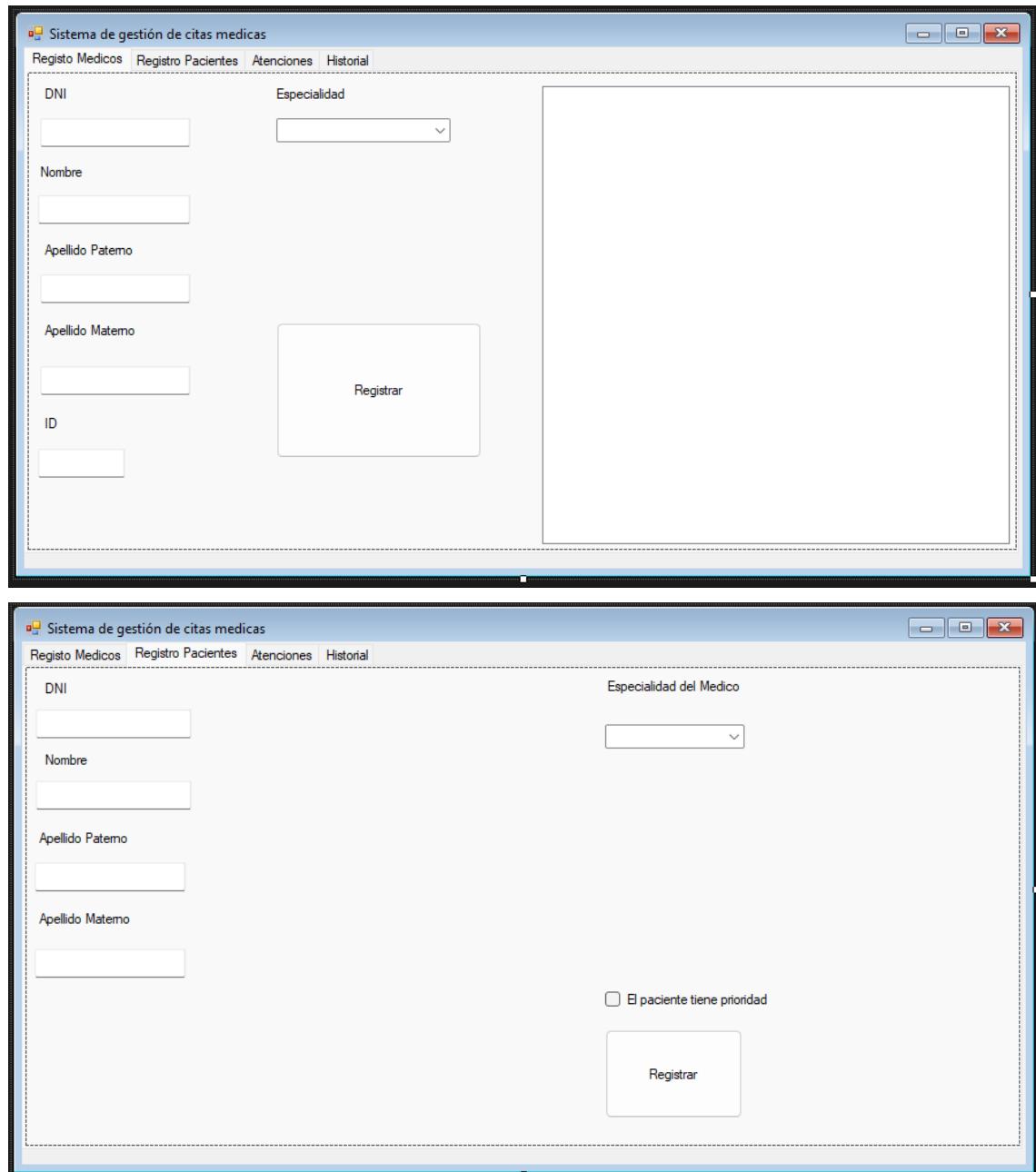
Meléndez Gárate, R. (2023). *Sistema informático hospitalario y la gestión en la historia clínica electrónica ambulatoria en una clínica privada, Lima – 2023* [Tesis de maestría, Universidad César Vallejo]. Repositorio Institucional UCV. <https://hdl.handle.net/20.500.12692/125047>

Ruiz Ricra, F. M. (2024). *Sistema de gestión de clínica dental para odontólogos independientes* [Tesis de licenciatura, Pontificia Universidad Católica del Perú]. Repositorio Institucional PUCP. <http://hdl.handle.net/20.500.12404/27078>

Caicedo Saavedra, R. (2024). *Sistema web de historias clínicas electrónicas para la gestión de pacientes ambulatorios en la Clínica San Felipe, Lima – 2023* [Tesis de

licenciatura, Universidad Privada de Iquitos]. Repositorio Institucional UNAPI.
<https://hdl.handle.net/20.500.12737/10292>

XI. ANEXOS



The image displays two windows of a medical appointment management system. Both windows have a title bar 'Sistema de gestión de citas médicas' and a menu bar with tabs: 'Registro Medicos', 'Registro Pacientes', 'Atenciones', and 'Historial'.

The top window shows a registration form with the following fields:

- DNI (input field)
- Especialidad (dropdown menu)
- Nombre (input field)
- Apellido Paterno (input field)
- Apellido Materno (input field)
- ID (input field)
- A large empty text area on the right.
- A 'Registrar' button.

The bottom window shows a similar registration form but includes an additional checkbox and a larger text area:

- DNI (input field)
- Especialidad del Medico (dropdown menu)
- Nombre (input field)
- Apellido Paterno (input field)
- Apellido Materno (input field)
- A checkbox labeled 'El paciente tiene prioridad' (The patient has priority).
- A large empty text area on the right.
- A 'Registrar' button.

