

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESTRUCTURA DE DATOS

CATEDRÁTICO: ING. WILLIAM ESTUARDO ESCOBAR ARGUETA

TUTOR ACADÉMICO: AYESER CRISTIÁN OXLAJ JUÁREZ



Brandon Antonio Marroquin Pérez

CARNÉ: 202300813

SECCIÓN: F

GUATEMALA, 09 DE MARZO DEL 2,024

## ÍNDICE

INTRODUCCIÓN .....	2
OBJETIVOS .....	2
1.    GENERAL .....	2
2.    ESPECÍFICOS .....	2
ALCANCES DEL SISTEMA.....	3
ESPECIFICACIÓN TÉCNICA .....	4
•    REQUISITOS DE HARDWARE .....	4
•    REQUISITOS DE SOFTWARE.....	4
DESCRIPCIÓN DE LA SOLUCIÓN .....	5
LÓGICA DEL PROGRAMA .....	6
➤    Librerías.....	6
➤    Variables Globales y privadas de la .....	7
➤    Función Main .....	10
➤    Métodos y Funciones utilizadas .....	10

# **INTRODUCCIÓN**

Este Manual Técnico describe la creación de un SISTEMA DE CONTROL DE CITAS DE HOSPITAL en el lenguaje Java. El propósito primordial de este sistema es proporcionar a los profesionales de la salud una herramienta integral que permita la gestión centralizada y efectiva de las citas médicas, optimizando los recursos disponibles, garantizando la puntualidad en la atención y mejorando la experiencia tanto para los pacientes como para el personal médico.

## **OBJETIVOS**

### **1. GENERAL**

- 1.1. Desarrollar un Sistema de Control de Citas de Hospital en Java, utilizando programación orientada a objetos, que optimice la gestión de citas médicas, mejore la eficiencia operativa del personal de salud y brinde una experiencia más eficaz a pacientes y profesionales involucrados.

### **2. ESPECÍFICOS**

- 2.1. Crear una interfaz de usuario amigable que permita tanto al personal médico como a los pacientes acceder de manera intuitiva y eficiente a las funciones del sistema.
- 2.2. Establecer un sistema de gestión de permisos y autenticación para controlar el acceso a funciones específicas, asegurando que solo el personal autorizado pueda realizar ciertas acciones.

## **ALCANCES DEL SISTEMA**

El Sistema de Control de Citas de Hospital en Java se concentra en la gestión ágil de citas médicas, abarcando la programación, modificación y cancelación para optimizar los horarios del personal médico y mejorar la experiencia del paciente. Incluye funciones de registro de información del paciente, interfaz de usuario adaptada, notificaciones automatizadas, medidas de seguridad, gestión de recursos médicos y reportes básicos. El manual proporciona una guía concisa para la instalación, configuración y uso del sistema, orientada a personal administrativo y usuarios finales, con el objetivo de lograr una implementación efectiva y un manejo eficiente en el entorno hospitalario.

# ESPECIFICACIÓN TÉCNICA

## ● REQUISITOS DE HARDWARE

- **Sistema operativo:**
- Windows 10 o superior
- macOS 10.14 o superior
- Ubuntu 18.04 o superior
- **Procesador:**
- Intel Core i5 o superior
- AMD Ryzen 5 o superior
- **Memoria RAM:**
- 12 GB o superior
- **Almacenamiento:**
- 5 GB de espacio libre en disco
- **Tarjeta gráfica:**
- Intel HD Graphics 620 o superior
- AMD Radeon R5 o superior
- **Pantalla:**
- Resolución de 1280 x 720 o superior
- **Software:**
- Java Development Kit (JDK) 17 o superior
- IntelliJ IDEA o similar IDE para Java
- Git (opcional para control de versiones)

## ● REQUISITOS DE SOFTWARE

- **Software esencial:**
- Java Development Kit (JDK) 17 o superior: Es el entorno de desarrollo necesario para compilar y ejecutar código Java.

- IntelliJ IDEA o similar IDE para Java: Es un entorno de desarrollo integrado (IDE) que proporciona herramientas para escribir, compilar, ejecutar y depurar código Java.
- Git (opcional para control de versiones): Es un sistema de control de versiones que permite realizar un seguimiento de los cambios en el código fuente y colaborar con otros desarrolladores.

## **DESCRIPCIÓN DE LA SOLUCIÓN**

Primeramente se creo el Login (apartado para iniciar sesión o registrarse), debido a que es la primera ventana y es la que administra cada módulo, luego se creo la ventana registro y sus funciones debido a que si uno no tuviera aun cuenta en la misma pues ahí lo podría registrar, después se creo el módulo 1, donde van los registros de los doctores, pacientes y productos, para su iniciación primero se crearon los botones, pestañas y tablas, ya después la parte difícil la cual es la agregación de cada uno de los doctores, pacientes y productos, y para finalizar la gráfica, luego se creo la ventana de pacientes la cual se entra a base del código y contraseña previamente creada en el modulo 1. Para la ventana Pacientes se creo primeramente la interfaz con sus colores, pestañas y botones, luego las fechas, horarios, lastimosamente no pude terminar el segundo modulo así que hasta aquí podría describirlo.

# LÓGICA DEL PROGRAMA

## IPC1\_Proyecto1\_202300813

```
import java.awt.*;
import java.awt.event.ActionEvent;
import javax.swing.*;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.Map;
import java.util.List;

//libreria de las graficas
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.CategoryAxis;
import org.jfree.chart.axis.CategoryLabelPositions;
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.data.category.DefaultCategoryDataset;

import java.util.ArrayList;
```

### ➤ Librerías

**java.awt:** Esta librería proporciona clases y métodos para la creación de interfaces gráficas de usuario (GUI) en Java. Incluye componentes como ventanas, botones, campos de texto y otros elementos que permiten la construcción de aplicaciones con una interfaz visual.

**javax.swing:** Extendiendo la funcionalidad de java.awt, javax.swing ofrece componentes de interfaz gráfica más avanzados y flexibles. Incluye elementos como botones, cuadros de diálogo, paneles y marcos, que permiten la creación de interfaces de usuario más modernas y dinámicas.

**java.awt.event.ActionEvent y java.awt.event.ActionListener:** Estas clases forman parte del paquete java.awt.event y se utilizan para manejar eventos de acción, como clics de botones. ActionEvent representa un evento de acción, mientras que ActionListener es una interfaz que escucha y responde a dichos eventos.

**java.util.ArrayList, java.util.Collections y java.util.Comparator:** Estas clases pertenecen al paquete java.util y se utilizan para trabajar con colecciones de objetos en Java. ArrayList proporciona una implementación dinámica de listas, Collections ofrece métodos útiles para manipular colecciones, y Comparator permite la personalización del ordenamiento de elementos en una colección.

**java.util.HashMap:** Esta clase pertenece al paquete java.util y representa una estructura de datos de tipo mapa, que almacena pares clave-valor. Es eficiente para la búsqueda y recuperación de datos mediante una clave.

**java.util.List:** La interfaz List del paquete java.util define una colección ordenada de elementos que permite el acceso y manipulación de datos de manera secuencial. Implementaciones comunes son ArrayList y LinkedList.

**org.jfree.chart:** Esta librería, representada por las clases que comienzan con org.jfree.chart, se utiliza para la creación de gráficos en Java. Incluye la creación de diversos tipos de gráficos, como barras, líneas y pasteles, facilitando la visualización de datos. La clase JFreeChart es central en la creación de los gráficos, y ChartPanel proporciona una interfaz gráfica para mostrarlos.

#### ➤ Variables Globales y privadas de la

IPC1\_Proyecto1\_202300813

```
public static ArrayList <Doc> listadoDoctores = new ArrayList<>();  
public static ArrayList<CreadorPaciente> listadoPaciente = new ArrayList<>();  
public static ArrayList<CreadorProducto> listadoProducto = new ArrayList<>();
```

```
        JTextArea motiCita;  
private JComboBox<String> Fecha;  
private JComboBox<String> Hora;  
private JComboBox<String> Especialidad;  
private JComboBox<String> cboDoctor;  
JButton exitButton;
```



```

private JTextField firstNameField;
private JTextField lastNameField;
private JPasswordField passwordField;
private JTextField especialidadField;
private JTextField telefonoField;
private JTextField edadField;

private JComboBox<String> genderComboBox;

private JButton updateButton;

private Doc doctorActual;

```

```

private JTextField nombrePaciente;
private JTextField apellidoPaciente;
private JPasswordField contraseñaPaciente;
private JTextField edadPaciente;
private JButton updateButton;
private CreadorPaciente pacienteActual;

```

```

private JTextField NombreProducto;
private JTextField PrecioProducto;
private JTextField DescripcionProducto;
private JTextField CantidadProducto;
private JButton updateButton4;
private CreadorProducto productoActual;

```

```

private int codigoPaciente;
private String nombrePaciente;
private String generoPaciente;
private int edadPaciente;
private String apellidoPaciente; //
private String contraseñaPaciente;

```

```

private int codigoProducto;
private String nombreProducto;
private double PrecioProducto;
private String DescripciónProdcuto;
private int cantidadProducto;

```

```

private int codigo;
private String nombre;
private String genero;
private int edad;
private String especialidad;
private String telefono;
private String apellido; //
private String contraseña;

```

listadoDoctores: Un ArrayList que almacena objetos de la clase Doc, presumiblemente representando a los doctores en el sistema.

listadoPaciente: Un ArrayList que almacena objetos de la clase CreadorPaciente, posiblemente para mantener un registro de pacientes.

listadoProducto: Un ArrayList que almacena objetos de la clase CreadorProducto, probablemente utilizado para gestionar productos o servicios relacionados con la aplicación.

motiCita: Un JTextArea, que probablemente se utiliza para mostrar o ingresar motivos de citas.

Fecha, Hora, Especialidad, cboDoctor: JComboBox, componentes de selección que podrían estar relacionados con la programación de citas y la elección de especialidades y médicos.

exitButton: JButton, un botón que podría utilizarse para cerrar la aplicación.

firstNameField, lastNameField, passwordField, etc.: JTextField y JPasswordField, campos de texto utilizados para ingresar información como nombre, apellido, contraseña, etc.

genderComboBox: JComboBox, posiblemente utilizado para seleccionar el género.

updateButton: JButton, botón de actualización que probablemente se utilice para modificar información en la interfaz.

doctorActual, pacienteActual, productoActual: Variables que probablemente mantienen una referencia al doctor, paciente o producto actualmente seleccionado o en proceso de modificación en la interfaz.

codigoPaciente: Un entero que podría servir como identificador único para un paciente.

nombrePaciente, generoPaciente, apellidoPaciente, contraseñaPaciente: Cadenas de texto que representan el nombre, género, apellido y contraseña del paciente respectivamente.

edadPaciente: Un entero que indica la edad del paciente.

codigoProducto: Un entero que podría actuar como un identificador único para un producto.

nombreProducto, DescripciónProducto: Cadenas de texto que representan el nombre y la descripción del producto.

PrecioProducto: Un número de punto flotante que indica el precio del producto.

cantidadProducto: Un entero que representa la cantidad disponible del producto.

codigo: Un entero que podría ser un identificador único para un doctor.

nombre, genero, apellido, contraseña: Cadenas de texto que representan el nombre, género, apellido y contraseña del doctor respectivamente.

edad: Un entero que indica la edad del doctor.

especialidad: Una cadena de texto que describe la especialidad del doctor.

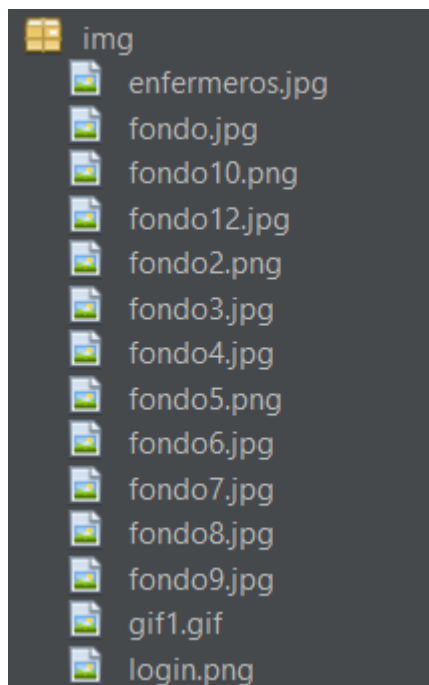
telefono: Una cadena de texto que contiene el número de teléfono del doctor.  
entre otros 85 más así similares...

### ➤ Función Main

```
public static void main(String[] args)
```

Se uso en si para dar inicio al programa ya que ahí es donde empieza toda la magia.

### ➤ Métodos y Funciones utilizadas



Imágenes utilizadas un 80% de todas las que están ahí.

```
public static Object[][] convertirDatosProductos tabla()
```

Se crea una matriz para guardar datos de los productos ingresados

```
public static Object[][] convertirDatosPacientes_tabla()
```

Se crea una matriz para guardar datos de los pacientes ingresados

```
public static Object[][] convertirDatosDoctores_tabla()
```

Se crea una matriz para guardar datos de los Doctores ingresados

```
public static void agregarproducto
```

El método agregarproducto tiene como objetivo agregar un nuevo producto a la lista listadoProducto


```
public static void agregarpaciente
```

El método agregarproducto tiene como objetivo agregar un nuevo producto a la lista listadoPaciente


```
public static void agregardoc
```

El método agregarproducto tiene como objetivo agregar un nuevo producto a la lista listadodoctores


Todas las clases utilizadas

 ActualizarDoc.java


Interfaz de la ventana Actualizar Doctor

 ActualizarPaciente.java


Interfaz de la ventana Actualizar Paciente

 ActualizarProducto.java


Interfaz de la ventana Actualizar producto

 Administrador.java

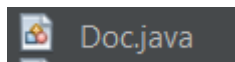
El modulo 1

 CreadorPaciente.java

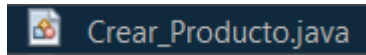
Esta clase permite la creación y manipulación de objetos que representan pacientes en el sistema, proporcionando métodos para obtener y establecer los valores de los atributos.

 CreadorProducto.java

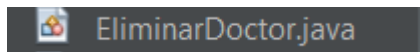
Esta clase permite la creación y manipulación de objetos que representan los productos en el sistema, proporcionando métodos para obtener y establecer los valores de los atributos



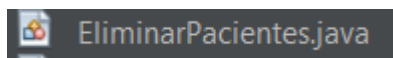
Esta clase permite la creación y manipulación de objetos que representan los Doctores en el sistema, proporcionando métodos para obtener y establecer los valores de los atributos



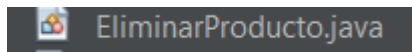
Para la creación de un producto



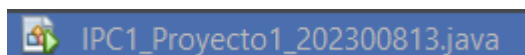
Para la eliminación del doctor



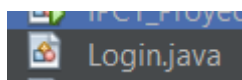
Para la eliminación de los pacientes



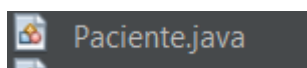
Para la eliminación de los Productos



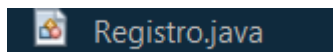
La clase donde se inicia todo y donde están los parámetros para dar inicio al programa



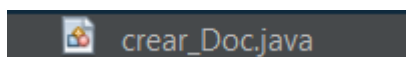
Ventana que uno ve de primero, donde se inicia sesión o se registra



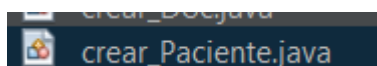
Ventana paciente (modulo 2)



Interfaz de la ventana donde uno se registra y con ello iniciar sesión



Interfaz y parámetros donde se sitúa para la creación de un doctor



Interfaz y parámetros donde se sitúa para la creación de un paciente



`guardardatosregistro.java`

Esta clase permite la creación y manipulación de objetos que representa en el registro en el sistema, proporcionando métodos para obtener y establecer los valores de los atributos.