

ALGORITMOS Y ESTRUCTURA DE DATOS**PROYECTO****Aplicativo para el Registro de Historias Clínicas de una Posta**

■ Ariza R., Sergio

Universidad Tecnológica del Perú – Sede: Lima Norte.

Lima, 17 de enero de 2025

Índice

1. Capítulo 1: ASPECTOS GENERALES	6
1.1. Presentación de la Empresa	6
1.2. Problemática	6
1.2.1. Problema General	6
1.2.2. Problema Específico	6
1.3. Objetivos	7
1.3.1. Objetivo General	7
1.3.2. Objetivo Específicos	7
1.4. Alcances	7
1.5. Justificaciones	8
1.5.1. Justificación Institucional	8
1.5.2. Justificación tecnológica	8
1.5.3. Justificación operacional	8
1.5.4. Justificación procedural	9
2. Capítulo 2: MARCO TEORICO	9
2.1. Antecedentes	9
2.2. Marco teorico	10
3. Capítulo 3: DESARROLLO DE LA SOLUCIÓN	11
3.1. Lista de Requerimientos funcionales	11
3.2. Lista de Requerimientos no funcionales	11
3.3. Prototipos del proyecto	12

3.4. Arreglo de Objetos	18
3.5. ArrayList	20
3.6. Listas Enlazadas Simples	22
3.7. Pilas	26
3.8. Listas Circulares	28
3.9. Persistencias de datos	31
3.10. Evidencia de imágenes de la funcionalidad de cada RF	36
4. Capítulo 4: Conclusiones y Recomendaciones	44
4.1. Conclusiones	44
4.2. Recomendaciones	45
5. Referencias Bibliográficas	45

Índice de cuadros

1. Requerimientos Funcionales	11
---	----

Índice de figuras

1. Diseño de login	12
2. Página de inicio	13
3. Página de registro de paciente	14
4. Página de registro de especialidad	14
5. Página de registro de médicos	15
6. Página de registro de historia clínica	15
7. Página de búsqueda de paciente	16

8. Página de búsqueda de especialidad	16
9. Página de búsqueda de médico	17
10. Página de búsqueda de historia clínica	17
11. Estructura para Especialidad	18
12. Estructura para Especialidad	19
13. Estructura para Especialidad	19
14. Estructura para Atencion	20
15. Estructura para Atencion	21
16. Estructura para Atencion	21
17. Estructura para Doctor	22
18. Estructura para Doctor	23
19. Estructura para Doctor	24
20. Estructura para Doctor	25
21. Estructura para Doctor	25
22. Estructura para Historia Clinica	26
23. Estructura para Historia Clinica	27
24. Estructura para Paciente	28
25. Estructura para Paciente	29
26. Estructura para Paciente	30
27. Estructura para Paciente	30
28. Estructura para Paciente	31
29. Persistencia de Atencion	31
30. Persistencia de Doctor	32

31. Persistencia de Especialidad	33
32. Persistencia de Historia Clinica	34
33. Persistencia de Paciente	35
34. Internal Frame de Pacientes	36
35. Internal Frame de Doctores	36
36. Internal Frame de Atenciones	37
37. Internal Frame de Especialidad	37
38. Internal Frame de Historia Clínica	38
39. Internal Frame de Historia Clínica	38
40. Internal Frame de Historia Clínica	39
41. Internal Frame de Historia Clínica	39
42. Internal Frame de Historia Clínica	40
43. Internal Frame de Historia Clínica	40
44. Generación de Pdf	41
45. Generación de Pdf	42
46. Búsqueda por Doctor para atención durante el día	42
47. Búsqueda por Doctor para atención durante el día	43
48. Búsqueda por Especialidad para atención por rango de fecha	43
49. Búsqueda por Doctor para atención durante el día	44

1. Capítulo 1: ASPECTOS GENERALES

1.1. Presentación de la Empresa

La Posta Médica "Salud para Todos" es una institución de atención primaria de salud que brinda servicios médicos a una población diversa en áreas rurales y urbanas. Está comprometida con el cuidado integral de los pacientes y busca mejorar la calidad del servicio mediante la implementación de tecnologías que permitan la gestión eficiente de los datos médicos.

1.2. Problemática

1.2.1. Problema General

Actualmente, la Posta Médica enfrenta dificultades en la gestión de las historias clínicas de los pacientes, las cuales se manejan de manera manual, lo que genera ineficiencias en el proceso, retrasos en la atención y riesgo de pérdida de información.

1.2.2. Problema Específico

- Retraso en la actualización de las historias clínicas debido al manejo manual.
- Falta de un acceso rápido y eficiente a los datos médicos en consultas.
- Riesgo de errores en el registro de datos médicos debido a la falta de automatización.
- Dificultad para realizar seguimientos y estadísticas de los pacientes a lo largo del tiempo.

1.3. Objetivos

1.3.1. Objetivo General

Desarrollar un Aplicativo para el Registro de Historias Clínicas que permita a la Posta Médica gestionar de manera digital la información de los pacientes, mejorando la eficiencia y reduciendo los errores en la atención médica.

1.3.2. Objetivo Específicos

- Implementar una plataforma digital para el registro y consulta de historias clínicas.
- Optimizar el tiempo de atención al reducir la búsqueda manual de datos.
- Minimizar los errores en el registro de la información médica de los pacientes.
- Facilitar la generación de informes y estadísticas médicas para el seguimiento de pacientes y la toma de decisiones clínicas.

1.4. Alcances

- **Implementación del Sistema Digital:** Desarrollar un aplicativo funcional que permita registrar, consultar y actualizar las historias clínicas de los pacientes de manera digital, eliminando el manejo manual de la información.
- **Manejo de Archivos Binarios:** Diseñar una solución que utilice archivos binarios para almacenar y gestionar los datos médicos, lo que simplificará el manejo de información sin necesidad de una base de datos.
- **Optimización de Procesos:** Agilizar los tiempos de atención y mejorar la eficiencia del personal médico mediante la automatización del registro y consulta de historias clínicas.

1.5. Justificaciones

1.5.1. Justificación Institucional

En la **Posta Médica "Salud para Todos"** están comprometidos a una mejora de calidad en sus servicios a través de la modernización de sus procesos. Por esto se implementará un sistema digital para gestionar las historias clínicas, se harán uso de los archivos binarios que ofrece una solución a la capacidad tecnológica de la institución, permitiendo gestionar información médica sin necesidad de una estructura de base de datos compleja.

1.5.2. Justificación tecnológica

El uso de archivos binarios se justifica por la simplicidad y eficiencia que ofrece en entornos donde una base de datos completa podría ser innecesaria y costosa. Los archivos binarios permitirán almacenar datos estructurados de manera directa en el sistema de archivos, lo que facilita el acceso, almacenamiento y recuperación de la información sin necesidad de depender de servidores. Además, la implementación en Java permite aprovechar sus capacidades para manejar archivos binarios de manera eficiente, garantizando seguridad y rapidez en la lectura y escritura de los datos.

1.5.3. Justificación operacional

Operacionalmente, el uso de archivos binarios agiliza el proceso de registros y consulta de las historias clínicas, eliminando la dependencia de bases de datos. Esto facilita el acceso a la información médica de los pacientes en cualquier momento y desde cualquier terminal dentro de la institución, reduciendo los tiempos de espera y mejorando la atención al paciente. Al no requerir una configuración compleja de la base de datos, se minimizan los costos operativos y los riesgos asociados a fallos en servidores externos, ofreciendo una solución simple pero efectiva.

1.5.4. Justificación procedural

El proyecto seguirá una estructura de desarrollo que permite la implementación del sistema, con un enfoque en el manejo seguro y eficiente de archivos binarios. Se utilizarán buenas prácticas de programación en java, como el uso de la serialización de objetos y validación de datos para asegurar que la información almacenada en los archivos binarios se integre y esté disponible de manera confiable.

2. Capítulo 2: MARCO TEORICO

2.1. Antecedentes

El manejo manual de historias clínicas es una problemática recurrente en centros de salud con infraestructura limitada. Esto genera retrasos, errores y dificultades en el acceso a la información. En diversos estudios, la automatización de estos procesos ha demostrado mejorar significativamente la eficiencia operativa, reducir tiempos de espera y minimizar errores humanos. Un sistema digital ofrece no solo mejoras en la gestión de información, sino que también facilita el seguimiento continuo de los pacientes a lo largo del tiempo.

En diversos países, se han implementado sistemas de historias clínicas electrónicas (HCE), lo que ha mejorado tanto la atención médica como la capacidad de análisis de datos a largo plazo. Estos sistemas permiten consultas rápidas de la información médica, generación de informes y estadísticas de manera automática, optimizando la toma de decisiones clínicas.

Mejia y Quiroga (2014) realizaron un estudio sobre la implementación de sistemas electrónicos de salud en centros de atención primaria, destacando que la digitalización de las historias clínicas mejora la eficiencia y reduce los errores humanos en un 25 % en comparación con sistemas manuales. Este estudio proporciona evidencia de cómo la tecnología aplicada al manejo de información

médica puede optimizar los procesos clínicos.

2.2. Marco teorico

- Automatización y Gestión de la Información Médica

La automatización en la gestión de información ha permitido a las instituciones de salud optimizar sus procesos, reduciendo errores humanos y facilitando el acceso a los datos. Esto se vincula con la teoría de los sistemas de información, donde la digitalización permite que los datos sean fácilmente accesibles, almacenados de forma segura y consultables en tiempo real.

- Almacenamiento de Datos con Archivos Binarios

En vez de utilizar bases de datos complejas, el uso de archivos binarios representa una solución simplificada en instituciones con recursos limitados. Estos archivos permiten el almacenamiento directo de información estructurada, sin requerir la implementación de servidores costosos o bases de datos relacionales. El lenguaje Java, que es ampliamente utilizado en sistemas de gestión de archivos binarios, ofrece una manera eficiente y segura de manejar estos datos.

- Principios de Ingeniería de Software

La implementación de un sistema digital para la gestión de historias clínicas sigue los principios de la ingeniería de software, que incluyen la modularidad, la escalabilidad y la seguridad. Estos sistemas deben estar diseñados para ofrecer acceso seguro y permitir un uso eficiente por parte del personal médico, además de permitir su futura actualización.

Cuadro 1: Requerimientos Funcionales

Manejo de Aplicativo Clínica	
Requerimiento	Descripción
Consultas/informes	
R1	El aplicativo permite búsqueda por DNI y por nombres de un paciente y realizar el mantenimiento correspondiente a una historia clínica.
R2	El aplicativo permite visualizar las atenciones que hizo un médico durante el día.
R3	El aplicativo permite visualizar por rango de fechas las atenciones realizadas por especialidad.
R4	El aplicativo permite visualizar en un PDF la historia clínica de un paciente.
Almacenamiento	
R5	El aplicativo permite registrar los datos de un paciente y realizar el mantenimiento correspondiente.
R6	El aplicativo permite registrar los datos de un doctor y realizar el mantenimiento correspondiente.
R7	El aplicativo permite registrar las especialidades de atención y realizar el mantenimiento correspondiente.
R8	El aplicativo permite registrar la historia clínica de un paciente y los detalles de una atención según la especialidad.
Procesamiento	
R9	El aplicativo permite el acceso a los recursos mediante un login.
R10	El aplicativo debe permitir la navegabilidad de los formularios mediante un menú.

3. Capítulo 3: DESARROLLO DE LA SOLUCIÓN

3.1. Lista de Requerimientos funcionales

3.2. Lista de Requerimientos no funcionales

- El sistema debe implementar un mecanismo de autenticación para controlar el acceso a los recursos, con diferentes niveles de permisos para usuarios (médicos, pacientes).
- Los datos médicos deben ser precisos y completos.

- El sistema debe estar documentado en su totalidad para permitir mejoras o cambios en el futuro.
- El lenguaje que se usará, deberá ser fuertemente tipado para evitar errores en los datos e información delicada como son los registros médicos, por consecuencia, se usará el lenguaje Java.

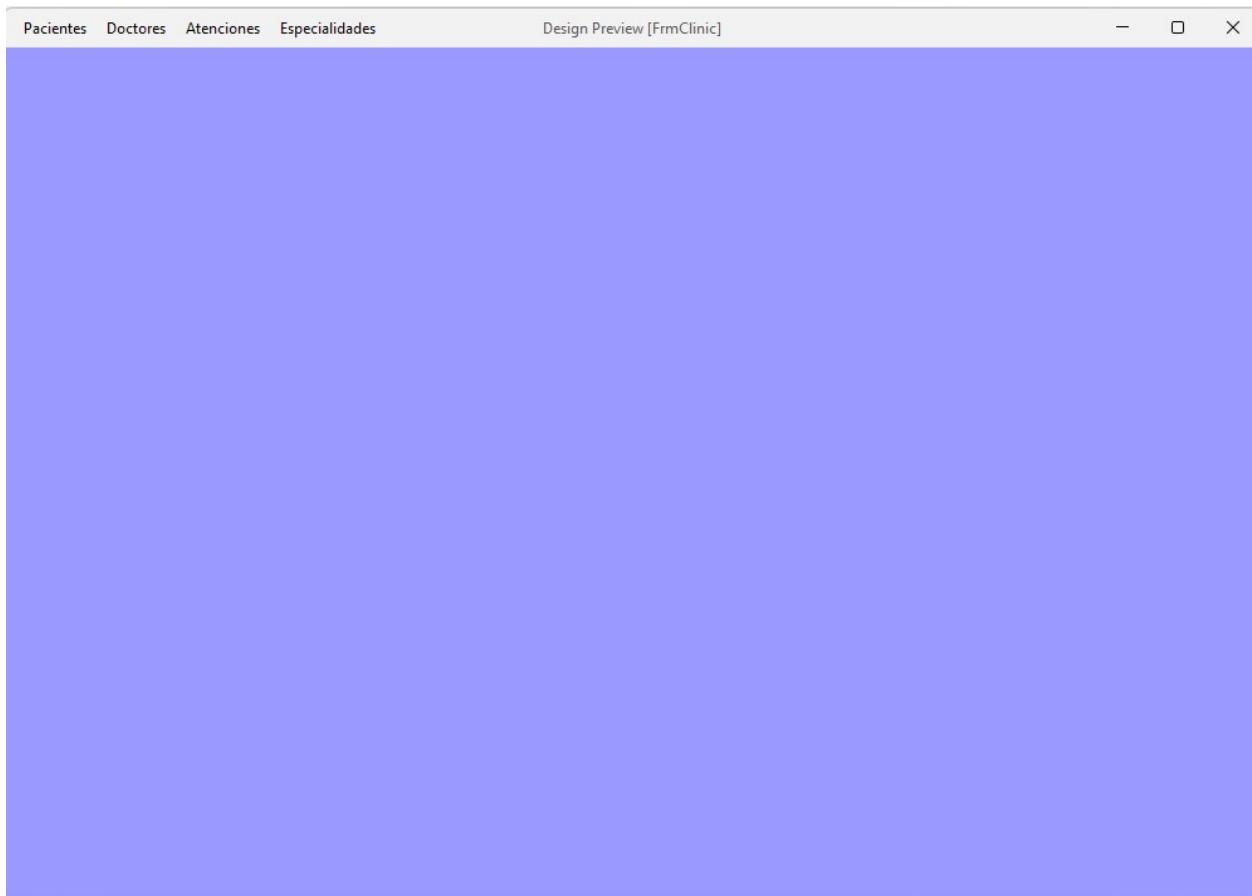
3.3. Prototipos del proyecto

Figura 1: Diseño de login



Nota: El usuario deberá ingresar su nombre de usuario y clave para ingresar al sistema de gestión clínico.

Figura 2: Página de inicio



Nota: Luego de que el usuario se autentifique podrá ingresar a la página de inicio.

Figura 3: Página de registro de paciente

Design Preview [FrmAddPatient]

DNI	Nombre	Género	Teléfono

Sección Pacientes

Dni

Nombre

Género

Masculino

Telefono

AGREGAR PACIENTE

Nota: En este formulario se podrá guardar la información del paciente.

Figura 4: Página de registro de especialidad

Design Preview [FrmAddEspec]

Código De Especialidad	Tipo De Especialidad	Descripción

Sección Especialidad

Código de Especialidad

Tipo de Especialidad

Descripción

AGREGAR ESPECIALIDAD

Nota: En este formulario se podrá guardar la información de la especialidad.

Figura 5: Página de registro de médicos

Design Preview [FrmAddDoctor]

Código	Nombre	DNI	Edad	Telefono	Especialidad
Médicos					
Código					
Nombres					
Dni					
Edad					
Telefono					
Especialidad					
Especialidad 1					
AGREGAR MÉDICO					

Nota: En este formulario se podrá guardar la información del médico.

Figura 6: Página de registro de historia clínica

Design Preview [FrmAddAttent]

Código De Médico	Código De Atención	DNI de paciente	Atención Brindada	Tipo de Especialidad
Atención				
Código de Médico				
Código de Atención				
Dni Paciente				
Atención Brindada				
Atención 1				
Especialidad				
Especialidad 1				
Detalles				
AGREGAR ATENCIÓN				

Nota: En este formulario se podrá guardar la información de historia clínica.

Figura 7: Página de búsqueda de paciente

The screenshot shows a Windows application window titled "Design Preview [FrmSearchPatient]". At the top left, there is a section labeled "Sección" with a button labeled "Pacientes". To the right are two search input fields: "Buscar por Nombre" and "Buscar por Dni", followed by a blue "BUSCAR" button. Below the search area is a table header row with columns labeled "DNI", "Nombre", "Género", and "Teléfono". A large, light blue rectangular area follows, likely representing a data grid or placeholder. At the bottom of the window is a teal-colored bar containing a single button labeled "IMPRIMIR HISTORIA CLÍNICA PDF".

Nota: En este formulario se podrá buscar la información del paciente.

Figura 8: Página de búsqueda de especialidad

The screenshot shows a Windows application window titled "Design Preview [FrmSearchEspec]". At the top left, there is a section labeled "Sección" with a button labeled "Especialidad". To the right are two search input fields: "Buscar por Tipo" and "Buscar por Código", followed by a blue "BUSCAR" button. Below the search area is a table header row with columns labeled "Código De Especialidad", "Tipo De Especialidad", and "Descripción". A large, light blue rectangular area follows, likely representing a data grid or placeholder.

Nota: En este formulario se podrá buscar la información de la especialidad.

Figura 9: Página de búsqueda de médico

The screenshot shows a Windows application window titled "Design Preview [FrmSearchDoctor]". At the top left is a section labeled "Sección Médicos". To its right are two search input fields: "Buscar por Nombre" and "Buscar por Código", both preceded by small labels. A large blue "BUSCAR" button is positioned to the right of these fields. Below this header is a horizontal row of six input fields with labels: "Código", "Nombre", "DNI", "Edad", "Teléfono", and "Especialidad". The main body of the form is a large, empty light-blue rectangular area.

Nota: En este formulario se podrá buscar la información del médico.

Figura 10: Página de búsqueda de historia clínica

The screenshot shows a Windows application window titled "Design Preview [FrmSearchAttent]". At the top left is a section labeled "Sección Atención". To its right are two search input fields: "Buscar por Médico" and "Buscar Tipo de Atención", both preceded by small labels. A large blue "BUSCAR" button is positioned to the right of these fields. Below this header is a horizontal row of five input fields with labels: "Código de Atención", "Médico a A cargo", "Paciente", "Tipo de Atención", and "Fecha". The main body of the form is a large, empty light-blue rectangular area.

Nota: En este formulario se podrá buscar la información de la historia clínica.

3.4. Arreglo de Objetos

Figura 11: Estructura para Especialidad

```
1  package Estructura.Atencion;
2  import Modelo.Atencion;
3  import Persistencia.Persistencia_Atencion;
4  import Procesos.Mensajes;
5  import java.io.Serializable;
6  import java.util.ArrayList;
7  import java.util.Calendar;
8  import java.util.Date;
9  import java.util.stream.Collectors;
10 public class ArrayListAtencion implements Serializable {
11     private ArrayList<Atencion> Lista;
12     private Persistencia_Atencion persistencia;
13     public ArrayListAtencion() {
14         persistencia = new Persistencia_Atencion();
15         Lista = persistencia.RecupereAtenciones();
16     }
17     private void GuardarCambios() {
18         persistencia.GuardarAtenciones(Lista);
19     }
20     public void AñadirAtencion(Atencion a) {
21         Lista.add(a);
22         GuardarCambios();
23     }
24     public void EliminarAtencion(Atencion a) {
25         Lista.remove(a);
26         GuardarCambios();
27     }
28     public Atencion ObtenerAtencion(int i) {
29         return Lista.get(i);
30     }
31     public void ActualizarAtencion(int i, Atencion atencion) {
32         Lista.set(i, atencion);
33         GuardarCambios();
34     }
35     public int BuscarIndexAtencion(String codigo) {
36         for (int i = 0; i < Lista.size(); i++) {
37             if (Lista.get(i).getCodigo().contains(codigo)) {
38                 return i;
39             }
40         }
41         Mensajes.LeerMensaje("No se encontro el index de esa atencion");
42     }
43 }
```

Figura 12: Estructura para Especialidad

```

44   public ArrayList<Atencion> Lista_Doctor(String codigo, String dni) {
45       ArrayList<Atencion> atencionesDoctor = new ArrayList<>();
46       for (Atencion atencion : Lista) {
47           if (atencion.getDoctor().equalsIgnoreCase(codigo) && atencion.getDoctor().getDni().equalsIgnoreCase(dni)) {
48               atencionesDoctor.add(atencion);
49           }
50       }
51       return atencionesDoctor;
52   }
53   public String CrearCodigoAtencion() {
54       int c1 = 5;
55       int c2 = Lista.size() + 1;
56
57       String codigo;
58       do {
59           codigo = "At" + c1 + "GT" + String.format("%03d", c2);
60           c2++;
61       } while (CodigoExiste(codigo));
62
63       return codigo;
64   }
65   private boolean CodigoExiste(String codigo) {
66       for (Atencion atencion : Lista) {
67           if (atencion.getCodigo().equals(codigo)) {
68               return true;
69           }
70       }
71       return false;
72   }
73   public ArrayList<Atencion> FiltrarPorEspecialidadYFechas(String especialidad, Date inicio, Date fin) {
74       return (ArrayList<Atencion>) Lista.stream()
75           .filter(a -> a.getEspecialidad_atencion().getNombre().equalsIgnoreCase(especialidad)
76               && (inicio == null || !a.getFecha().before(inicio))
77               && (fin == null || a.getFecha().after(fin)))
78           .collect(Collectors.toList());
79   }

```

Figura 13: Estructura para Especialidad

```

public ArrayList<Atencion> FiltrarPorDoctorYFecha(String codDoctor, Date fecha) {
    if (Lista == null) {
        return new ArrayList<>();
    }

    return new ArrayList<>(Lista.stream()
        .filter(a -> a.getDoctor().equalsIgnoreCase(codDoctor)
            && truncarFecha(a.getFecha()).equals(truncarFecha(fecha)))
        .collect(Collectors.toList()));
}

private Date truncarFecha(Date fecha) {
    Calendar cal = Calendar.getInstance();
    cal.setTime(fecha);
    cal.set(Calendar.HOUR_OF_DAY, 0);
    cal.set(Calendar.MINUTE, 0);
    cal.set(Calendar.SECOND, 0);
    cal.set(Calendar.MILLISECOND, 0);
    return cal.getTime();
}

public ArrayList<Atencion> getList() {
    return Lista;
}
}

```

3.5. ArrayList

Figura 14: Estructura para Atencion

```
1  package Estructura.Atencion;
2  import Modelo.Atencion;
3  import Persistencia.Persistencia_Atencion;
4  import Procesos.Mensajes;
5  import java.io.Serializable;
6  import java.util.ArrayList;
7  import java.util.Calendar;
8  import java.util.Date;
9  import java.util.stream.Collectors;
10 public class ArrayListAtencion implements Serializable {
11     private ArrayList<Atencion> Lista;
12     private Persistencia_Atencion persistencia;
13     public ArrayListAtencion() {
14         persistencia = new Persistencia_Atencion();
15         Lista = persistencia.RecupereAtenciones();
16     }
17     private void GuardarCambios() {
18         persistencia.GuardarAtenciones(Lista);
19     }
20     public void AñadirAtencion(Atencion a) {
21         Lista.add(a);
22         GuardarCambios();
23     }
24     public void EliminarAtencion(Atencion a) {
25         Lista.remove(a);
26         GuardarCambios();
27     }
28     public Atencion ObtenerAtencion(int i) {
29         return Lista.get(i);
30     }
31     public void ActualizarAtencion(int i, Atencion atencion) {
32         Lista.set(i, atencion);
33         GuardarCambios();
34     }
35     public int BuscarIndexAtencion(String codigo) {
36         for (int i = 0; i < Lista.size(); i++) {
37             if (Lista.get(i).getCodigo().contains(codigo)) {
38                 return i;
39             }
40         }
41         Mensajes.LeerMensaje("No se encontro el index de esa atencion");
42     }
43 }
```

Figura 15: Estructura para Atencion

```

44   public ArrayList<Atencion> Lista_Doctor(String codigo, String dni) {
45       ArrayList<Atencion> atencionesDoctor = new ArrayList<>();
46       for (Atencion atencion : Lista) {
47           if (atencion.getDoctor().equalsIgnoreCase(codigo) && atencion.getDoctor().getDni().equalsIgnoreCase(dni)) {
48               atencionesDoctor.add(atencion);
49           }
50       }
51       return atencionesDoctor;
52   }
53   public String CrearCodigoAtencion() {
54       int c1 = 5;
55       int c2 = Lista.size() + 1;
56
57       String codigo;
58       do {
59           codigo = "At" + c1 + "GT" + String.format("%03d", c2);
60           c2++;
61       } while (CodigoExiste(codigo));
62
63       return codigo;
64   }
65   private boolean CodigoExiste(String codigo) {
66       for (Atencion atencion : Lista) {
67           if (atencion.getCodigo().equals(codigo)) {
68               return true;
69           }
70       }
71       return false;
72   }
73   public ArrayList<Atencion> FiltrarPorEspecialidadYFechas(String especialidad, Date inicio, Date fin) {
74       return (ArrayList<Atencion>) Lista.stream()
75           .filter(a -> a.getEspecialidad_atencion().getNombre().equalsIgnoreCase(especialidad)
76               && (inicio == null || !a.getFecha().before(inicio))
77               && (fin == null || a.getFecha().after(fin)))
78           .collect(Collectors.toList());
79   }

```

Figura 16: Estructura para Atencion

```

public ArrayList<Atencion> FiltrarPorDoctorYFecha(String codDoctor, Date fecha) {
    if (Lista == null) {
        return new ArrayList<>();
    }

    return new ArrayList<>(Lista.stream()
        .filter(a -> a.getDoctor().equalsIgnoreCase(codDoctor)
            && truncarFecha(a.getFecha()).equals(truncarFecha(fecha)))
        .collect(Collectors.toList()));
}

private Date truncarFecha(Date fecha) {
    Calendar cal = Calendar.getInstance();
    cal.setTime(fecha);
    cal.set(Calendar.HOUR_OF_DAY, 0);
    cal.set(Calendar.MINUTE, 0);
    cal.set(Calendar.SECOND, 0);
    cal.set(Calendar.MILLISECOND, 0);
    return cal.getTime();
}

public ArrayList<Atencion> getList() {
    return Lista;
}
}

```

3.6. Listas Enlazadas Simples

Figura 17: Estructura para Doctor

```

1  package Estructura.Doctor;
2  import Modelo.Doctor;
3  import java.io.Serializable;
4  public class EnlaceNodo implements Serializable{
5      private Nodo ini;
6      private Nodo fin;
7      public EnlaceNodo(){
8          this.ini=null;
9          this.fin=null;
10     }
11    public void ActualizarDoctor(Nodo actual,Doctor empleadosActualizado){
12        actual.setdoctor(empleadosActualizado);
13    }
14    public void EliminarDoctor(Nodo actual){
15        Nodo anterior=ini;
16        while (anterior.getSig()!=actual && anterior.getSig()!=null) {
17            anterior=anterior.getSig();
18        }
19        if(actual!=null){
20            if(ini==actual){
21                ini=actual.getSig();
22            }else{
23                anterior.setSig(actual.getSig());
24            }
25        }
26    }
27    public void AgregarNodoInicio(Nodo Nuevo){
28        if(ini==null && fin==null){
29            ini=Nuevo;
30            fin=Nuevo;
31        }else{
32            Nuevo.setSig(ini);
33            ini=Nuevo;
34        }
35    }
36    public void AgregarNodoFinal(Nodo Nuevo){
37        if(ini==null && fin==null){
38            ini=Nuevo;
39            fin=Nuevo;
40        }else{
41            fin.setSig(Nuevo);
42        }
43        fin=Nuevo;
44        fin.setSig(null);
45    }

```

Figura 18: Estructura para Doctor

```

46   public boolean VerificarDuplicados(Doctor doctor){
47       Nodo aux=ini;
48       while(aux!=null){
49           if(aux.getdoctor().getCodigo().equals(doctor.getCodigo()) || aux.getdoctor().getDni().equals(doctor.getDni())){
50               return true;
51           }
52           aux=aux.getSig();
53       }
54       return false;
55   }
56   public Nodo BuscarCodigo(String codigo){
57       Nodo aux=ini;
58       while (aux!=null) {
59           if(aux.getdoctor().getCodigo().equalsIgnoreCase(codigo)){
60               return aux;
61           }
62           aux=aux.getSig();
63       }
64       return null;
65   }
66   public Nodo Buscar_Doctor_Sesion(String codigo,String dni){
67       Nodo aux=ini;
68       while (aux!=null) {
69           if(aux.getdoctor().getCodigo().equalsIgnoreCase(codigo) && aux.getdoctor().getDni().equalsIgnoreCase(dni)){
70               return aux;
71           }
72           aux=aux.getSig();
73       }
74       return null;
75   }
76   public boolean VerificarSesionDoctor(String codigo, String dni){
77       Nodo aux = ini;
78       while (aux != null) {
79           if(aux.getdoctor().getCodigo().equals(codigo) && aux.getdoctor().getDni().equals(dni)){
80               return true;
81           }
82           aux = aux.getSig();
83       }
84       return false;
85   }

```

Figura 19: Estructura para Doctor

```

1  package Estructura.Doctor;
2  import Modelo.Doctor;
3  import java.io.Serializable;
4  import Persistencia.Persistencia_Docente;
5  import Ordenamiento.OrdenamientoDoctor;
6  public class ListaNodoEnlazados implements Serializable{
7      private EnlaceNodo enlace;
8      private Persistencia_Docente persistencia;
9      public ListaNodoEnlazados() {
10         persistencia=new Persistencia_Docente();
11         enlace=persistencia.RecuperarEnlaceNodo();
12     }
13     private void GuardarCambios(){persistencia.GuardarEnlaceNodo(enlace);}
14     public void AñadirDoctorInicio(Doctor e){
15         Nodo nodo = new Nodo(e);
16         enlace.AgregarNodoInicio(nodo);
17         GuardarCambios();
18     }
19     public void AñadirDoctorFinal(Doctor e){
20         Nodo nodo = new Nodo(e);
21         enlace.AgregarNodoFinal(nodo);
22         GuardarCambios();
23     }
24     public void EliminarDoctor(String codigo){
25         Nodo Eliminar=enlace.BuscarCodigo(codigo);
26         enlace.EliminarDoctor(Eliminar);
27         GuardarCambios();
28     }
29     public void Actualizar(String codigo,Doctor Actualizar){
30         Nodo BuscarDescontinuado=enlace.BuscarCodigo(codigo);
31         enlace.ActualizarDoctor(BuscarDescontinuado, Actualizar);
32         GuardarCambios();
33     }
34     public Nodo Iniciar_Sesion_Doctor(String codigo,String dni){
35         return enlace.Buscar_Docente_Sesion(codigo, dni);
36     }
37     public boolean VerificarSesion(String codigo,String dni){
38         return enlace.VerificarSesionDoctor(codigo, dni);
39     }
40     public boolean VerificarDuplicados(Doctor d){
41         return enlace.VerificarDuplicados(d);
42     }
43     public Doctor BuscarDoctor(String codigo){
44         return enlace.BuscarCodigo(codigo).getdoctor();
45     }

```

Figura 20: Estructura para Doctor

```
43  public Doctor BuscarDoctor(String codigo){  
44      return enlace.BuscarCodigo(codigo).getdoctor();  
45  }  
46  public Nodo BuscarNodoDoctor(String codigo){  
47      return enlace.BuscarCodigo(codigo);  
48  }  
49  public EnlaceNodo getEnlace() {  
50      return enlace;  
51  }  
52  public void setEnlace(EnlaceNodo enlace) {  
53      this.enlace = enlace;  
54  }  
55  public Persistencia_Doctor getPersistencia() {  
56      return persistencia;  
57  }  
58  public void setPersistencia(Persistencia_Doctor persistencia) {  
59      this.persistencia = persistencia;  
60  }  
61  // Método para ordenar por nombre usando la clase OrdenamientoDoctor  
62  public void OrdenarPorNombre() {  
63      OrdenamientoDoctor.OrdenarPorNombre(enlace);  
64  }  
65 }
```

Figura 21: Estructura para Doctor

```
1  package Estructura.Doctor;  
2  import Modelo.Doctor;  
3  import java.io.Serializable;  
4  public class Nodo implements Serializable{  
5      private Doctor doctor;  
6      private Nodo sig;  
7      public Nodo(Doctor doctor) {  
8          this.doctor = doctor;  
9          this.sig = null;  
10     }  
11     public Doctor getdoctor() {return doctor;}  
12     public void setdoctor(Doctor doctor) {this.doctor = doctor;}  
13     public Nodo getsig() {return sig;}  
14     public void setsig(Nodo sig) {this.sig = sig;}  
15 }
```

3.7. Pilas

Figura 22: Estructura para Historia Clinica

```

1  package Estructura.HistoriaClinica;
2  import Modelo.HistoriaClinica;
3  import Persistencia.Persistencia_HistoriaClinica;
4  import Procesos.Mensajes;
5  import java.io.Serializable;
6  import java.util.Stack;
7  public class ListaPilaHistoriaClinica implements Serializable{
8      private Stack<HistoriaClinica> Pila;
9      private Persistencia_HistoriaClinica persistencia;
10     public ListaPilaHistoriaClinica() {
11         persistencia=new Persistencia_HistoriaClinica();
12         Pila=persistencia.RecupereHistoriaClinica();
13     }
14     private void GuardarCambios(){
15         persistencia.GuardarHistoriaClinica(Pila);
16     }
17     public void AgregarPila(HistoriaClinica historiaClinica){
18         Pila.push(historiaClinica);
19         GuardarCambios();
20     }
21     public void Despilar(){
22         Pila.pop();
23         GuardarCambios();
24     }
25     public void VerPrimerElemento(){
26         Mensajes.LeerMensaje(Pila.firstElement().toString());
27     }
28     public void VerUltimoElemento(){
29         Mensajes.LeerMensaje(Pila.peek().toString());
30     }
31     public HistoriaClinica BuscarPilaHistoriaClinica(String codigo){
32         for (HistoriaClinica historiaClinica : Pila) {
33             if(historiaClinica.getCodigo().contains(codigo)){
34                 return historiaClinica;
35             }
36         }
37         Mensajes.LeerMensaje("No se encontro la pila");
38         return null;
39     }
40     public boolean Detector_Duplicados(String dni_paciente){
41         for (HistoriaClinica historiaClinica : Pila) {
42             if(historiaClinica.getpaciente().getDni().equals(dni_paciente)){
43                 return true;
44             }
45         }
46         return false;
47     }

```

Figura 23: Estructura para Historia Clinica

```

48   public HistoriaClinica BuscarPilaHistoriaClinica_Paciente(String DNI_paciente) {
49     for (HistoriaClinica historiaClinica : Pila) {
50       if (historiaClinica.getpaciente().getDni().equals(DNI_paciente)) {
51         return historiaClinica;
52       }
53     }
54     Mensajes.LeerMensaje("No se encontro la pila");
55     return null;
56   }
57   public HistoriaClinica BuscarHistoria_Imprimir(String DNI_paciente, String Nombre_paciente) {
58     for (HistoriaClinica historiaClinica : Pila) {
59       if (historiaClinica.getpaciente().getDni().equals(DNI_paciente)&&
60           historiaClinica.getpaciente().getNombres().equals(Nombre_paciente)) {
61         return historiaClinica;
62       }
63     }
64     Mensajes.LeerMensaje("No se encontro la pila");
65     return null;
66   }
67   public void ActualizarPila(String codigo, HistoriaClinica historiaActualizar) {
68     for (int i = 0; i < Pila.size(); i++) {
69       if (Pila.get(i).getCodigo().equals(codigo)) {
70         Pila.set(i, historiaActualizar);
71         GuardarCambios();
72         Mensajes.LeerMensaje("Historia clínica actualizada correctamente.");
73         return;
74       }
75     }
76     Mensajes.LeerMensaje("No se encontró la historia clínica con el código especificado.");
77   }
78   public void ActualizarPila_Atencion(String codigo, HistoriaClinica historiaActualizar) {
79     for (int i = 0; i < Pila.size(); i++) {
80       if (Pila.get(i).getCodigo().equals(codigo)) {
81         Pila.set(i, historiaActualizar);
82         GuardarCambios();
83         return;
84       }
85     }
86     Mensajes.LeerMensaje("Hubo un error al juntar la atención con Historia Clinica");
87   }
88   public Stack<HistoriaClinica> getPila() {
89     return Pila;
90   }
91 }
```

3.8. Listas Circulares

Figura 24: Estructura para Paciente

```

1  package Estructura.Paciente;
2  import Modelo.Paciente;
3  import java.io.Serializable;
4  public class EnlaceCircular implements Serializable {
5      public Nodo lc;
6      public EnlaceCircular() {
7          lc = null;
8      }
9      public void ActualizarPaciente(String dnibuscado, Paciente nuevoPaciente) {
10         if (lc != null && nuevoPaciente != null && dnibuscado != null) { // Asegurarse de que no son nulos
11             Nodo actual = lc.enlace;
12             do {
13                 if (actual.pac.getDni().equals(dnibuscado)) {
14                     actual.pac = nuevoPaciente; // Actualizamos el paciente
15                     break;
16                 }
17                 actual = actual.enlace;
18             } while (actual != lc.enlace); // Recorrer hasta encontrar el nodo
19         }
20     }
21     public void InsertarFinal(Paciente pac) {
22         Nodo nuevo = new Nodo(pac);
23         if (lc == null) {
24             lc = nuevo;
25             lc.enlace = lc; // El primer nodo apunta a sí mismo
26         } else {
27             nuevo.enlace = lc.enlace; // El nuevo nodo apunta al primer nodo
28             lc.enlace = nuevo; // El último nodo apunta al nuevo nodo
29             lc = nuevo; // Actualizamos el último nodo a ser el nuevo nodo
30         }
31     }
32     public Nodo BuscarDni(String dnibuscado) {
33         if (lc != null && dnibuscado != null) {
34             Nodo actual = lc.enlace;
35             do {
36                 if (actual.pac.getDni().equals(dnibuscado)) {
37                     return actual;
38                 }
39                 actual = actual.enlace;
40             } while (actual != lc.enlace);
41         }
42         return null; // No se encuentra el paciente con el DNI buscado
43     }

```

Figura 25: Estructura para Paciente

```
44  public void EliminarNodo(String dnieliminar) {
45      if (lc == null || dnieliminar == null) {
46          return;
47      }
48      Nodo actual = lc.enlace;
49      Nodo anterior = lc;
50      do {
51          if (actual.pac.getDni().equals(dnieliminar)) {
52              if (actual == lc && actual.enlace == lc) { // Un solo nodo en la lista
53                  lc = null;
54              } else {
55                  anterior.enlace = actual.enlace; // El nodo anterior apunta al siguiente
56                  if (actual == lc) { // Si el nodo a eliminar es el último
57                      lc = anterior;
58                  }
59              }
60              break; // Nodo encontrado y eliminado
61          }
62          anterior = actual;
63          actual = actual.enlace;
64      } while (actual != lc.enlace); // Recorrer toda la lista
65  }
66  public void OrdenarPorNombre() {
67      if (lc == null || lc.enlace == lc) {
68          return; // Lista vacía o con un solo elemento, no es necesario ordenar
69      }
70      boolean huboIntercambio;
71      do {
72          Nodo actual = lc.enlace;
73          Nodo siguiente = actual.enlace;
74          huboIntercambio = false;
75          do {
76              if (actual.pac.getNombres().compareTo(siguiente.pac.getNombres()) > 0) {
77                  // Intercambiamos los pacientes entre los nodos
78                  Paciente temp = actual.pac;
79                  actual.pac = siguiente.pac;
80                  siguiente.pac = temp;
81                  huboIntercambio = true; // Hubo un intercambio, seguimos ordenando
82              }
83              actual = siguiente;
84              siguiente = siguiente.enlace;
85          } while (siguiente != lc.enlace);
86      } while (huboIntercambio); // Repetimos mientras haya intercambios
87  }
88
89 }
```

Figura 26: Estructura para Paciente

```

1  package Estructura.Paciente;
2  import Modelo.Paciente;
3  import Ordenamiento.OrdenamientoPacientes;
4  import Persistencia.Persistencia_Paciente;
5  import Procesos.Mensajes;
6  import java.io.Serializable;
7  public class ListaCircularPaciente implements Serializable {
8      private EnlaceCircular enlaceCircular;
9      private Persistencia_Paciente persistencia;
10     public ListaCircularPaciente() {
11         persistencia = new Persistencia_Paciente();
12         enlaceCircular = persistencia.recuperarLista();
13     }
14     private void GuardaCambios() {
15         persistencia.guardarLista(enlaceCircular);
16     }
17     public void AgregarPaciente(Paciente paciente) {
18         enlaceCircular.InsertarFinal(paciente);
19         GuardaCambios();
20     }
21     public Nodo BuscarNodoPaciente(String dni) {
22         if (enlaceCircular.BuscarDni(dni) != null) {
23             return enlaceCircular.BuscarDni(dni);
24         }
25         Mensajes.LeerMensaje("Paciente no encontrado");
26         return null;
27     }
28     public void EliminarNodoPaciente(String dni) {
29         enlaceCircular.EliminarNodo(dni);
30         GuardaCambios();
31     }
32     public void ActualizarNodoPaciente(String dni, Paciente pacienteNuevo) {
33         enlaceCircular.ActualizarPaciente(dni, pacienteNuevo);
34         GuardaCambios();
35     }
36     public EnlaceCircular getEnlaceCircular() {
37         return enlaceCircular;
38     }
39     public void setEnlaceCircular(EnlaceCircular enlaceCircular) {
40         this.enlaceCircular = enlaceCircular;
41     }
42     // Ordenar por nombre
43     public void OrdenarPacientesPorNombre() {
44         OrdenamientoPacientes.OrdenarPorNombre(enlaceCircular);
45         GuardaCambios();
46         Mensajes.LeerMensaje("Pacientes ordenados por nombre.");
47     }

```

Figura 27: Estructura para Paciente

```

48     // Ordenar por edad
49     public void OrdenarPacientesPorEdad() {
50         OrdenamientoPacientes.OrdenarPorEdad(enlaceCircular);
51         GuardaCambios();
52     }
53 }
54

```

Figura 28: Estructura para Paciente

```
1 package Estructura.Paciente;
2 import java.io.Serializable;
3 import Modelo.Paciente;
4 public class Nodo implements Serializable{
5     public Paciente pac;
6     public Nodo enlace;
7     public Nodo(Paciente pac){
8         this.pac = pac;
9         enlace = this;
10    }
11 }
```

3.9. Persistencias de datos

Figura 29: Persistencia de Atencion

```
1 package Persistencia;
2 import Modelo.Atencion;
3 import Modelo.Especialidad;
4 import Procesos.Mensajes;
5 import java.io.FileInputStream;
6 import java.io.FileOutputStream;
7 import java.io.ObjectInputStream;
8 import java.io.ObjectOutputStream;
9 import java.util.ArrayList;
10 public class Persistencia_Atencion {
11     private String Archivo="Atenciones";
12     public void GuardarAtenciones(ArrayList<Atencion>Lista){
13         try{
14             FileOutputStream fos = new FileOutputStream(Archivo);
15             ObjectOutputStream oos = new ObjectOutputStream(fos);
16             oos.writeObject(Lista);
17             oos.close();
18             System.out.println("Se guardo el arraylist con exito");
19         }catch(Exception e){
20             Mensajes.LeerMensaje("Ocurrio un error al guardar Atenciones: "+e);
21         }
22     }
23     public ArrayList<Atencion> RecupereAtenciones(){
24         ArrayList<Atencion>Lista = new ArrayList<>();
25         try{
26             FileInputStream fis = new FileInputStream(Archivo);
27             ObjectInputStream ois = new ObjectInputStream(fis);
28             Lista=(ArrayList<Atencion>)ois.readObject();
29             ois.close();
30             System.out.println("Se recuperó la lista con éxito");
31         }catch(Exception e){
32             Mensajes.LeerMensaje("Ocurrió un error al recuperar Atenciones: "+e);
33         }
34     }
35 }
36 }
```

Figura 30: Persistencia de Doctor

```
1 package Persistencia;
2 import Estructura.Doctor.EnlazeNodo;
3 import java.io.FileInputStream;
4 import java.io.FileOutputStream;
5 import java.io.ObjectInputStream;
6 import java.io.ObjectOutputStream;
7 import javax.swing.JOptionPane;
8 public class Persistencia_Docor {
9     private static String Archivo="Doctores";
10    public void GuardarEnlazeNodo(EnlazeNodo enlace){
11        try{
12            FileOutputStream fos = new FileOutputStream(Archivo);
13            ObjectOutputStream oos = new ObjectOutputStream(fos);
14            oos.writeObject(enlace);
15            oos.close();
16        }catch(Exception e){
17            JOptionPane.showMessageDialog(null, "Hubo un error al guardar el enlace: "+e);
18        }
19    }
20    public EnlazeNodo RecuperarEnlazeNodo(){
21        EnlazeNodo enlaceNodo = new EnlazeNodo();
22        try{
23            FileInputStream fis = new FileInputStream(Archivo);
24            ObjectInputStream ois = new ObjectInputStream(fis);
25            enlaceNodo=(EnlazeNodo)ois.readObject();
26            ois.close();
27        }catch(Exception e){
28            JOptionPane.showMessageDialog(null, "Hubo un error al recuperar el enlace: "+e);
29        }
30        return enlaceNodo;
31    }
32}
```

Figura 31: Persistencia de Especialidad

```
1 package Persistencia;
2 import Modelo.Atencion;
3 import Modelo.Especialidad;
4 import Procesos.Mensajes;
5 import java.io.FileInputStream;
6 import java.io.FileOutputStream;
7 import java.io.ObjectInputStream;
8 import java.io.ObjectOutputStream;
9 public class Persistencia_Especialidad {
10     private String Archivo="Especialidad";
11     public void GuardarEspecialidad(Especialidad[]Lista){
12         try{
13             FileOutputStream fos = new FileOutputStream(Archivo);
14             ObjectOutputStream oos = new ObjectOutputStream(fos);
15             oos.writeObject(Lista);
16             oos.close();
17             System.out.println("Se guardo la lista con exito");
18         }catch(Exception e){
19             Mensajes.LeerMensaje("Ocurrio un error al guardar Especialidad: "+e);
20         }
21     }
22     public Especialidad[] RecupereEspecialidad(int num){
23         Especialidad[]Lista = new Especialidad[num];
24         try{
25             FileInputStream fis = new FileInputStream(Archivo);
26             ObjectInputStream ois = new ObjectInputStream(fis);
27             Lista=(Especialidad[])ois.readObject();
28             ois.close();
29             System.out.println("Se recuperó la lista con exito");
30         }catch(Exception e){
31             Mensajes.LeerMensaje("Ocurrio un error al recuperar Especialidad: "+e);
32         }
33     }
34 }
35 }
```

Figura 32: Persistencia de Historia Clinica

```

1 package Persistencia;
2 import Modelo.HistoriaClinica;
3 import Procesos.Mensajes;
4 import java.io.FileInputStream;
5 import java.io.FileOutputStream;
6 import java.io.ObjectInputStream;
7 import java.io.ObjectOutputStream;
8 import java.util.Stack;
9 public class Persistencia_HistoriaClinica {
10     private static String Archivo="HistoriasClinicas";
11     public void GuardarHistoriaClinica(Stack<HistoriaClinica>Pila){
12         try{
13             FileOutputStream fos = new FileOutputStream(Archivo);
14             ObjectOutputStream oos = new ObjectOutputStream(fos);
15             oos.writeObject(Pila);
16             oos.close();
17             System.out.println("Se guardo la pila con exito");
18         }catch(Exception e){
19             Mensajes.LeerMensaje("Ocurrio un error al guardar HistoriasClinicas: "+e);
20         }
21     }
22     public Stack<HistoriaClinica> RecupereHistoriaClinica(){
23         Stack<HistoriaClinica>Pila=new Stack<>();
24         try{
25             FileInputStream fis = new FileInputStream(Archivo);
26             ObjectInputStream ois = new ObjectInputStream(fis);
27             Pila=(Stack<HistoriaClinica>)ois.readObject();
28             ois.close();
29             System.out.println("Se recuperó la pila con éxito");
30         }catch(Exception e){
31             Mensajes.LeerMensaje("Ocurrio un error al recuperar HistoriasClinicas: "+e);
32         }
33         return Pila;
34     }
35 }
```

Figura 33: Persistencia de Paciente

```
1 package Persistencia;
2 import Estructura.Paciente.EnlaceCircular;
3 import java.io.FileInputStream;
4 import java.io.FileOutputStream;
5 import java.io.ObjectInputStream;
6 import java.io.ObjectOutputStream;
7 import Procesos.Mensajes;
8 public class Persistencia_Paciente {
9     public static String ARCHIVO = "DatosListaPacientes.bin";
10
11    public void guardarLista(EnlaceCircular Lista){
12        try {
13            FileOutputStream fos = new FileOutputStream(ARCHIVO);
14            ObjectOutputStream oos = new ObjectOutputStream(fos);
15            oos.writeObject(Lista);
16            oos.close();
17        } catch (Exception e) {
18            Mensajes.LeerMensaje("ERROR no se puede guardar "+e);
19        }
20    }
21    public EnlaceCircular recuperarLista(){
22        EnlaceCircular Lista = new EnlaceCircular();
23        try {
24            FileInputStream fis = new FileInputStream(ARCHIVO);
25            ObjectInputStream ois = new ObjectInputStream(fis);
26            Lista = (EnlaceCircular) ois.readObject();
27        } catch (Exception e) {
28            Mensajes.LeerMensaje("ERROR no se puede recuperar... "+e);
29        }
30        return Lista;
31    }
32}
```

3.10. Evidencia de imágenes de la funcionalidad de cada RF

Figura 34: Internal Frame de Pacientes

ADMIN : Aplicación del curso de Algoritmos y Estructuras de Datos-Sección 24606

PACIENTE DOCTORES ESPECIALIDADES ATENCIONES HISTORIAS CLINICAS BUSQUEDAS CERRAR SESIÓN

Gestion Pacientes

DATOS DE PACIENTES

DNI	ID	DNI	NOMBRE	TELEFONO	EDAD	DIRECCION	GENERO	FECHA NAC.
58963214	1	47896521	Juan Carlos López	987654321	39	Av. Las Palmeras ...	Hombre	14/05/1985
69874125	2	58963214	Maria Fernanda	912345678	34	Jr. Los Cedros 4...	Mujer	14/08/1990
85236974	3	69874125	Pedro Ramírez	998741235	37	Calle Las Flores ...	Hombre	10/06/1987
96325974	4	85236974	Sofía Martínez S...	965974123	29	Jr. Primavera 23...	Mujer	02/10/1995
87456932	5	96325974	Ricardo Gutiérre...	976543219	36	Av. Independenc...	Hombre	11/12/1988
74589632	6	87456932	Diana Vargas P...	986321547	31	Jr. San Martín 89...	Mujer	11/02/1993
36985241	7	74589632	Roberto Fernández	912365478	39	Calle Pardo 678...	Hombre	03/12/1985
69857412	8	36985241	Laura González	987214563	27	Jr. Los Almendr...	Mujer	01/09/1997
58741236	9	69857412	José Antonio Ca...	921478563	35	Av. Amazonas 2...	Hombre	22/03/1989
	10	58741236	Gabriela Soto Vil...	934785612	28	Jr. La Libertad 3...	Mujer	01/01/1996

AÑADIR ACTUALIZAR
ELIMINAR BUSCAR
Por Nombres
Selección FILTRAR

Figura 35: Internal Frame de Doctores

ADMIN : Aplicación del curso de Algoritmos y Estructuras de Datos-Sección 24606

PACIENTE DOCTORES ESPECIALIDADES ATENCIONES HISTORIAS CLINICAS BUSQUEDAS CERRAR SESIÓN

Gestion de Doctor

DATOS DE DOCTOR

CODIGO	ID	DNI	CODIGO	NOMBRE	TELEFONO	DIRECCION
65478932	1	74896321	D1001	Juan Pérez López	987654321	Av. Principal 123, Lima
78965412	2	65478932	D1002	Maria Torres García	912345678	Jr. Los Cedros 456, Are...
89562341	3	89562341	D1003	Carlos Mendoza Ríos	998745632	Calle Las Flores 789, Li...
65874129	4	78965412	D1004	Ana Fernández Ruiz	965974123	Jr. San Martín 234, Lima
78451236	5	65874129	D1005	Luis Chávez Campos	976543219	Av. Independencia 567, ...
45678913	6	78451236	D1006	Rosa Gutiérrez Salazar	986321547	Calle Pardo 890, Lima
96325974	7	45678913	D1007	Jorge Castro Vega	987123654	Av. La Marina 345, Lima
85236974	8	96325974	D1008	Elena Rojas Fernández	923456789	Calle Los Pinos 678, Li...
74125836	9	85236974	D1009	Ricardo Salazar Gómez	954321678	Jr. Primavera 234, Trujillo
	10	74125836	D1010	Sofía Mendoza Palacios	965478321	Av. Amazonas 890, Lima

BUSCAR AÑADIR INCIO
ACTUALIZAR AÑADIR FINAL
ELIMINAR FILTRAR
Por Nombres
Selección

Figura 36: Internal Frame de Atenciones

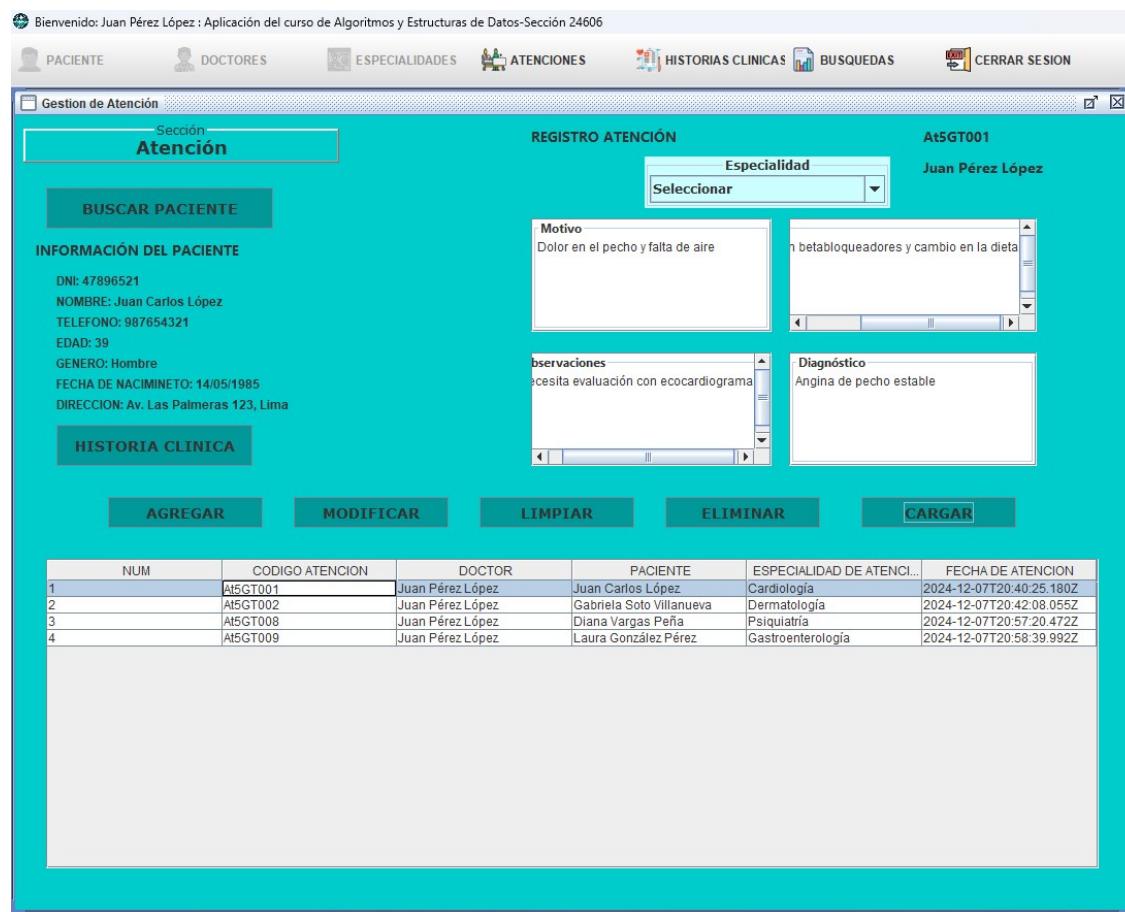


Figura 37: Internal Frame de Especialidad

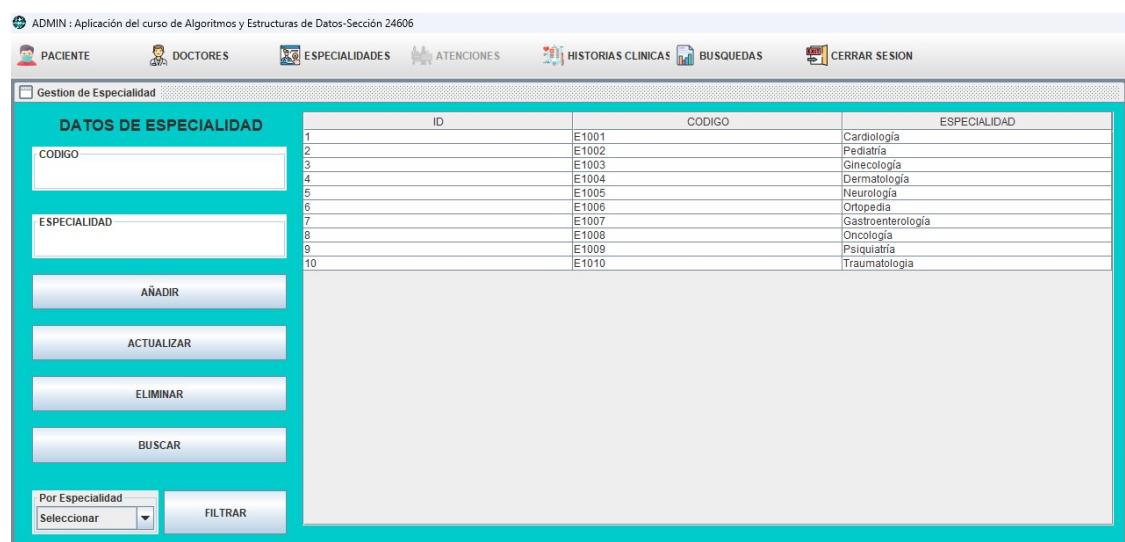


Figura 38: Internal Frame de Historia Clínica

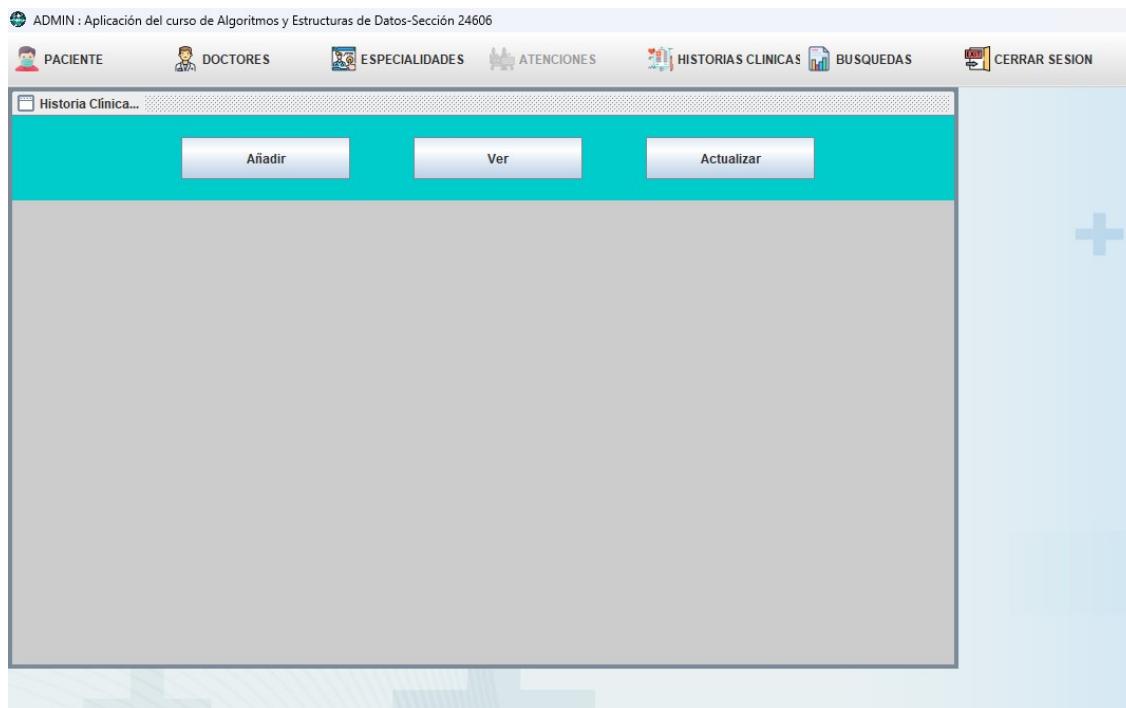


Figura 39: Internal Frame de Historia Clínica

The screenshot shows the internal frame of a web application titled "Historia Clínica...". The top navigation bar includes links for PACIENTE, DOCTORES, ESPECIALIDADES, ATENCIÓNES, HISTORIAS CLÍNICAS, BUSQUEDAS, and CERRAR SESIÓN. Below the navigation is a teal header bar with buttons for "Añadir", "Ver", and "Actualizar". Underneath is a search section with "BUSCAR" and "ELIMINAR" buttons. A table displays a list of clinical histories:

ID	CODIGO	PACIENTE	FECHA
1	HC1001	Juan Carlos López	Sat Dec 07 16:07:00 PET 2024
2	HC1002	Maria Fernanda Torres	Sat Dec 07 16:08:37 PET 2024
3	HC1003	Pedro Ramírez Castillo	Sat Dec 07 16:11:03 PET 2024
4	HC1004	Sofía Martínez Salazar	Sat Dec 07 16:12:26 PET 2024
5	HC1005	Ricardo Gutiérrez Mendoza	Sat Dec 07 16:13:59 PET 2024

Figura 40: Internal Frame de Historia Clínica

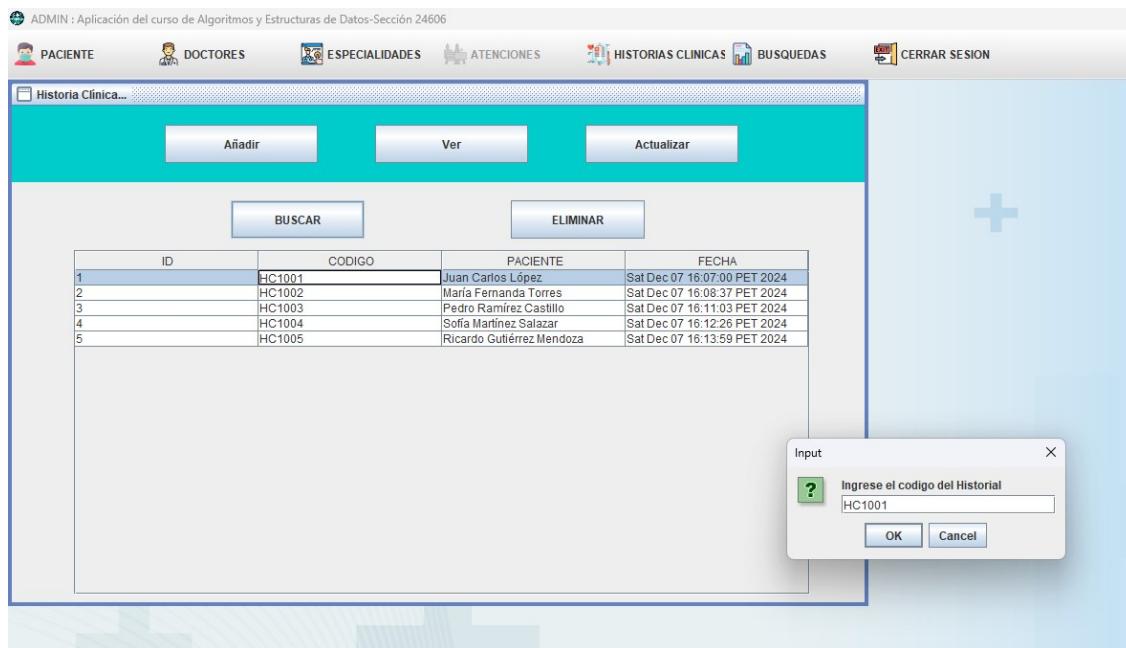


Figura 41: Internal Frame de Historia Clínica

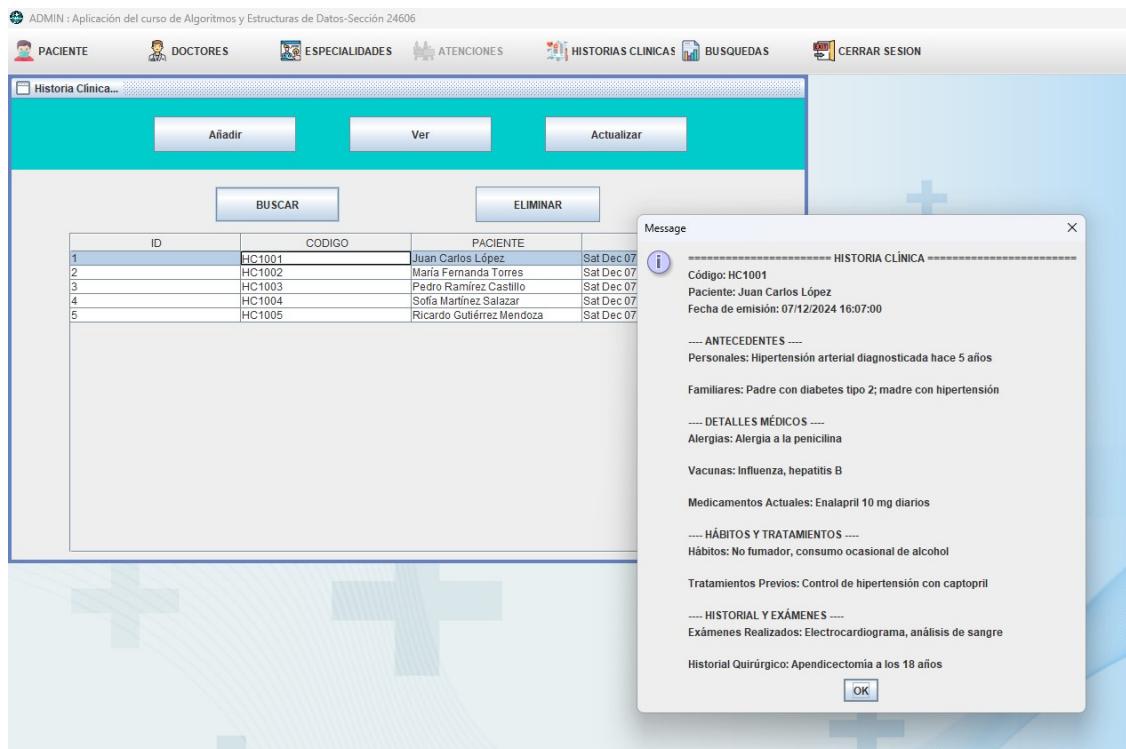


Figura 42: Internal Frame de Historia Clínica

ADMIN : Aplicación del curso de Algoritmos y Estructuras de Datos-Sección 24606

PACIENTE DOCTORES ESPECIALIDADES ATENCIÓNES HISTORIAS CLÍNICAS BUSQUEDAS CERRAR SESIÓN

Historia Clínica...

Añadir Ver Actualizar

NOMBRE: Juan Carlos López
DNI: 47896521
GENERO: Hombre

ANTECEDENTES PERSONALES: Hipertensión arterial diagnosticada hace 5 años.

MEDICAMENTOS ACTUALES: Enalapril 10 mg diarios.

HABITOS: No fumador, consumo ocasional de alcohol.

ANTECEDENTES FAMILIARES: Padre con diabetes tipo 2; madre con hipertensión.

EXAMENES REALIZADOS: Electrocardiograma, análisis de sangre.

TRATAMINETOS PREVIOS: Control de hipertensión con captopril.

ALERGIAS: Alergia a la penicilina.

VACUNAS: Influenza, hepatitis B.

HISTORIAL QUIRURJICO: Apendicectomía a los 18 años.

Figura 43: Internal Frame de Historia Clínica

ADMIN : Aplicación del curso de Algoritmos y Estructuras de Datos-Sección 24606

PACIENTE DOCTORES ESPECIALIDADES ATENCIÓNES HISTORIAS CLÍNICAS BUSQUEDAS CERRAR SESIÓN

Imprimir Historia Clínica

Sección: Historias Clínicas

Nombre del paciente: Juan Carlos López

Dni del paciente: 47896521

BUSCAR

DETALLES:

===== HISTORIA CLÍNICA =====

Código: HC1001
Paciente: Juan Carlos López
Fecha de emisión: 07/12/2024 16:07:00

---- ANTECEDENTES ----
Personales: Hipertensión arterial diagnosticada hace 5 años
Familiares: Padre con diabetes tipo 2; madre con hipertensión

---- DETALLES MÉDICOS ----
Alergias: Alergia a la penicilina

Vacunas: Influenza, hepatitis B

Medicamentos Actuales: Enalapril 10 mg diarios

---- HÁBITOS Y TRATAMIENTOS ----

IMPRIMIR ATENCIÓNES PDF IMPRIMIR HISTORIA CLÍNICA PDF

Figura 44: Generación de Pdf



HISTORIA CLINICA

DATOS PERSONALES:

Código de Historia: HC1001

Dni: 47896521

Nombres: Juan Carlos López

Telef: 987654321

Fecha de Nacimiento: 14/05/1985

Edad: 39

Sexo: Hombre

Domicilio: Av. Las Palmeras 123, Lima

Fecha de emisión: 07/12/2024 16:07:00

ANTECEDENTES:

Personales: Hipertensión arterial diagnosticada hace 5 años

Familiares: Padre con diabetes tipo 2; madre con hipertensión

DETALLES MÉDICOS:

Alergias: Alergia a la penicilina

Vacunas: Influenza, hepatitis B

Medicamentos Actuales: Enalapril 10 mg diarios

HÁBITOS Y TRATAMIENTOS:

Hábitos: No fumador, consumo ocasional de alcohol

Tratamientos Previos: Control de hipertensión con captopril

HISTORIAL Y EXÁMENES:

Exámenes Realizados: Electrocardiograma, análisis de sangre

Historial Quirúrgico: Apendicectomía a los 18 años

Figura 45: Generación de Pdf

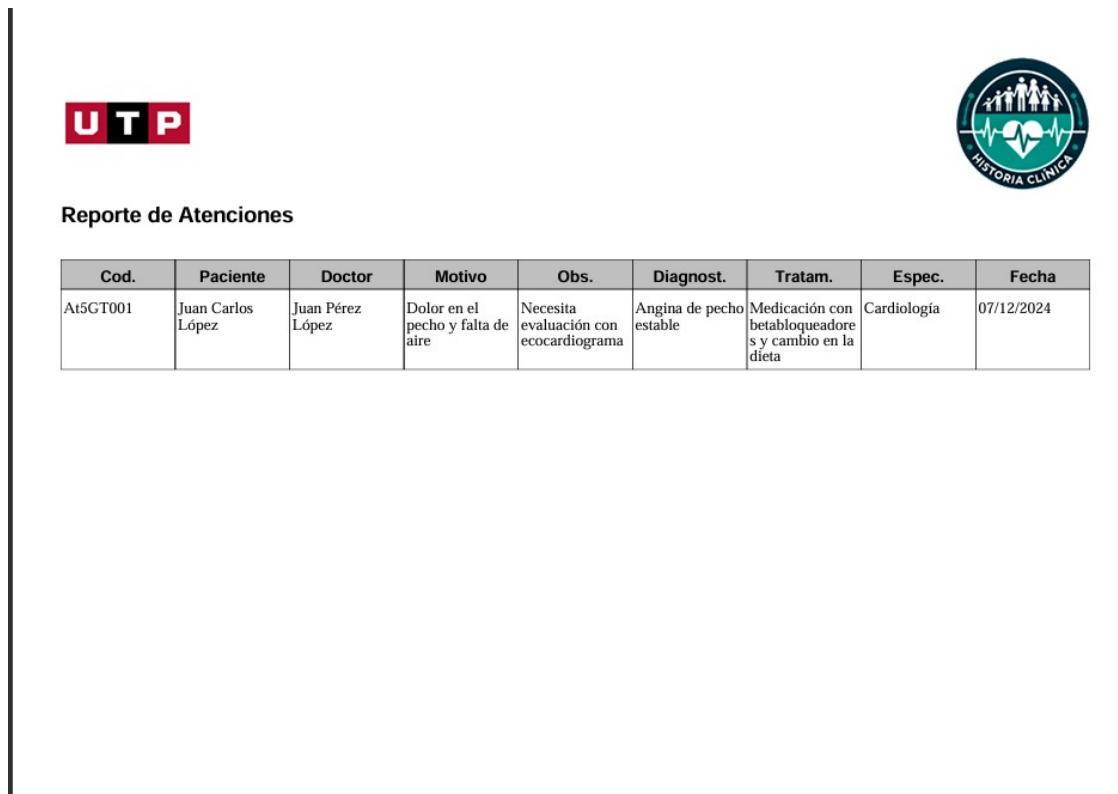


Figura 46: Búsqueda por Doctor para atención durante el día

Código de Atención	DNI Paciente	Nombre Paciente	Fecha de Atención	Especialidad	Doctor
AT5GT001	47880521	Juan Carlos López	07/12/2024	Cardiología	Juan Pérez López
AT5GT002	58741236	Gabriela Soto Villanueva	07/12/2024	Dermatología	Juan Pérez López
AT5GT008	87456932	Diana Vargas Peña	07/12/2024	Psiquiatría	Juan Pérez López
AT5GT009	36985241	Laura González Pérez	07/12/2024	Gastroenterología	Juan Pérez López

Message

Búsqueda realizada con éxito para el médico con código: D1001

OK

Figura 47: Búsqueda por Doctor para atención durante el día

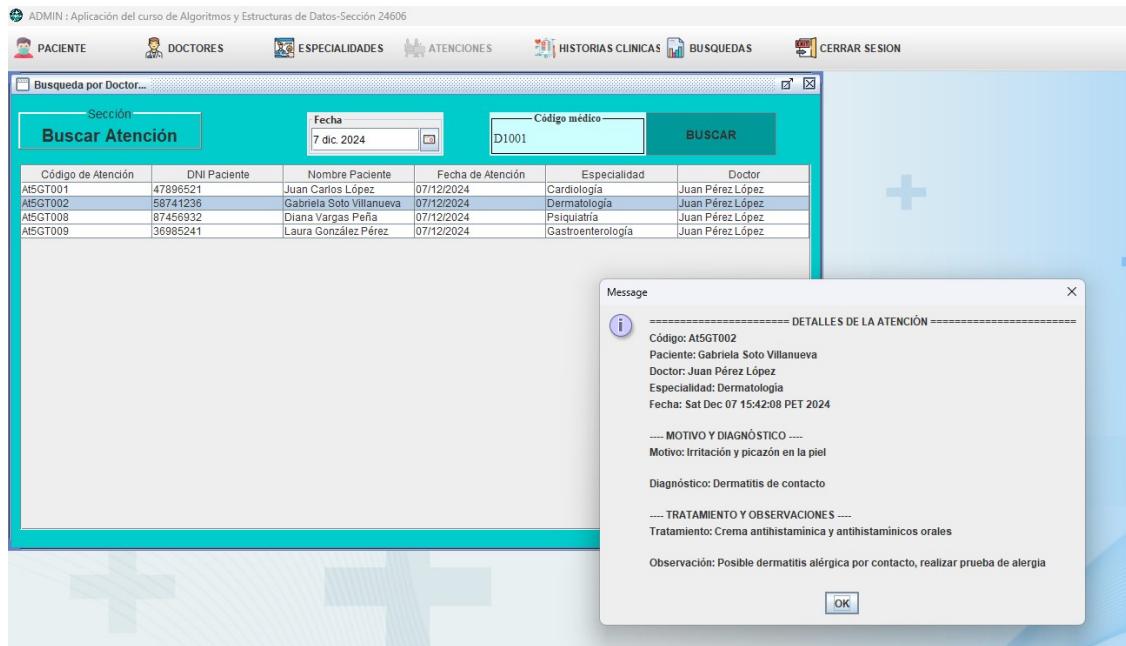


Figura 48: Búsqueda por Especialidad para atención por rango de fecha

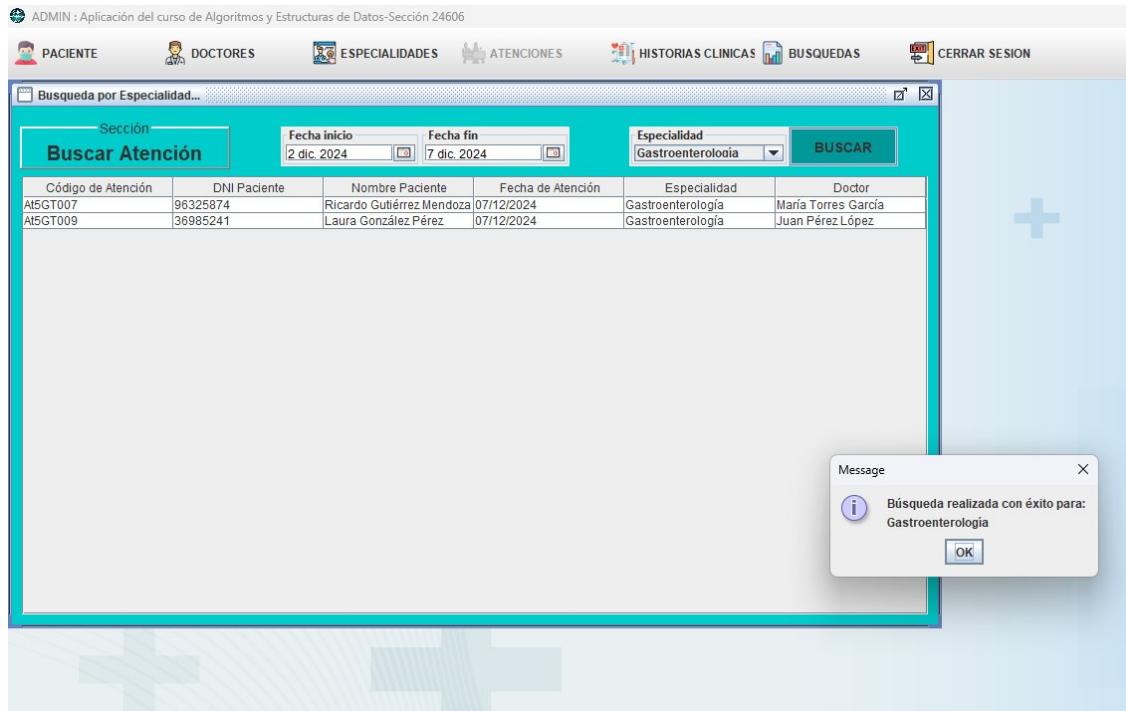
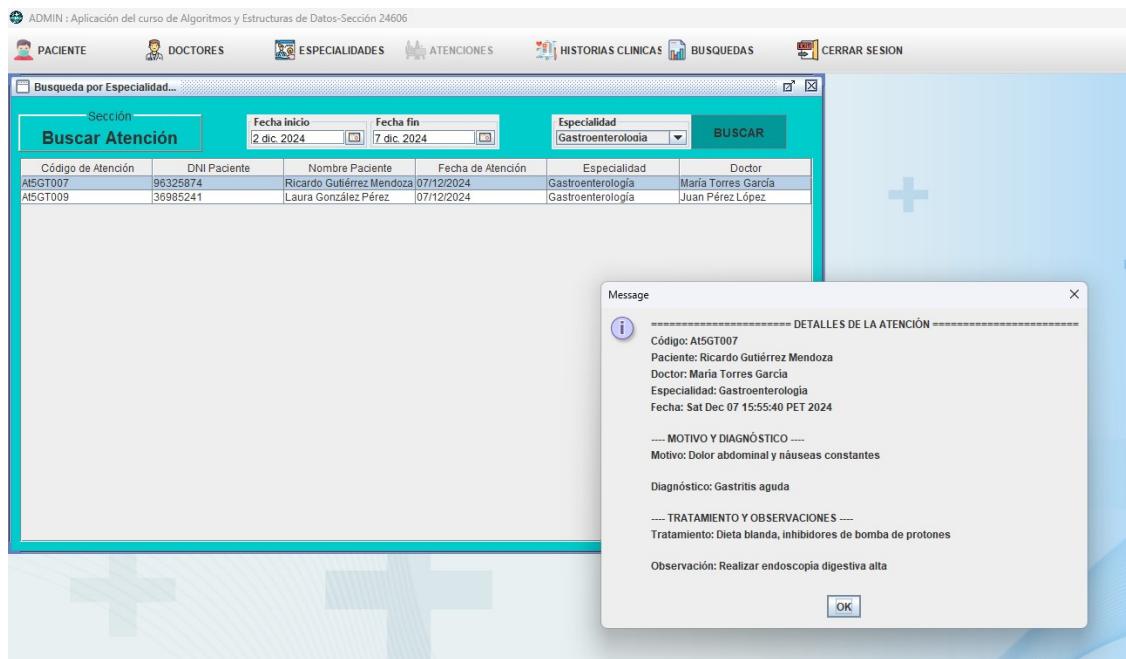


Figura 49: Búsqueda por Doctor para atención durante el día



4. Capítulo 4: Conclusiones y Recomendaciones

4.1. Conclusiones

- La implementación de un aplicativo para el registro y consulta de historias clínicas de manera digital permitirá a la Posta Médica gestionar de forma más eficiente la información de los pacientes, reduciendo el manejo manual y el riesgo de errores humanos en la atención médica.
- Al eliminar la necesidad de búsqueda manual de datos, el tiempo de atención se optimiza significativamente. Los profesionales de la salud podrán acceder rápidamente a la información médica de los pacientes, lo que mejora la calidad del servicio y acelera el proceso de atención.
- La automatización del registro de historias clínicas ayuda a reducir los errores en la recopilación de datos médicos. La estandarización del proceso de entrada y consulta de datos garantiza una mayor precisión en el registro de la información del paciente.
- El sistema digital facilita la generación de informes y estadísticas médicas, lo que será esencial

para el seguimiento de los pacientes y la toma de decisiones clínicas informadas. Esto optimiza la gestión del cuidado de salud a largo plazo.

- La implementación de archivos binarios como solución para almacenar y gestionar los datos médicos simplifica el proceso sin la necesidad de una base de datos compleja, permitiendo una solución ligera, eficiente y de fácil implementación en entornos con recursos limitados.

4.2. Recomendaciones

- Es fundamental proporcionar capacitación adecuada al personal médico y administrativo de la Posta Médica sobre el uso del nuevo sistema. Esto asegurará que el aplicativo se utilice de manera eficiente y que se maximicen sus beneficios.

- Asegurarse de que el sistema sea mantenido y actualizado regularmente para corregir posibles fallos y adaptarse a cambios en las normativas médicas o en las necesidades de los usuarios.

Es importante contar con un plan de soporte técnico continuo.

- Se debe establecer un proceso de monitoreo continuo para evaluar la efectividad del sistema en cuanto a la reducción de errores, tiempos de atención y mejora de la calidad del servicio.

Esto permitirá realizar ajustes y mejoras conforme sea necesario para satisfacer las demandas y expectativas de los usuarios.

5. Referencias Bibliográficas

Referencias

- [1] Mejía y Quiroga (2014). *Impacto de la digitalización en la gestión de historias clínicas en centros de salud.* (USMP) Revista de Sistemas Médicos, 10(3), 145-160. Recuperado de **Link**