

TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE OAXACA
SISTEMAS Y COMPUTACIÓN

INGENIERÍA EN SISTEMAS COMPUTACIONALES

INGENIERÍA EN SOFTWARE

GARCIA OSORIO BOLIVAR

HIDALGO TEJADA YEUDIEL AUGUSTO

VALENTÍN SÁNCHEZ GERSON MERARI

**PROYECTO DEL SISTEMA DE LA CLÍNICA SAGRADO
CORAZÓN**

ESPINOSA PÉREZ JACOB

“6SU” - 19:00-20:00

ÍNDICE	4
CAPÍTULO 1 MARCO CONTEXTUAL	5
1.1 Generalidades de la Institución	5
1.1.1 Nombre de la Institución	5
1.1.2 Misión	5
1.1.3 Visión	5
1.1.4 Giro de la empresa	5
1.1.5 Dirección de la empresa	6
1.1.6 Organigrama de la Empresa	6
1.1.7 DESCRIPCIÓN DE PROCESOS DE LA EMPRESA	6
1.1.8 Área donde se desarrolló el proyecto	6
1.2 GENERALIDADES DEL PROYECTO	7
1.2.1 Nombre del Proyecto	7
1.2.2 Planteamiento del Problema	7
1.2.3 OBJETIVOS	7
1.2.3.1 Objetivo General	7
1.2.3.2 Objetivos Específicos	7
1.2.4 Justificación	8
1.2.5 ALCANCES Y LIMITACIONES	8
1.2.5.1 Alcances	8
1.2.5.2 Limitaciones	9
1.2.6 Cronograma de Actividades	9
CAPÍTULO 2 MARCO METODOLÓGICO	10
2.1 Metodología de Desarrollo	11
2.2 Estudio de Factibilidad	12
Actualizaciones y mantenimiento:	12
2.2.1 Estudio Operacional	13
2.2.2 Estudio Económico	14
2.3 Herramientas Empleadas	15
2.4 Fase de Exploración	16
2.4.1 ENCUESTA	18
2.4.2 Información de observación (Encuesta)	19
2.4.3 Diagramas BPMN	23
2.5 Fase de Planeación	25
2.5.1 HISTORIAS DE USUARIO	34
2.5.2 PLAN DE ENTREGAS	37
2.5.3 VELOCIDAD DE PROYECTO	37
2.6 FASE DE ITERACIONES	38
2.6.1 PLANIFICACIÓN	38
2.7 FASE DE PRODUCCIÓN	74
2.8 FASE DE MANTENIMIENTO	75
2.9 FASE DE MUERTE	76

ÍNDICE DE TABLAS

- Tabla de Cronograma de Actividades
- Tabla de Estudio de Factibilidad
- Tabla de Estudio Operacional
- Tabla de Estudio Económico
- Tabla de Fase de Planeación
- Tabla de Planificación 2da etapa
- Tabla de Lista Maestra de Historias de Usuario
- Tabla de 1ER ITERACIÓN
- Tabla de 2DA ITERACIÓN
- Tabla de 3RA ITERACIÓN
- Tabla de Tareas Establecidas por Cada Iteración
- Tabla de PLAN DE ENTREGAS
- Tabla de VELOCIDAD DE PROYECTO
- Tabla de Historia de usuario 1
- Tabla de FASE DISEÑO TARJETAS CRC
- Tabla de PRUEBAS (CASOS DE PRUEBAS)
- Tabla de 2DA Historia de usuario
- Tabla de FASE DISEÑO TARJETAS CRC
- Tabla de PRUEBAS (CASOS DE PRUEBAS)
- Tabla de 3era Historia de usuario
- Tabla de PRUEBAS (CASOS DE PRUEBAS)
- Tabla de TARJETA CRC
- Tabla de 3RA ITERACIÓN
- Tabla de TARJETA CRC
- Tabla de Historia de usuario 6
- Tabla de FASE DISEÑO TARJETAS CRC
- Tabla de PRUEBAS (CASOS DE PRUEBAS)

ÍNDICE DE FIGURAS

- Figura 1 Organigrama de la Empresa
- Figura 2 ENCUESTA
- Figura 3 Información de observación (Encuesta)
- Figura 4 Diagramas BPMN 1
- Figura 5 Diagramas BPMN 2
- Figura 6 Diagramas BPMN 3
- Figura 7 Diagramas BPMN 4
- Figura 8 Diagramas BPMN 5
- Figura 9 Diagramas BPMN 6
- Figura 10 HISTORIA DE USUARIO 1
- Figura 11 HISTORIA DE USUARIO 2
- Figura 12 HISTORIA DE USUARIO 3
- Figura 13 HISTORIA DE USUARIO 4
- Figura 14 HISTORIA DE USUARIO 5
- Figura 15 HISTORIA DE USUARIO 6
- Figura 16 MODELO ENTIDAD RELACIÓN 1
- Figura 17 MODELO RELACIONAL 1
- Figura 18 DISEÑO 1
- Figura 19 DISEÑO 2
- Figura 20 DISEÑO 3
- Figura 21 DISEÑO 4
- Figura 22 DESARROLLO 1
- Figura 23 DESARROLLO 2
- Figura 24 DESARROLLO 3
- Figura 25 DESARROLLO 4
- Figura 26 DESARROLLO 5

INTRODUCCIÓN

En el ámbito de la salud, la optimización de la gestión de información y la mejora de los procesos internos son factores clave para ofrecer una atención médica de calidad y centrada en el paciente. La Clínica Sagrado Corazón, al igual que muchas otras instituciones de salud, enfrenta desafíos en la administración eficiente de los registros de pacientes, el control de procesos y la gestión de citas médicas. Estos problemas pueden generar errores, demoras y costos adicionales, afectando negativamente la calidad de los servicios brindados.

Ante esta situación, surge la necesidad de desarrollar un sistema integral que automatice y optimice estos procesos, permitiendo a la clínica mejorar su eficiencia operativa y la calidad de la atención que ofrece a los pacientes. Este proyecto tiene como objetivo el diseño e implementación de un sistema de registro y control para la Clínica Sagrado Corazón, que facilite el registro de pacientes, la gestión de citas y el seguimiento de tratamientos, además de integrar herramientas para la gestión interna.

CAPÍTULO 1 MARCO CONTEXTUAL

1.1 Generalidades de la Institución

Se centra en la prestación de servicios de salud de alta complejidad, con un enfoque particular en la atención segura, de calidad y centrada en las personas. La institución se dedica a proporcionar diagnósticos, tratamientos y cuidados médicos avanzados para una amplia gama de enfermedades y condiciones médicas. Sus servicios abarcan diversas especialidades médicas, procedimientos quirúrgicos, terapias y tratamientos especializados.

1.1.1 Nombre de la Institución

INSTITUTO TECNOLÓGICO DE OAXACA

1.1.2 Misión

Brindar servicios de salud de alta complejidad con un enfoque en la seguridad, calidad y atención centrada en las personas. A través de la implementación de diagnósticos, tratamientos y cuidados médicos avanzados, buscamos mejorar la salud y bienestar de nuestros pacientes, aplicando estrictos protocolos de seguridad y gestión del riesgo para garantizar su integridad en cada etapa de su atención.

1.1.3 Visión

Consolidarse como un referente nacional e internacional en la prestación de servicios de salud de alta complejidad, siendo reconocida por su excelencia médica, su enfoque humanizado en la atención al paciente y su liderazgo en innovación, seguridad y calidad. Aspiramos a convertirnos en una institución que transforme vidas a través de soluciones médicas avanzadas y un compromiso inquebrantable con la mejora continua y la prevención de riesgos.

1.1.4 Giro de la empresa

La prestación de servicios de salud de alta complejidad, con un enfoque especializado en diagnósticos, tratamientos y cuidados médicos avanzados y de igual manera gestionando de manera proactiva el riesgo clínico para garantizar el bienestar de los pacientes.

1.1.5 Dirección de la empresa

C. Rosa 1, El Tesoro, 54957 Buenavista, Mex., México.

Tultitlán de Mariano Escobedo, Estado de México, 54957

1.1.6 Organigrama de la Empresa



1.1.7 DESCRIPCIÓN DE PROCESOS DE LA EMPRESA

1.1.8 Área donde se desarrolló el proyecto

El Proyecto de la Clinica “Sagrado Corazon” , es desarrollado por alumnos de la carrera de Ingeniería en Sistemas Computacionales del Instituto Tecnológico de Oaxaca, mayormente el proyecto fue desarrollado en las instalaciones del laboratorio de centro de cómputo del instituto.

1.2 GENERALIDADES DEL PROYECTO

1.2.1 Nombre del Proyecto

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE REGISTRO Y CONTROL PARA LA CLINICA SAGRADO CORAZON

1.2.2 Planteamiento del Problema

En el contexto actual de la salud, es crucial optimizar la gestión de información y procesos para asegurar una atención médica de calidad y centrada en el paciente. Muchas clínicas aún enfrentan problemas por la falta de sistemas integrales para el registro de pacientes, el control de procesos y la gestión de citas. La ausencia de estas herramientas puede generar errores, ineficiencias y costos adicionales, afectando tanto la calidad de la atención como la experiencia del paciente.

Para resolver esto, se propone implementar un sistema integral que automatice el registro de pacientes, facilite el control de los procesos internos y optimice el agendamiento de citas. Esto mejorará la eficiencia operativa, permitirá una atención médica más centrada en el paciente y aumentará la satisfacción tanto del personal como de los pacientes.

Además, un sistema integral permitirá una mejor coordinación entre las distintas áreas de la clínica, asegurando que la información se comparta de manera fluida y sin retrasos. Esto contribuirá a una toma de decisiones más ágil y basada en datos precisos, mejorando la calidad del diagnóstico y tratamiento. También reducirá la carga administrativa, permitiendo al personal concentrarse más en la atención al paciente. En última instancia, este enfoque integral fortalecerá la confianza de los pacientes en la clínica, al ofrecer un servicio más eficiente y personalizado.

1.2.3 OBJETIVOS

1.2.3.1 Objetivo General

Desarrollar un programa con el propósito de tener una buena gestión de información y facilite el control de procesos internos para el control de pacientes

1.2.3.2 Objetivos Específicos

- Implementar un sistema de gestión de citas: Permitir a los pacientes programar citas médicas en línea y gestionar eficientemente la agenda de los médicos y especialistas.
- Desarrollar un sistema de registro electrónico de pacientes: Crear un sistema centralizado para registrar y mantener actualizada la información personal, médica y de seguros de cada paciente.
- Integrar herramientas de seguimiento de tratamientos: Implementar funcionalidades para registrar y monitorear el progreso de los tratamientos médicos y terapias de los pacientes.
- Establecer un sistema de facturación y cobro: Automatizar el proceso de facturación y cobro, integrando los registros médicos con los sistemas de facturación para una gestión más eficiente.

1.2.4 Justificación

Teniendo en cuenta las principales necesidades y prioridades de la clínica Sagrado Corazón hemos querido ofrecer un sistema de registro, control y información para esta clínica, hemos tomado en cuenta sistemas parecidos en el sector de salud y hemos visto los puntos fuertes y las deficiencias que notamos en los otros sistemas para así ofrecer una mejor versión.

La Clínica Sagrado Corazón es una Institución que brinda servicios de salud que ofrece a la comunidad una atención eficiente. Su objetivo principal es realizar acciones encaminadas a satisfacer las necesidades de salud del paciente, su familia y la comunidad, planeando,

organizando, ejecutando y controlando las actividades asistenciales y generales de los servicios que ofrece.

También busca ejercer un control para cada una de sus actividades de tal forma que los resultados obtenidos favorezcan tanto a la clínica como a la comunidad, este proceso ha puesto en evidencia fallas que tiene consecuencias negativas a los propietarios, doctores y todo el plantel de la clínica como en la atención a los usuarios.

La importancia que tiene el desarrollo de diseñar un sistema de control interno administrativo en la clínica Sagrado Corazón, que continuamente evalúa los procesos, detecte fallas y recomiende acciones a implementar y las diferentes etapas de ejecución en forma programada, con el fin de garantizar un manejo oportuno y objetivos de los resultados que quiere conseguir la clínica.

Este sistema de control interno administrativo le servirá a la administración de la Clínica Sagrado Corazón, para ejercer de manera eficaz la función de control logrando así mejorar los procesos y la forma en que se lleva a cabo y el cumplimiento de los objetivos como Clínica prestadora de servicios de salud.

1.2.5 ALCANCES Y LIMITACIONES

1.2.5.1 Alcances

- Integración con dispositivos médicos: Permitir la integración del sistema con dispositivos médicos para la captura automática de datos, como resultados de pruebas de laboratorio, signos vitales o imágenes médicas.
- Acceso remoto: Facilitar el acceso remoto al sistema para que el personal médico pueda revisar y actualizar la información de los pacientes desde cualquier lugar, lo que podría mejorar la atención o consultas fuera del consultorio.
- Seguimiento de inventario: Incluir funciones para el seguimiento y gestión del inventario de medicamentos, lo que ayudaría a optimizar la logística y evitar que falten o se exceda el inventario.
- Gestión de reseñas: Incluir un apartado para la gestión de reseñas, como el registro de horas laborales, programación de turnos, evaluación del desempeño del personal.

1.2.5.2 Limitaciones

- Aceptación del proyecto: Demora en la revisión y análisis de la propuesta por parte de la Clínica, podría tomar semanas o meses.
- Demora de la información: Se puede demorar en que se entregue la información de los Médicos, Pacientes, Medicinas, Citas, etc.
- Costos y mantenimiento: La implementación y el mantenimiento del Sistema requerirán una inversión y la formación del personal, así como gastos continuos de mantenimiento y actualización.

1.2.6 Cronograma de Actividades

Semana	Fase	Actividad
1-2	Inicio del Proyecto	Reunión con la clínica para definir funcionalidades clave y obtener requisitos detallados del sistema.
3-4	Planificación	Diseño de la arquitectura base del sistema y estructura de la base de datos.
5-8	Desarrollo Iteración 1	Configuración del entorno de desarrollo y creación de la base de datos. Implementación de primeros modelos y estructuras.
9-12	Desarrollo Iteración 2	Implementación de módulos iniciales de registro de pacientes y empleados. Pruebas unitarias de funcionalidad básica.
13-16	Desarrollo Iteración 3	Desarrollo del módulo de gestión de citas médicas, incluyendo calendario de citas y gestión de turnos. Pruebas unitarias.
17-18	Pruebas y Ajustes	Integración de módulos desarrollados. Pruebas de integración y revisión del código.
18-21	Implementación y Despliegue	Implementación del sistema de facturación y pagos.
1-2	Inicio del Proyecto	Desarrollo del módulo de generación de reportes médicos.
3-4	Planificación	Revisión final del sistema para asegurar que cumple con todos los requisitos. Instalación y configuración de hardware y software necesarios. Realización de sesiones de formación para el personal del restaurante. Creación de manuales y guías de usuario.

CAPÍTULO 2 MARCO METODOLÓGICO

Actividades realizadas:

- Análisis de la viabilidad técnica, económica y operativa del proyecto.
- Se describen las herramientas que se ocuparon.

Fase de Exploración:

- Recopilación de requisitos y expectativas acorde del restaurante.
- Creación de historias de usuario iniciales con sus respectivos BPMN.

Fase de planeación:

- Prioridad de las historias de usuarios.
- Plan de entrega.
- Lista Maestra
- Tabla de Planificación Iteraciones
- Tareas por iteración
- Velocidad del proyectos

Fase de iteraciones

Iteración:

Planificación

Diseño

Desarrollo

Pruebas Casos de prueba

2.1 Metodología de Desarrollo

El modelo de desarrollo utilizado para el Proyecto de la “Clínica Sagrado Corazón” es el modelo de desarrollo ágil Extreme Programming (XP).

El modelo de desarrollo ágil Extreme Programming (XP) es una metodología que se centra en la entrega rápida y continua de software de alta calidad. Se caracteriza por su enfoque en la comunicación cercana entre los miembros del equipo de desarrollo y los clientes, así como en la adaptación continua a los cambios y la mejora constante del producto.

En el contexto de este proyecto, la utilización de XP permitirá un desarrollo ágil, adaptable y orientado a la entrega de valor de manera continua, asegurando así la satisfacción de los usuarios y la eficiencia en la gestión del proyecto.

2.2 Estudio de Factibilidad

Lenguaje: Para la generación del programa se necesita de Java.

Base de datos: Selección de tecnologías como MySQL que soporte un volumen grande de datos y aseguren alta disponibilidad.

Ficha técnica del equipo:

CPU: Intel core i5 de 13va generación

RAM: 8 GB

Almacenamiento: 512 GB en almacenamiento de estado sólido.

Actualizaciones y mantenimiento:

Establecer si se tendrá un sistema de actualización continua o un esquema basado en versiones. Esto implica tener una metodología de desarrollo ágil (Scrum) para adaptarse a cambios en las normativas o necesidades de los usuarios.

Equipo	Detalles
--------	----------

PC	<ul style="list-style-type: none"> - Procesador con Core i5 o i7 de 13va generación. - RAM con 16GB - Almacenamiento 512 GB
Impresora	<ul style="list-style-type: none"> - Impresión a laser - velocidad de impresión de 30-40 ppm - Conectividad USB, WIFI, Ethernet e impresión móvil.
Wifi	<ul style="list-style-type: none"> - Velocidad de hasta 500Mbps de velocidad de descarga.

2.2.1 Estudio Operacional

Objetivos Operacionales

Proporcionar una plataforma confiable y eficiente para que los pacientes agenduen citas médicas.

Reducir el número de citas perdidas o canceladas.

Mejorar la gestión de tiempo y recursos de las clínicas y hospitales.

Ofrecer soporte continuo para garantizar la disponibilidad del sistema.

Gestión de Recursos

Servidor y Almacenamiento en la Nube

Seguridad y Protección de Datos

Infraestructura de Red

Recursos Humanos

Costos Operacionales

Aspecto operacional	Descripción
Facilidad de uso	<ul style="list-style-type: none"> - Diseñar una interfaz de usuario (UI) intuitiva y accesible para los distintos tipos de usuarios. - Se debe considerar una curva de aprendizaje baja para minimizar el tiempo de capacitación
Capacitación del personal	<ul style="list-style-type: none"> - Estimar si será necesario capacitar al personal médico y administrativo en el uso del sistema. - Incluir módulos de autoaprendizaje o tutoriales dentro del software para facilitar la adopción.
Satisfacción de necesidades	<ul style="list-style-type: none"> - Administración eficiente de registros. - Control de procesos clínicos. - Gestión de citas. - Reportes.
Escalabilidad	<ul style="list-style-type: none"> - Evaluar la capacidad del sistema para crecer según aumente el número de usuarios y de registros médicos.

2.2.2 Estudio Económico

3 Desarrolladores	\$4000 al mes c/u
Herramientas y Licencias	\$5,000
Infraestructura tecnológica	\$1,000
Mantenimiento del Software	\$10,000 anuales
Servidores en la nube	\$2,000 anuales
Dominio y hosting web	\$500 anuales

Viabilidad: El programa es rentable desde el primer año con una base de 50 clientes, y la proyección de ingresos en el segundo año muestra un crecimiento significativo.

Retorno de Inversión (ROI): El ROI es alto, especialmente a partir del segundo año, cuando los costos iniciales de desarrollo ya no son relevantes.

Escalabilidad: El programa puede escalar fácilmente para atender más clínicas sin un aumento considerable en los costos operativos.

Análisis de Rentabilidad

Beneficio Bruto Año 1

$292,000(\text{ingresos}) - 91,500(\text{costos iniciales}) - 48,000(\text{costos operativos anuales}) = \$152,500$.

Proyección Año 2

Se espera un crecimiento del 50% en la base de clientes

Nuevas clínicas suscritas: 75 (45 anuales, 30 mensuales).

Ingresos estimados: $292,000 \times 1.5 = \$438,000$

Beneficio bruto: $438,000 - 48,000 (\text{costos operativos}) = \$390,000$

2.3 Herramientas Empleadas

1. Entorno de Desarrollo Integrado (IDE): Usamos NetBeans para la codificación en Java y la gestión del proyecto.
2. Modelado de Procesos de Negocio: Utilizamos Bizagi para crear diagramas de flujo y analizar procesos.
3. Gestión de Base de Datos: Empleamos MySQL para almacenar y gestionar los datos del proyecto.
4. Diagramas y Visualización: Lucidchart nos ayudó a crear diagramas y visualizaciones, como diagramas de base de datos y de flujo.
5. Microsoft Office: Usamos Microsoft Office para crear documentos, hojas de cálculo y presentaciones, parte de la documentación del proyecto.

Estas herramientas facilitaron el desarrollo y la gestión del proyecto de manera eficiente.

2.4 Fase de Exploración

Para identificar los requisitos del sistema de la Clínica Sagrado Corazón, se realizaron encuestas digitales y entrevistas con el personal médico y administrativo. Esto permitió recoger información clave sobre los desafíos en la gestión de citas, inventarios y seguimiento de tratamientos, destacando áreas de mejora y riesgos en los procesos actuales.

Además, se observaron las actividades diarias en la clínica, lo cual brindó una visión completa de los flujos de trabajo y las necesidades operativas. La información obtenida en esta fase será fundamental para definir un sistema adaptado a los problemas reales que enfrenta el personal y optimizar los procesos en beneficio de la atención al paciente.

Gracias a esta encuesta, se logró obtener información valiosa directamente de los involucrados en las operaciones del restaurante. Esta información será crucial para identificar

los requisitos específicos del sistema que se desarrollará y garantizar así que responda a los problemas reales que enfrenta el equipo.

Para complementar la información se utilizaron métodos de observación de las actividades y los procesos que se llevan a cabo para el desarrollo de requisiciones dentro de la empresa.

2.4.1 ENCUESTA

SISTEMA DE LA CLÍNICA SAGRADO CORAZÓN

Este formulario está diseñado para recoger información sobre las preferencias y expectativas de los usuarios, con el fin de optimizar el programa generador de citas médicas, mejorar la experiencia de usuario y adaptar el sistema a las necesidades de los clientes.

gersonvalentin15@gmail.com [Cambiar cuenta](#)



No compartido

¿Cómo prefiere acceder al sistema de citas médicas?

- Aplicación móvil
- Sitio web
- Llamada telefónica automatizada
- A través de un asistente virtual/chatbot

¿Qué tan importante es para usted recibir recordatorios antes de la cita?

- Muy importante
- Importante
- Poco importante
- No es importante

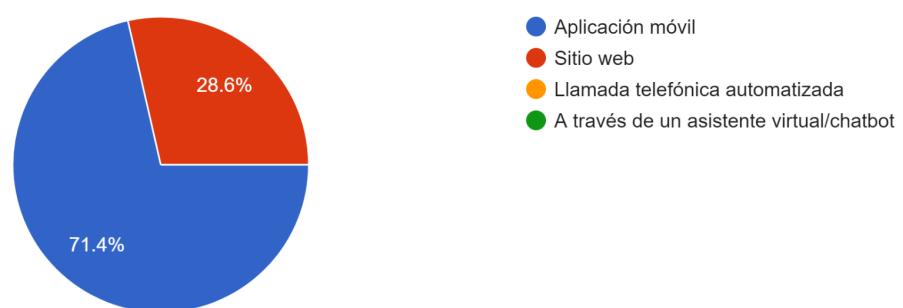
¿Cuál es el método de pago que usted prefiere?

- Tarjeta
- Efectivo
- Transferencia

2.4.2 Información de observación (Encuesta)

¿Cómo prefiere acceder al sistema de citas médicas?

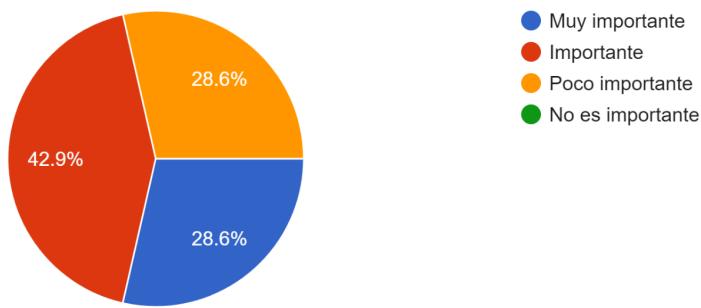
7 respuestas



En la primera pregunta de la encuesta es notable que la mayoría de usuarios prefieren que el sistema de citas médicas sea por aplicación móvil quizá porque prefieren tener el control en sus manos de manera rápida y sencilla con el simple hecho de dar un clic.

¿Qué tan importante es para usted recibir recordatorios antes de la cita?

7 respuestas



Con un 42.9% de votación de los usuarios los cuales piensan que es importante ser notificados antes de una cita podemos tener claro que las notificaciones son un punto a tener en cuenta para la comodidad de los usuarios.

¿Qué aspectos del programa de citas médicas podrían mejorarse?

3 respuestas

Un mejor diseño en la interfaz

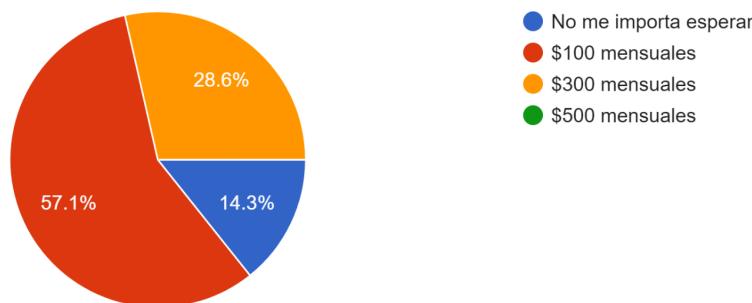
Hacerla más rápida

Hacerla más intuitiva

Para este inciso pocos usuarios respondieron lo cual hace pensar que es satisfactoria su experiencia con el programa, aún así podemos pulir algunos detalles como los de una interfaz más llamativa e intuitiva como lo recomienda el primer usuario así como una optimización para una respuesta inmediata como lo sugiere el segundo y por último hacer aún más intuitiva la interfaz tomando como ayuda iconos u otros detalles que nos ayuden a lograr este objetivo.

¿Cuánto estaría dispuesto a pagar por el uso de un programa que optimice la gestión de citas médicas y facilite el acceso a consultas?

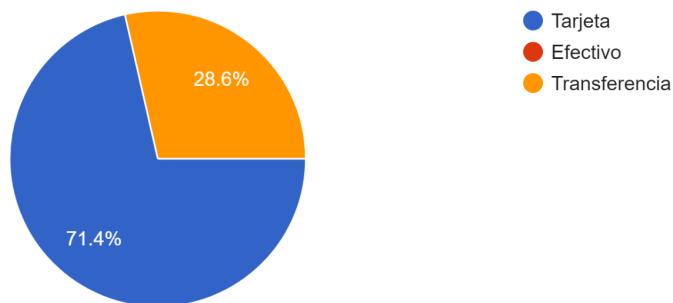
7 respuestas



Al parecer la optimización del programa no es tan importante para los usuarios como se esperaba pero eso no quiere decir que no lo sea pues la mayoría con un 57.1% está dispuesta a pagar \$100 pesos y otro 28.6% dispuestos a pagar \$300 podemos hacer un promedio de \$200 pesos para la optimización.

¿Cuál es el método de pago que usted prefiere?

7 respuestas



Con un alto porcentaje del 71.4% podemos estar seguros que el método de pago favorito de los usuarios es por tarjeta esto quizá por su fácil uso y seguridad para los usuarios aun así también se tendría en cuenta la otra parte que prefiere por transferencia.

Para el sistema de clínica sencillo sin facturación, que incluye roles de administrador, doctor, caja y secretaría, la información de la encuesta se estructuraría de la siguiente forma:

1. Roles y Funciones en el Sistema:

- **Administrador:** Responsable de gestionar el acceso de usuarios, la configuración del sistema y supervisar el uso de los módulos del sistema.
- **Doctor:** Accede a la información de los pacientes y el historial médico. Puede registrar diagnósticos, tratamientos y planificar futuras consultas.
- **Caja:** Responsable de recibir los pagos de los pacientes. El sistema permite registrar pagos, emitir recibos internos y ver el historial de pagos de cada paciente.
- **Secretaria:** Administra las citas, registrando nuevas y gestionando cambios en el horario. También puede acceder a la información básica de los pacientes para coordinar las citas y comunicar recordatorios.

2. Gestión de Citas:

- La creación y modificación de citas es posible en cualquier momento de la semana, sin restricciones de horarios.
- **Doctores** pueden visualizar su agenda de citas y marcar como consultas atendidas .
- **Secretaría** es quien ingresa y organiza las citas, según la disponibilidad del doctor y
- las necesidades del paciente.

3. Manejo de Pacientes:

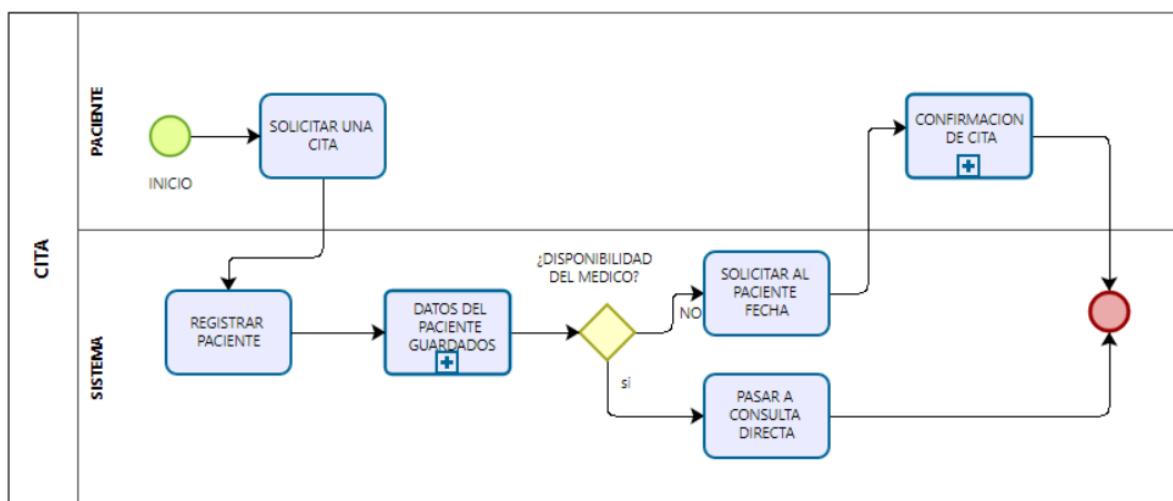
- Los registros de pacientes incluyen datos personales, historial médico y antecedentes de tratamientos previos.
- El **doctor** puede actualizar el historial del paciente y añadir nuevos diagnósticos o prescripciones.
- La **secretaria** tiene acceso limitado solo a datos necesarios para la gestión de citas, sin acceso a información médica.

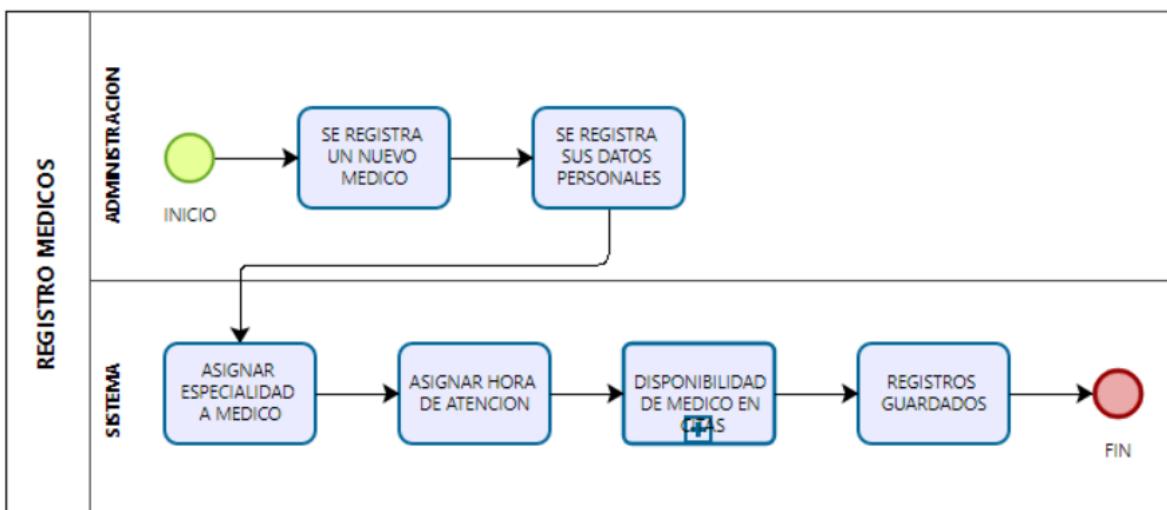
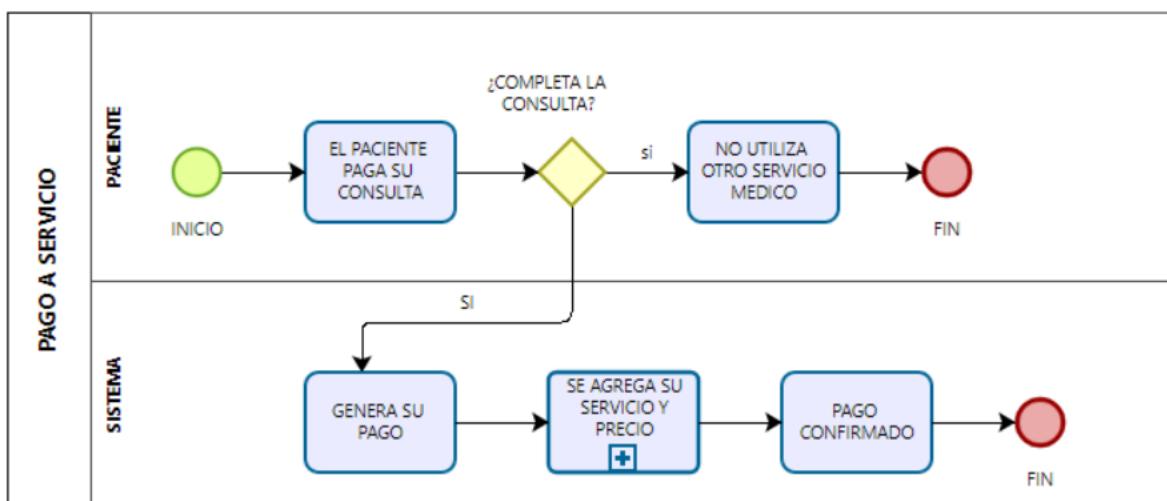
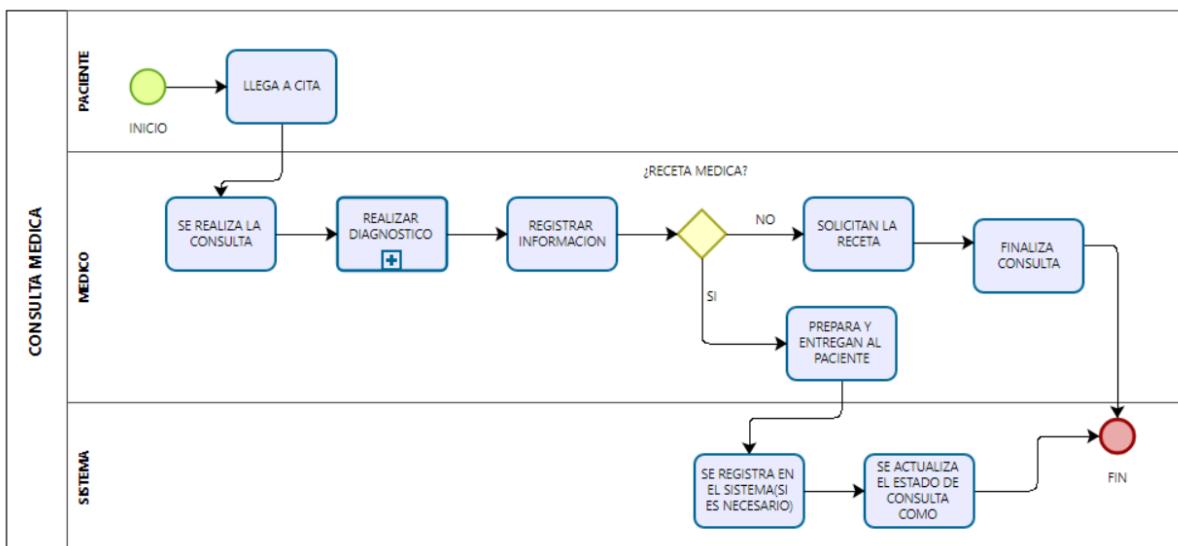
4. Pagos en caja (solo efectivo):

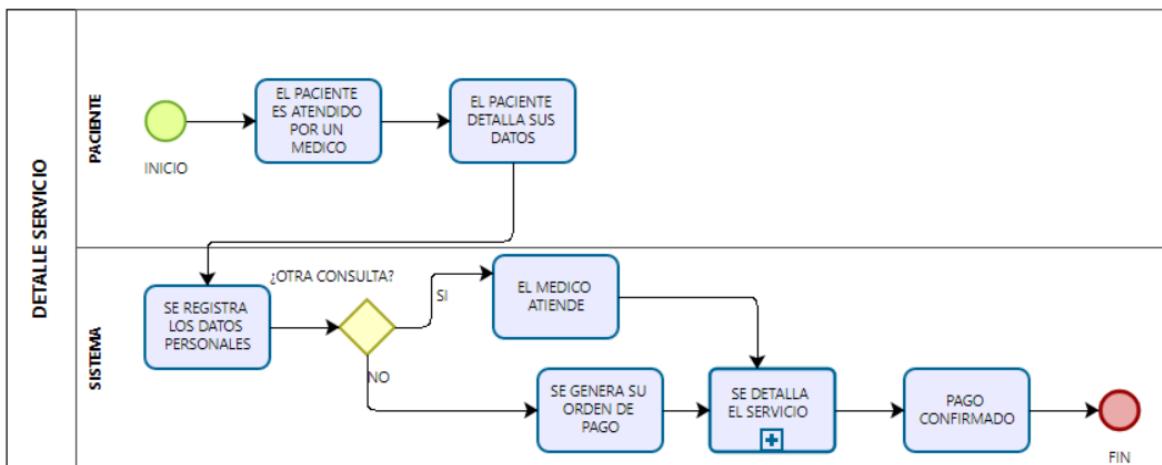
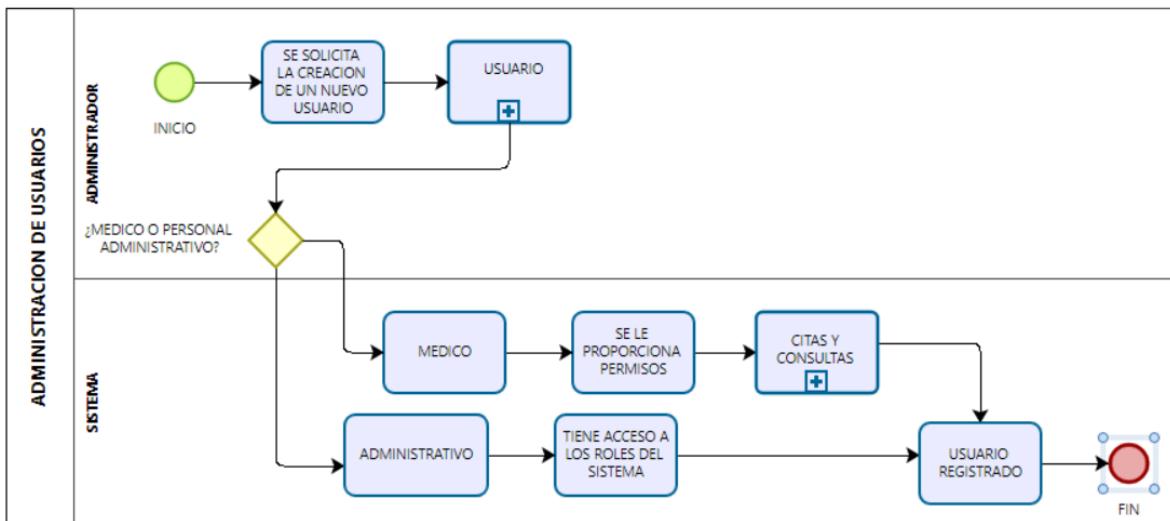
Al finalizar la consulta o procedimiento, el paciente se dirige a caja para realizar el pago en efectivo. El sistema registra el pago y emite un recibo simple, confirmando la transacción en efectivo sin manejo de otros medios de pago.

Procedimientos de observación utilizados: Se aplicaron entrevistas con cada rol para entender sus procesos diarios, observación directa para identificar el flujo de pacientes y entrevistas contextuales para asegurar que cada usuario entiende sus funciones dentro del sistema.

2.4.3 Diagramas BPMN







2.5 Fase de Planeación

Definir Historias de Usuario

Número	Nombre de la Historia	Prioridad
1	Registro y actualización de pacientes	Alta
2	Gestión y programación de citas	Alta
3	Seguimiento y control de tratamientos	Media

4	Gestión de inventario de medicamentos	Alta
5	Generación automática de facturas	Media
6	Acceso remoto para personal médico	Baja
7	Evaluación del desempeño del personal	Baja
8	Generación de reportes médicos	Media

Planificación 2da etapa

PLAN DE ENTREGAS

Iteración	Historias de Usuario	Duración de la Iteración
1ra Iteración	1. Solicitar una cita 2. Registrar información del paciente	3 semanas
2da Iteración	1. Verificar disponibilidad del médico 2. Realizar la consulta médica	4 semanas
3ra Iteración	1. Registrar diagnóstico y receta médica 2. Confirmar y realizar el pago de la consulta	3 semanas

Lista Maestra de Historias de Usuario

Historia de Usuario	Días Estimados	Prioridad
Solicitar una cita	8 días	Alta

Consulta Médica	10 días	Alta
Pago a servicio	7 días	Media
Detalle Servicio	8 días	Alta
Registro Medicos	6 días	Media
Administracion de usuarios	5 días	Baja

PLANIFICACIÓN 3ER PASO (ITERACIONES)

1ER ITERACIÓN

(Cita y Consulta Médica)

Historia de Usuario	Análisis	Puntos Estimados
Cita por pacientes	Crear una interfaz para solicitar cita y verificar la disponibilidad inicial del paciente.	3
	Almacenar los datos del paciente en la base de datos de manera segura.	4
	Notificar al paciente sobre la confirmación de su cita	2
Consulta Médica	Implementar lógica para verificar la disponibilidad de médicos y asignar horarios de consulta.	3
	Crear un módulo para que el médico pueda registrar el diagnóstico y recomendaciones del paciente.	5
	Registrar en el sistema el diagnóstico y receta médica generada por el doctor.	4

2DA ITERACIÓN

(Pago a Servicio y Detalle de Servicio)

Historia de Usuario	Análisis	Puntos Estimados
Confirmar y realizar el pago	Crear una interfaz de pago seguro para confirmar y procesar el pago de la consulta.	5
	Registrar los detalles de pago y emitir un recibo al paciente.	3
	Cambiar el estado de la consulta a "Completada" después del pago.	2
Detallar costos de servicio	Generar un resumen detallado de costos adicionales si se solicitan más servicios.	3
	Permitir al paciente agregar y pagar por servicios adicionales si es necesario.	4
	Notificar al paciente que la consulta ha sido completada y que no quedan servicios pendientes.	2

3RA ITERACIÓN

(Registro de Médicos y Administración de Usuarios)

Historia de Usuario	Análisis	Puntos Estimados
Registrar nuevo médico	Crear la funcionalidad para que un administrador registre los datos personales de un médico.	4
	Permitir al sistema clasificar al médico según su especialidad médica registrada.	3
	Configurar la disponibilidad del médico para futuras citas en el sistema	2
Crear usuario administrativo	Permitir la creación de usuarios con roles administrativos para la gestión de la clínica..	4
	Configurar los permisos de acceso en función del rol del usuario dentro del sistema.	5
	Garantizar que el usuario tenga acceso a los módulos de citas y consultas según su rol.	3

Tareas Establecidas por Cada Iteración

N	Historia	Tarea	Tipo	Puntos Estimados	Semana	Fecha	Descripción
1	1	Solicitar una cita	Diseño	4	1-3	28 de octubre - 29 de octubre	Crear el diseño de la interfaz para que el paciente solicite una cita médica.
2	1	Implementar solicitud de cita	Desarrollo	3	1-3	30 de octubre - 31 de octubre	Desarrollo de la funcionalidad para que el paciente solicite una cita en el sistema.
3	1	Validar datos del paciente	Diseño	3	1-3	1 de noviembre - 2 de noviembre	Verificación de los datos del paciente antes de confirmar la cita.
4	2	Registrar información del paciente	Desarrollo	4	1-3	3 de noviembre - 4 de noviembre	Implementación del sistema para guardar de forma segura los datos del paciente.
5	2	Crear interfaz de registro de datos	Diseño	3	4-7	5 de noviembre - 6 de noviembre	Diseño de una interfaz intuitiva para el registro de la información del paciente.
6	2	Implementar lógica de registro de datos	Diseño	2	4-7	7 de noviembre - 8 de	Desarrollo de la lógica para registrar y almacenar la

						noviemb re	información del paciente en el sistema.
7	3	Diseñar la interfaz de verificación de disponibilidad	Diseño	2	7-9	9 de noviemb re - 10 de noviemb re	Crear el diseño de la página para verificar la disponibilidad de médicos, permitiendo la selección de horarios disponibles.
8	3	Implementar la lógica de verificación de disponibilidad	Desarrollo	3	7-9	11 de noviemb re - 12 de noviemb re	Desarrollar la funcionalidad para comprobar la disponibilidad de los médicos en tiempo real.
9	3	Diseñar la interfaz para confirmación de cita	Diseño	2	7-9	13 de noviemb re - 14 de noviemb re	Crear una interfaz amigable y eficiente para la confirmación de citas por parte del paciente.
10	4	Implementar la consulta médica	Diseño	4	10-12	15 de noviemb re - 16 de noviemb re	Desarrollar el módulo de consulta médica para permitir que el médico registre el diagnóstico y recomendaciones del paciente.
11	4	Diseñar la interfaz de consulta médica	Diseño	3	10-12	17 de noviemb re - 18 de noviemb re	Crear una interfaz intuitiva que permita al médico registrar el diagnóstico y seguimiento del paciente.
12	5	Diseñar la interfaz para registrar diagnóstico	Diseño	2	13-15	19 de noviemb re - 20 de noviemb re	Crear el diseño de la interfaz para el registro del diagnóstico y prescripción de

							tratamientos para el paciente.
13	5	Implementar la lógica de registro de diagnóstico	Desarrollo	3	13-15	21 de noviembre - 22 de noviembre	Desarrollar la lógica para registrar y almacenar el diagnóstico y recetas en el expediente del paciente.
14	6	Diseñar la interfaz de pago de consulta	Diseño	2		23 de noviembre - 24 de noviembre	Crear el diseño de la interfaz para realizar el pago de consulta, asegurando que sea seguro y fácil de usar para el paciente.
15	6	Implementar la lógica de pago de consulta	Desarrollo	3		25 de noviembre - 26 de noviembre	Desarrollar la funcionalidad para procesar y confirmar el pago de la consulta médica, actualizando el estado de la cita como completada.

2.5.1 HISTORIAS DE USUARIO



HISTORIA DE USUARIO	
Número: 1	Usuario: Paciente
Nombre de Historia: Solicitar una cita	
Prioridad: Alta	
Riesgo de desarrollo: Bajo	
Puntos asignados: 3	
Iteración asignada: 1	



HISTORIA DE USUARIO	
Numero: 3	Usuario: Sistema
Nombre de Historia: Verificar disponibilidad del médico	
Prioridad: Alta	Riesgo de desarrollo: Bajo
Puntos asignados: 3	Iteración asignada: 2
Programador responsable: Yeudiel	
Descripción: El sistema necesita verificar la disponibilidad del médico en tiempo real para asignar horarios de consulta sin conflictos. Esto permitirá a los pacientes escoger horarios que se adapten a sus necesidades.	
Observaciones: Implementar lógica para actualizar la disponibilidad en caso de cambios o cancelaciones de citas.	

HISTORIA DE USUARIO	
Numero: 4	Usuario: Medico
Nombre de Historia: Realizar la consulta médica y registrar diagnóstico	
Prioridad: Alta	Riesgo de desarrollo: Medio
Puntos asignados: 5	Iteración asignada: 2
Programador responsable: Bolívar	
Descripción: El médico necesita una interfaz para registrar el diagnóstico y tratamiento del paciente durante la consulta. Esta información se guarda en el expediente del paciente y debe ser accesible en futuras consultas.	
Observaciones: Facilitar la búsqueda de historial médico del paciente dentro de la interfaz.	

HISTORIA DE USUARIO	
Numero: 5	Usuario: Paciente
Nombre de Historia: Confirmar y procesar el pago de la consulta	
Prioridad: Alta	Riesgo de desarrollo: Bajo
Puntos asignados: 3	Iteración asignada: 3
Programador responsable: Bolívar	
Descripción: El paciente debe confirmar y realizar el pago de la consulta de manera segura a través de la plataforma. Una vez completado el pago, el sistema debe actualizar el estado de la cita como "pagada" y enviar una confirmación al paciente.	
Observaciones: Asegurar métodos de pago seguros y notificaciones de confirmación pospago.	

HISTORIA DE USUARIO	
Numero: 6	Usuario: Administrador
Nombre de Historia: Crear y asignar roles de usuario en el sistema	
Prioridad: Media	Riesgo de desarrollo: Medio
Puntos asignados: 4	Iteración asignada: 3
Programador responsable: Yeudiel	
Descripción: El administrador necesita la capacidad de crear nuevos usuarios y asignarles roles específicos dentro del sistema, tales como médico, paciente o administrativo, con los permisos correspondientes.	
Observaciones: Asegurar que los permisos estén claramente definidos para evitar accesos no autorizados.	

2.5.2 PLAN DE ENTREGAS

Fase del proyecto	Objetivo Principal	Fecha de Inicio	Fecha de Fin Estimada	Estado	Comentarios
Análisis y Requisitos	Definir necesidades de usuarios	28/10/2024	01/10/2024	Terminado	Requiere revisión constante
Diseño de Sistema	Diseño de interfaz y base de datos	02/11/2024	05/11/2024	En proceso	Incluir diagramas ER y flujo
Desarrollo de Backend	Funcionalidad de citas y usuarios	06/11/2024	10/11/2024	En proceso	Uso de framework X
Desarrollo de Frontend	Interfaz de usuario para doctores y admins	11/11/2024	15/11/2024	En proceso	Basado en diseño de sistema
Pruebas y Corrección	Test de funcionalidad y	16/11/2024	18/11/2024	En proceso	Usar metodologías

	usabilidad				ágiles
Entrega Final y Capacitación	Implementación y capacitación	19/11/2024	20/11/2024	No iniciado	Feedback de usuarios

2.5.3 VELOCIDAD DE PROYECTO

Iteración	Duración (en días)	Tareas planeadas	Velocidad (puntos)
1	5	Análisis de requisitos y especificación	10
2	5	Diseño de sistema y base de datos	15
3	10	Desarrollo de backend (citas, usuarios)	25
4	5	Desarrollo de frontend	20
5	3	Pruebas y correcciones	15
6	2	Implementación y capacitación	10

2.6 FASE DE ITERACIONES

2.6.1 PLANIFICACIÓN

1RA ITERACIÓN

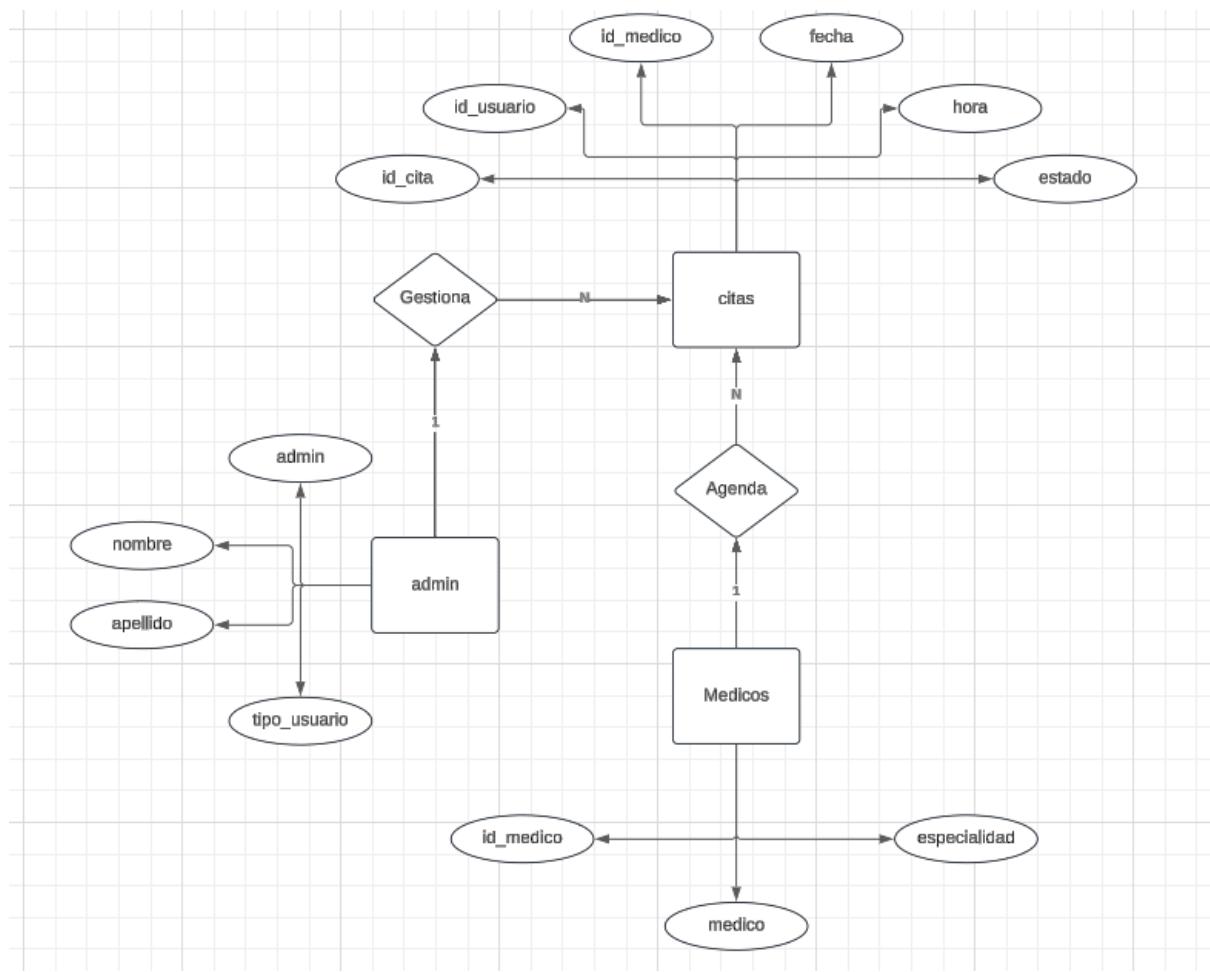
Historia de usuario 1

Historia de Usuario	Análisis	Puntos Estimados
Solicitar una cita	Crear una interfaz intuitiva para que el administrador médico seleccione médico, fecha y hora deseada para el registro de médico	3
	Confirmar la solicitud para evitar errores en la programación	2

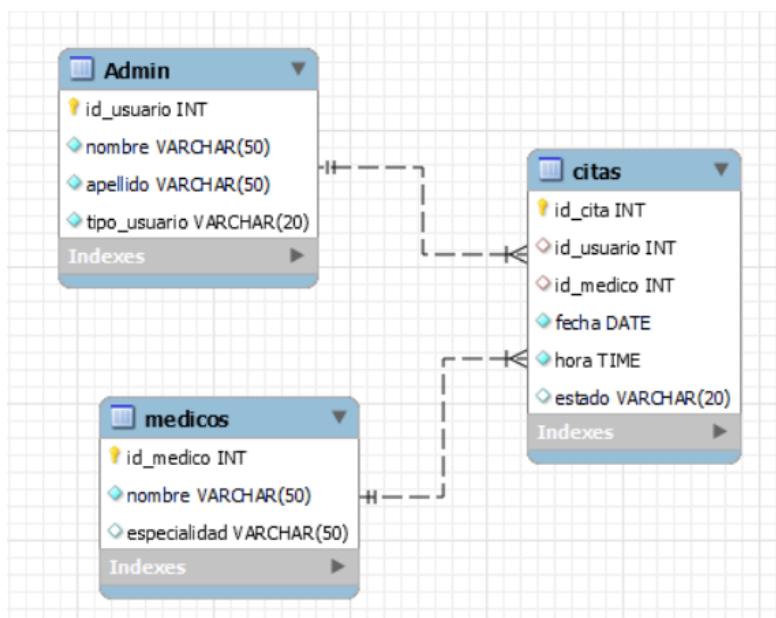
	Verificar que la interfaz sea accesible para mejorar la experiencia del administrador	1
--	---	---

HISTORIA DE USUARIO	
Número: 1	Usuario: Paciente
Nombre de Historia: Solicitar una cita	
Prioridad: Alta	Riesgo de desarrollo: Bajo
Puntos asignados: 3	Iteración asignada: 1
Programador responsable: Yeudiel	
Descripción: El paciente necesita solicitar una cita médica. El sistema debe ofrecer una interfaz intuitiva y rápida para que el paciente pueda escoger el médico, fecha, y horario deseado. Además, se debe confirmar la solicitud para evitar errores en la programación.	
Observaciones: La interfaz debe ser simple y accesible desde dispositivos móviles para mejorar la experiencia del usuario.	

MODELO ENTIDAD RELACIÓN



MODELO RELACIONAL

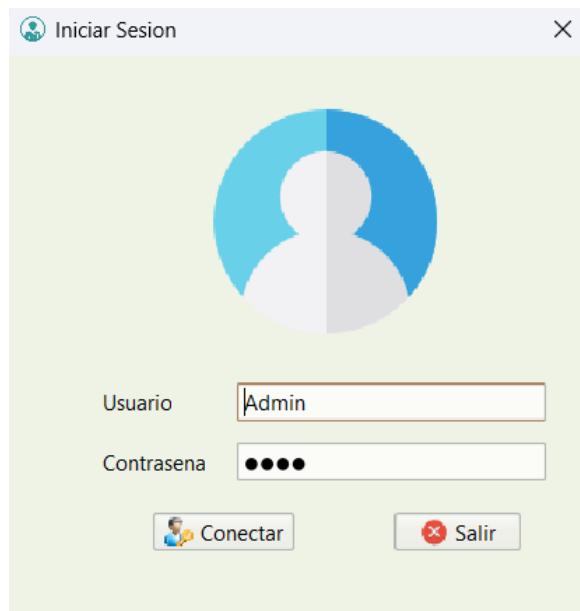


FASE DISEÑO TARJETAS CRC

CITA			
Responsabilidades		Colaboraciones	
Datos:	Acciones:	Clases:	Multiplicidad:
Información del paciente Selección del médico Fecha y hora de la cita Estado de confirmación de la cita	<ul style="list-style-type: none"> Crear una interfaz intuitiva para que el admin seleccione médico, fecha y hora deseada al momento de registrarla Confirmar la solicitud para evitar errores en la programación 	Sistema citas , Paciente , Médico	1..1

DISEÑO

Inicio Sesión admi



Agregando médico

A screenshot of a Windows-style application window titled "Agregar Medico". It contains three text input fields: "Nombres" with "Bolivar", "Apellidos" with "Garcia Osorio", and "Especialidad" with "Especialista". At the bottom are three buttons: "Nuevo" with a pencil icon, "Agregar Horario" with a calendar icon, and "Salir" with a red X icon.

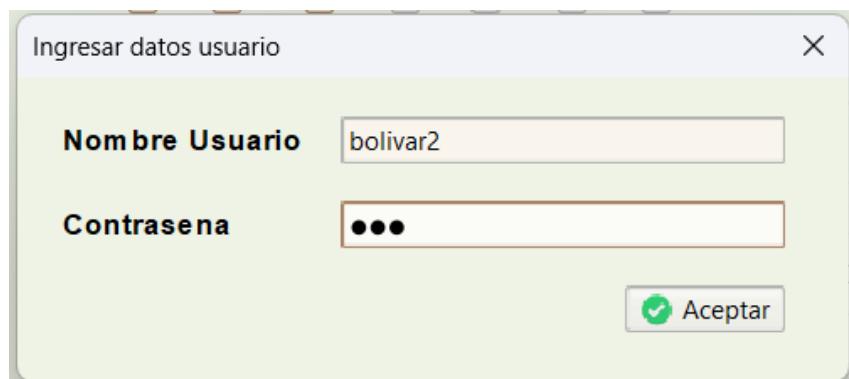
Selección de horario

Días disponibles

<input checked="" type="checkbox"/> L	<input checked="" type="checkbox"/> M	<input checked="" type="checkbox"/> X	<input type="checkbox"/> J	<input type="checkbox"/> V	<input type="checkbox"/> S	<input type="checkbox"/> D
---------------------------------------	---------------------------------------	---------------------------------------	----------------------------	----------------------------	----------------------------	----------------------------

Lunes	Desde	8:00 A.M	Hasta	5:00 P.M
Martes	Desde	10:00 A.M	Hasta	12:00 P.M
Miercoles	Desde	11:00 A.M	Hasta	3:00 P.M

Registro de médico con login para el acceso del sistema



DESARROLLO

Tenemos una clase conexión para que se pueda ejecutar correctamente el login del admin

```

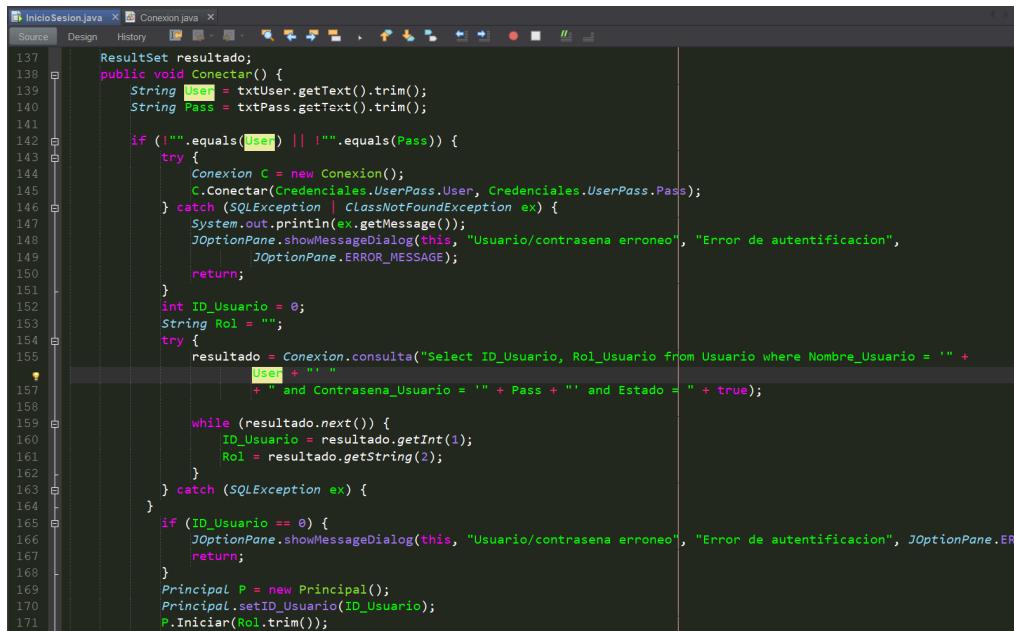
package Clases;
import java.sql.*;
import javax.swing.JOptionPane;
/*
 * @eudi
 */
public class Conexion {
    public static Connection con;
    public static Statement state, state1;
    public static ResultSet result, result1;

    public void Conectar(String user, String pass) throws SQLException, ClassNotFoundException {
        String url = "jdbc:mysql://localhost/hospital?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC";
        con = DriverManager.getConnection(url, "root", "12345");

        state = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);
        JOptionPane.showMessageDialog(null, "Conexión Establecida", "Conexión Establecida", JOptionPane.INFORMATION_MESSAGE);
    }
    public static ResultSet consulta(String sql) throws SQLException {
        // Crear un Statement para realizar la consulta
        state1 = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
        result1 = state1.executeQuery(sql);
        return result1;
    }
    public Object conexion() {
        throw new UnsupportedOperationException("Not supported yet.");
    }
}

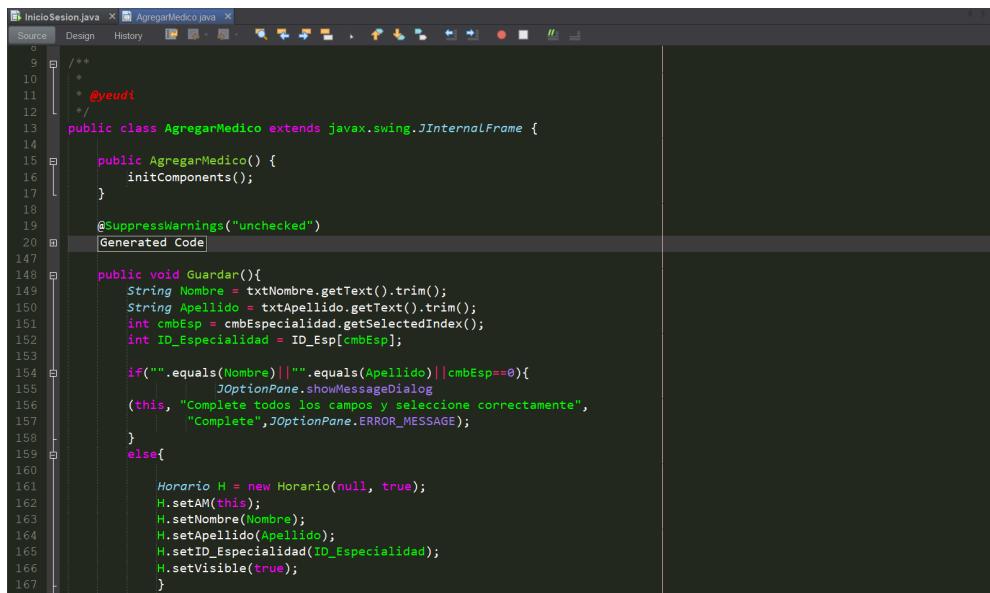
```

Esta es la clase de inicio sección para el administrador del sistema

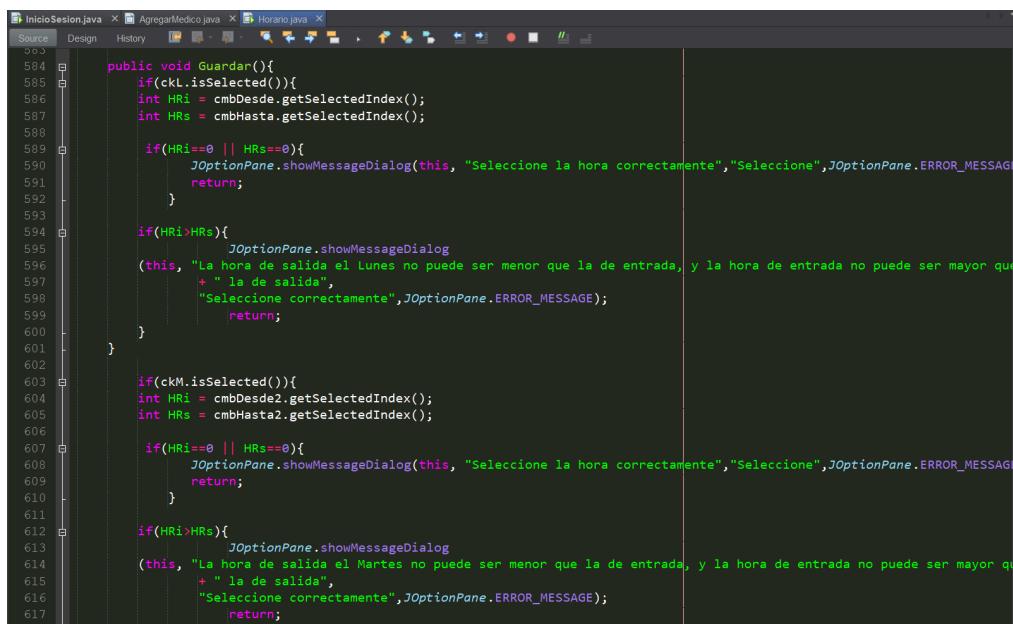


```
137     ResultSet resultado;
138     public void Conectar() {
139         String User = txtUser.getText().trim();
140         String Pass = txtPass.getText().trim();
141
142         if (!"".equals(User) || !"".equals(Pass)) {
143             try {
144                 Conexion C = new Conexion();
145                 C.Conectar(Credenciales.UserPass.User, Credenciales.UserPass.Pass);
146             } catch (SQLException | ClassNotFoundException ex) {
147                 System.out.println(ex.getMessage());
148                 JOptionPane.showMessageDialog(this, "Usuario/contraseña erroneo", "Error de autentificación", JOptionPane.ERROR_MESSAGE);
149             }
150             return;
151         }
152         int ID_Usuario = 0;
153         String Rol = "";
154         try {
155             resultado = Conexion.consulta("Select ID_Usuario, Rol_Usuario from Usuario where Nombre_Usuario = '" +
156                 User + "' " +
157                 + " and Contraseña_Usuario = '" + Pass + "' and Estado = " + true);
158
159             while (resultado.next()) {
160                 ID_Usuario = resultado.getInt(1);
161                 Rol = resultado.getString(2);
162             }
163         } catch (SQLException ex) {
164         }
165         if (ID_Usuario == 0) {
166             JOptionPane.showMessageDialog(this, "Usuario/contraseña erroneo", "Error de autentificación", JOptionPane.ERROR_MESSAGE);
167             return;
168         }
169         Principal P = new Principal();
170         Principal.setID_Usuario(ID_Usuario);
171         P.Iniciar(Rol.trim());
```

Posteriormente se muestra la asignación de registrar medio y horario



```
0 /**
1  *
2  * @yeudi
3 */
4 public class AgregarMedico extends javax.swing.JInternalFrame {
5
6     public AgregarMedico() {
7         initComponents();
8     }
9
10    @SuppressWarnings("unchecked")
11    Generated Code
12
13    public void Guardar(){
14        String Nombre = txtNombre.getText().trim();
15        String Apellido = txtApellido.getText().trim();
16        int cmbEsp = cmbEspecialidad.getSelectedIndex();
17        int ID_Especialidad = ID_Esp[cmbEsp];
18
19        if("".equals(Nombre)||"".equals(Apellido)||cmbEsp==0){
20            JOptionPane.showMessageDialog(
21                this, "Complete todos los campos y seleccione correctamente",
22                "Complete", JOptionPane.ERROR_MESSAGE);
23        }
24        else{
25
26            Horario H = new Horario(null, true);
27            H.setAM(this);
28            H.setNombre(Nombre);
29            H.setApellido(Apellido);
30            H.setID_Especialidad(ID_Especialidad);
31            H.setVisible(true);
32        }
33    }
34}
```

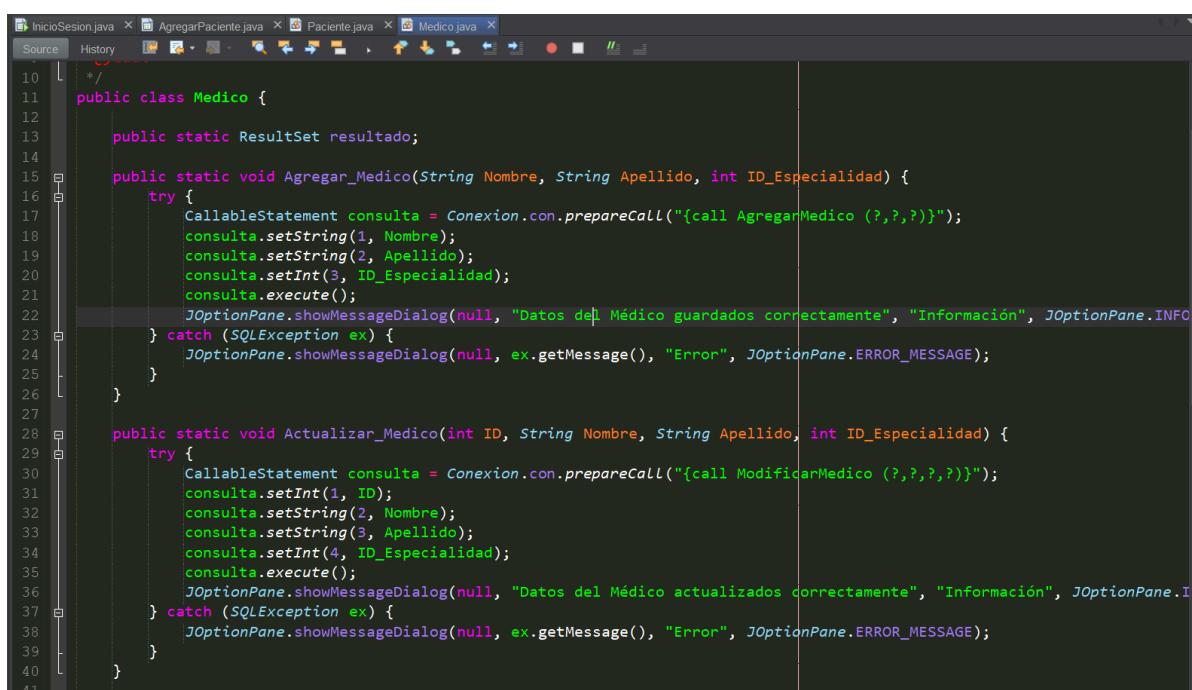


```

584     public void Guardar(){
585         if(ckL.isSelected()){
586             int HRl = cmbDesde.getSelectedIndex();
587             int HRs = cmbHasta.getSelectedIndex();
588
589             if(HRl==0 || HRs==0){
590                 JOptionPane.showMessageDialog(this, "Seleccione la hora correctamente", "Seleccione", JOptionPane.ERROR_MESSAGE);
591                 return;
592             }
593
594             if(HRl>HRs){
595                 JOptionPane.showMessageDialog(this, "La hora de salida el Lunes no puede ser menor que la de entrada, y la hora de entrada no puede ser mayor que la de salida", "Seleccione correctamente", JOptionPane.ERROR_MESSAGE);
596                 return;
597             }
598
599             if(ckM.isSelected()){
600                 int HRl2 = cmbDesde2.getSelectedIndex();
601                 int HRs2 = cmbHasta2.getSelectedIndex();
602
603                 if(HRl2==0 || HRs2==0){
604                     JOptionPane.showMessageDialog(this, "Seleccione la hora correctamente", "Seleccione", JOptionPane.ERROR_MESSAGE);
605                     return;
606                 }
607
608                 if(HRl2>HRs2){
609                     JOptionPane.showMessageDialog(this, "La hora de salida el Martes no puede ser menor que la de entrada, y la hora de entrada no puede ser mayor que la de salida", "Seleccione correctamente", JOptionPane.ERROR_MESSAGE);
610                     return;
611                 }
612             }
613         }
614     }

```

se muestran los métodos del médico.



```

10 */
11 public class Medico {
12
13     public static ResultSet resultado;
14
15     public static void Agregar_Medico(String Nombre, String Apellido, int ID_Especialidad) {
16         try {
17             CallableStatement consulta = Conexion.con.prepareCall("{call AgregarMedico (?, ?, ?)}");
18             consulta.setString(1, Nombre);
19             consulta.setString(2, Apellido);
20             consulta.setInt(3, ID_Especialidad);
21             consulta.execute();
22             JOptionPane.showMessageDialog(null, "Datos del Médico guardados correctamente", "Información", JOptionPane.INFORMATION_MESSAGE);
23         } catch (SQLException ex) {
24             JOptionPane.showMessageDialog(null, ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
25         }
26     }
27
28     public static void Actualizar_Medico(int ID, String Nombre, String Apellido, int ID_Especialidad) {
29         try {
30             CallableStatement consulta = Conexion.con.prepareCall("{call ModificarMedico (?, ?, ?, ?)}");
31             consulta.setInt(1, ID);
32             consulta.setString(2, Nombre);
33             consulta.setString(3, Apellido);
34             consulta.setInt(4, ID_Especialidad);
35             consulta.execute();
36             JOptionPane.showMessageDialog(null, "Datos del Médico actualizados correctamente", "Información", JOptionPane.INFORMATION_MESSAGE);
37         } catch (SQLException ex) {
38             JOptionPane.showMessageDialog(null, ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
39         }
40     }

```

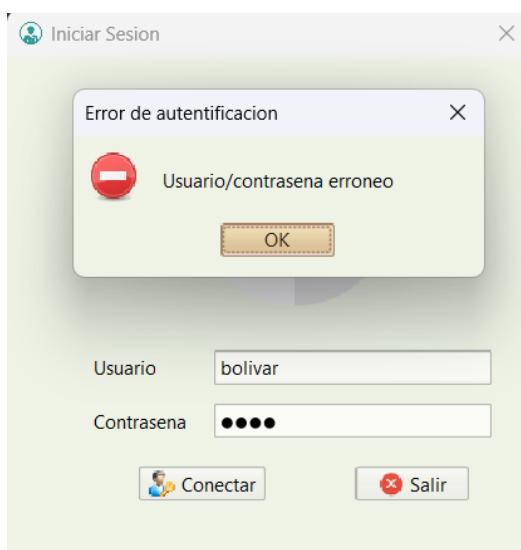
PRUEBAS (CASOS DE PRUEBAS)

PRUEBA	DESCRIPCIÓN	ENTRADAS	RESULTADO ESPERADO	ESTADO
1	Inicio de login exitoso con su correo valido password	usuario: admin contraseña: 1234	Se inicia correctamente el login y se dirige al menú principal	PASADO
2	Inicio de correo no válida o de lo contrario password no válido	usuario: bolívar contraseña: 123456	Detecta el error de correo y password como se espera	PASADO

Prueba 1:



Prueba 2:

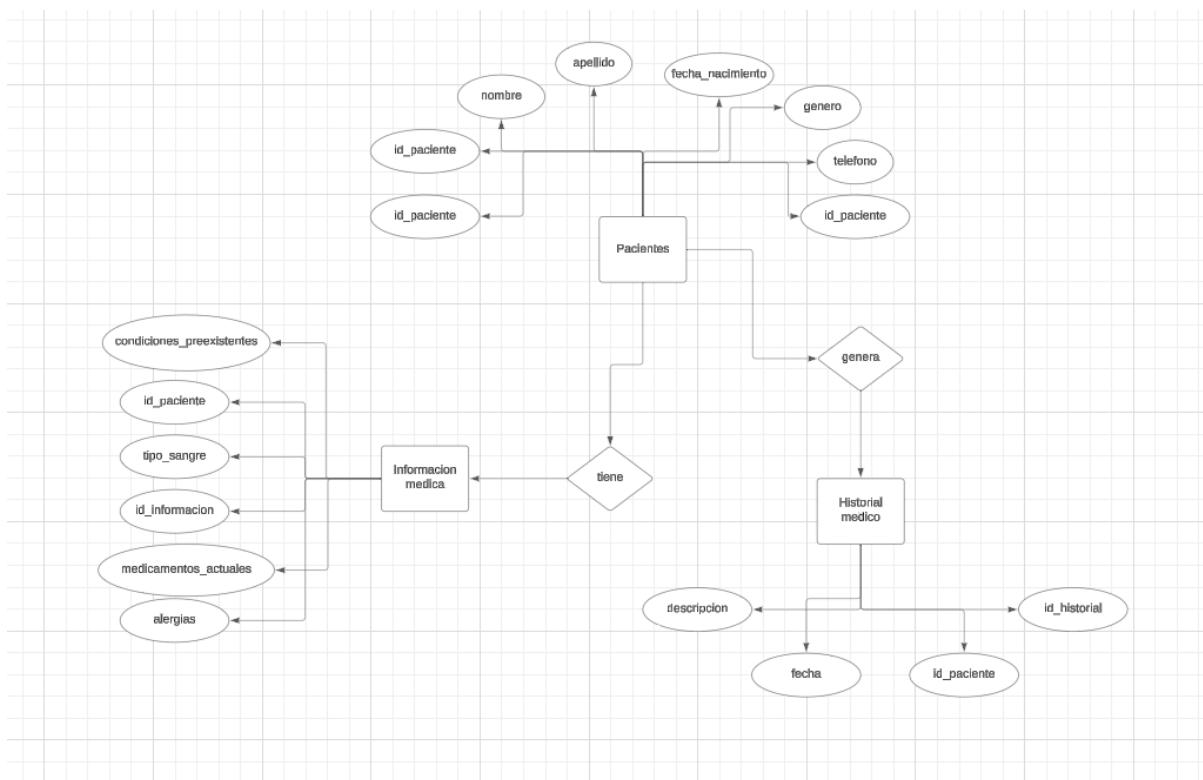


2DA Historia de usuario

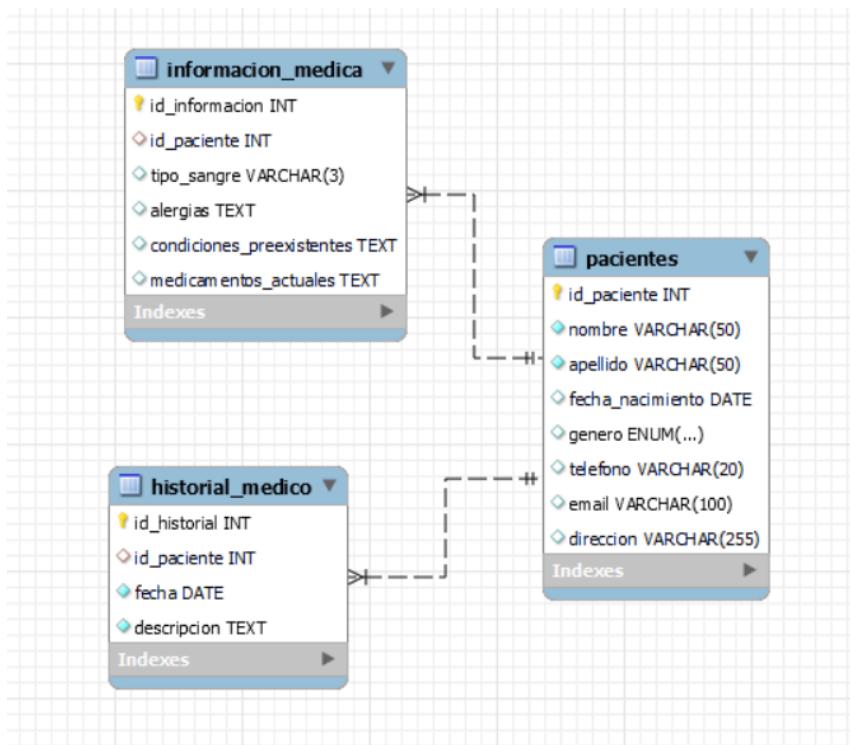
Historia de Usuario	Análisis	Puntos Estimados
Registrar y almacenar información del paciente	Registrar los datos personales y médicos del paciente, asegurando privacidad y precisión	4
	Implementar medidas de seguridad para cumplir con las normativas de protección de datos	2
	Crear un historial médico centralizado	2

HISTORIA DE USUARIO	
Número: 2	Usuario: Sistema
Nombre de Historia: Registrar y almacenar información del paciente	
Prioridad: Alta	Riesgo de desarrollo: Bajo
Puntos asignados: 4	Iteración asignada: 1
Programador responsable: Bolívar	
Descripción: El sistema debe registrar los datos personales y médicos del paciente, asegurando la privacidad y la precisión de la información. Esto permitirá acceder a un historial médico centralizado para futuras consultas.	
Observaciones: Implementar medidas de seguridad para cumplir con las normativas de protección de datos.	

MODELO ENTIDAD RELACIÓN



MODELO RELACIONAL



FASE DISEÑO TARJETAS CRC

Registro de pacientes			
Responsabilidades		Colaboraciones	
Datos:	Acciones:	Clases:	Multiplicidad:
Paciente Médico	<p>- Registrar los datos personales y médicos del paciente</p> <p>- Asegurar la privacidad y precisión de la información</p> <p>- Crear un historial médico centralizado para futuras consultas</p>	Sistema, Paciente	1..1

DISEÑO

Se hace el registro de pacientes con estos datos solicitados.

Se muestra el historial del paciente.

Ver Paciente

Buscar:

No.	Nombres	Apellidos	Peso	Altura	Edad	Telefono	Alergias	Enfermed...	Tipo Sang...	Estado
1	bolviar	n	90.0	135.0	23	1234-5678	n	n	O+	Activo
2	KEVIN	LARA	80.0	175.0	24	1234-5678	NINGUNA	NINGUNA	A+	Activo

Ver Expediente Activar/Desactivar Modificar Salir

Se muestra el registro de médicos integrados al sistema y en un apartado se muestra el estado del médico activo e inactivo.

Ver Medico

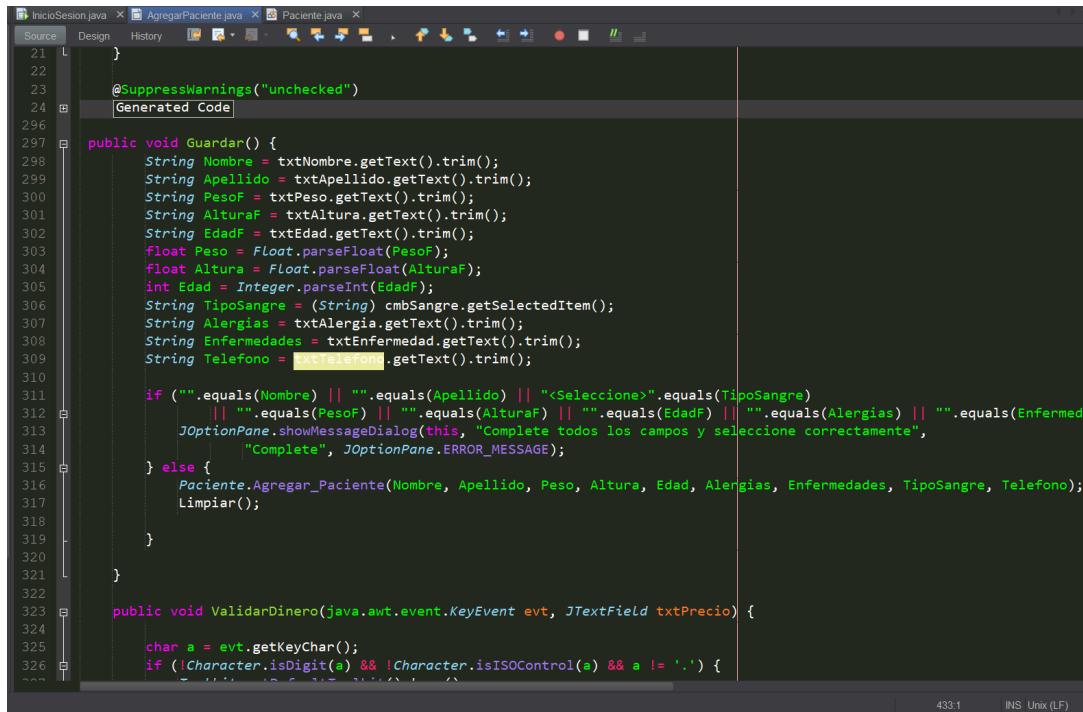
Buscar:

No.	Nombres	Apellidos	Especialidad	Estado
1	YEUDIEL AUGUSTO	HIDALGO TEJADA	Especialista	Activo
2	BOLIVAR	GARCIA OSORIO	Especialista	Activo

Activar/Desactivar Modificar Salir

DESARROLLO

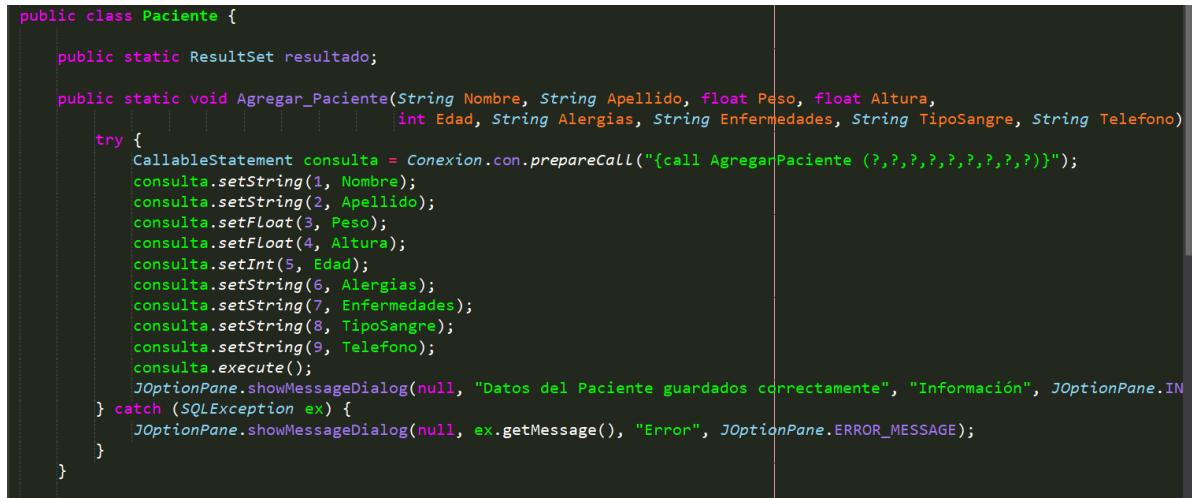
Esta es la parte en la que se guardan los datos de los pacientes.



The screenshot shows a Java code editor with several tabs at the top: 'IniciarSesion.java', 'AgregarPaciente.java', and 'Paciente.java'. The code is in 'Paciente.java' and defines a 'Guardar()' method. It reads various fields from text inputs and dropdowns, then checks if all fields are filled and correctly selected. If so, it calls a static method 'Agregar_Paciente()' from the same class. The code uses annotations like '@SuppressWarnings("unchecked")' and handles exceptions for non-digit input in a JTextField.

```
21 }
22
23     @SuppressWarnings("unchecked")
24     Generated Code
25
26     public void Guardar() {
27         String Nombre = txtNombre.getText().trim();
28         String Apellido = txtApellido.getText().trim();
29         String PesoF = txtPeso.getText().trim();
30         String AlturaF = txtAltura.getText().trim();
31         String EdadF = txtEdad.getText().trim();
32         float Peso = Float.parseFloat(PesoF);
33         float Altura = Float.parseFloat(AlturaF);
34         int Edad = Integer.parseInt(EdadF);
35         String TipoSangre = (String) cmbSangre.getSelectedItem();
36         String Alergias = txtAlergia.getText().trim();
37         String Enfermedades = txtEnfermedad.getText().trim();
38         String Telefono = txtTelefono.getText().trim();
39
40         if ("".equals(Nombre) || "".equals(Apellido) || "<Seleccione>".equals(TipoSangre)
41             || "".equals(PesoF) || "".equals(AlturaF) || "".equals(EdadF) || "".equals(Alergias) || "".equals(Enfermedades))
42             JOptionPane.showMessageDialog(this, "Complete todos los campos y seleccione correctamente",
43                 "Complete", JOptionPane.ERROR_MESSAGE);
44         } else {
45             Paciente.Agregar_Paciente(Nombre, Apellido, Peso, Altura, Edad, Alergias, Enfermedades, TipoSangre, Telefono);
46             Limpiar();
47         }
48     }
49
50     public void ValidarDinero(java.awt.event.KeyEvent evt, JTextField txtPrecio) {
51
52         char a = evt.getKeyChar();
53         if (!Character.isDigit(a) && !Character.isISOControl(a) && a != '.') {
54             evt.consume();
55         }
56     }
57 }
```

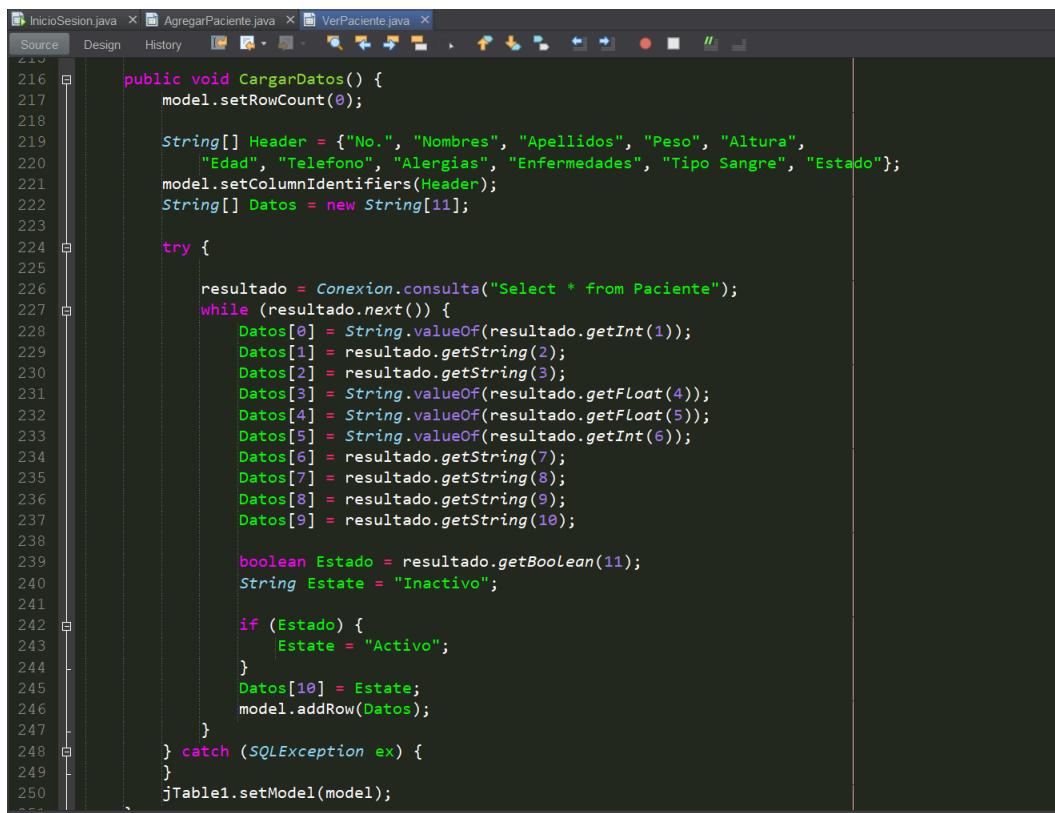
Esta es la condición para guardar los datos junto con la consulta.



The screenshot shows the 'Paciente.java' code again, specifically the 'Agregar_Paciente()' method. This is a static CallableStatement that prepares a call to a stored procedure named 'AgregarPaciente'. It sets various parameters (Nomebre, Apellido, Peso, Altura, Edad, Alergias, Enfermedades, TipoSangre, Telefono) and then executes the stored procedure. It handles both successful execution and SQL exceptions by displaying appropriate messages.

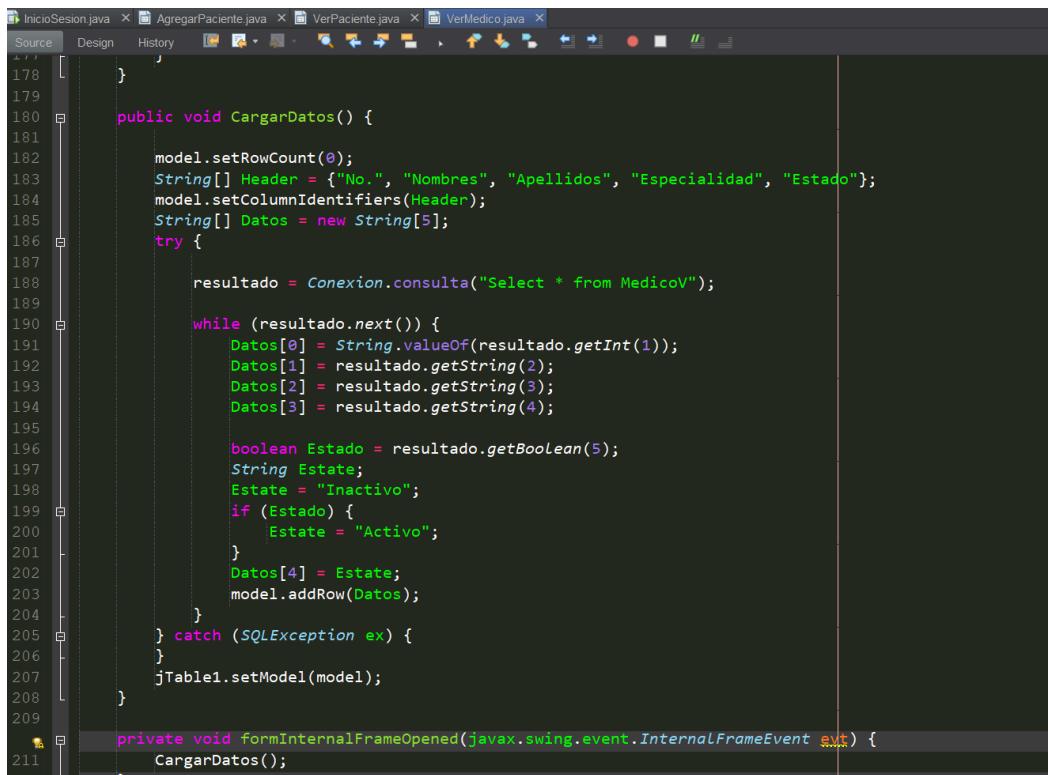
```
public class Paciente {
    public static ResultSet resultado;
    public static void Agregar_Paciente(String Nombre, String Apellido, float Peso, float Altura,
                                         int Edad, String Alergias, String Enfermedades, String TipoSangre, String Telefono)
    try {
        CallableStatement consulta = Conexion.con.prepareCall("{call AgregarPaciente (?, ?, ?, ?, ?, ?, ?, ?)}");
        consulta.setString(1, Nombre);
        consulta.setString(2, Apellido);
        consulta.setFloat(3, Peso);
        consulta.setFloat(4, Altura);
        consulta.setInt(5, Edad);
        consulta.setString(6, Alergias);
        consulta.setString(7, Enfermedades);
        consulta.setString(8, TipoSangre);
        consulta.setString(9, Telefono);
        consulta.execute();
        JOptionPane.showMessageDialog(null, "Datos del Paciente guardados correctamente", "Información", JOptionPane.INFORMATION_MESSAGE);
    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
}
```

Esta es la consulta para que muestre los pacientes.



```
216     public void CargarDatos() {
217         model.setRowCount(0);
218
219         String[] Header = {"No.", "Nombres", "Apellidos", "Peso", "Altura",
220                           "Edad", "Telefono", "Alergias", "Enfermedades", "Tipo Sangre", "Estado"};
221         model.setColumnIdentifiers(Header);
222         String[] Datos = new String[11];
223
224         try {
225
226             resultado = Conexion.consulta("Select * from Paciente");
227             while (resultado.next()) {
228                 Datos[0] = String.valueOf(resultado.getInt(1));
229                 Datos[1] = resultado.getString(2);
230                 Datos[2] = resultado.getString(3);
231                 Datos[3] = String.valueOf(resultado.getFloat(4));
232                 Datos[4] = String.valueOf(resultado.getFloat(5));
233                 Datos[5] = String.valueOf(resultado.getInt(6));
234                 Datos[6] = resultado.getString(7);
235                 Datos[7] = resultado.getString(8);
236                 Datos[8] = resultado.getString(9);
237                 Datos[9] = resultado.getString(10);
238
239                 boolean Estado = resultado.getBoolean(11);
240                 String Estate = "Inactivo";
241
242                 if (Estado) {
243                     Estate = "Activo";
244                 }
245                 Datos[10] = Estate;
246                 model.addRow(Datos);
247             }
248         } catch (SQLException ex) {
249         }
250         jTable1.setModel(model);
251     }
```

En esta parte se muestra el registro de cada médico integrado al sistema.

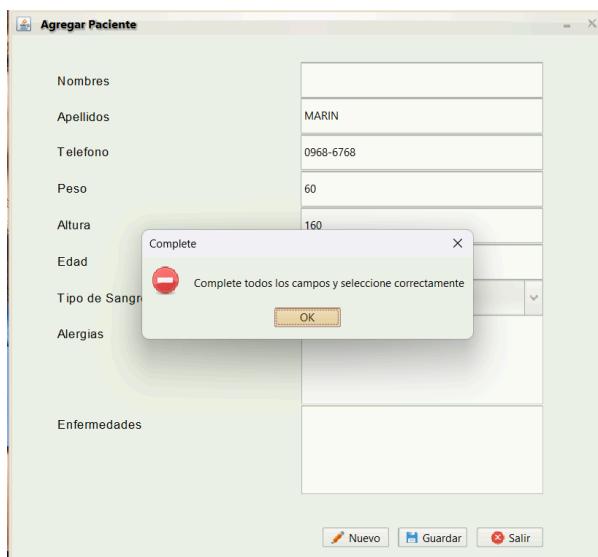


```
178     }
179
180     public void CargarDatos() {
181
182         model.setRowCount(0);
183         String[] Header = {"No.", "Nombres", "Apellidos", "Especialidad", "Estado"};
184         model.setColumnIdentifiers(Header);
185         String[] Datos = new String[5];
186         try {
187
188             resultado = Conexion.consulta("Select * from MedicoV");
189
190             while (resultado.next()) {
191                 Datos[0] = String.valueOf(resultado.getInt(1));
192                 Datos[1] = resultado.getString(2);
193                 Datos[2] = resultado.getString(3);
194                 Datos[3] = resultado.getString(4);
195
196                 boolean Estado = resultado.getBoolean(5);
197                 String Estate;
198                 Estate = "Inactivo";
199                 if (Estado) {
200                     Estate = "Activo";
201                 }
202                 Datos[4] = Estate;
203                 model.addRow(Datos);
204             }
205         } catch (SQLException ex) {
206         }
207         jTable1.setModel(model);
208     }
209
210     private void formInternalFrameOpened(javax.swing.event.InternalFrameEvent evt) {
211         CargarDatos();
212     }
213 }
```

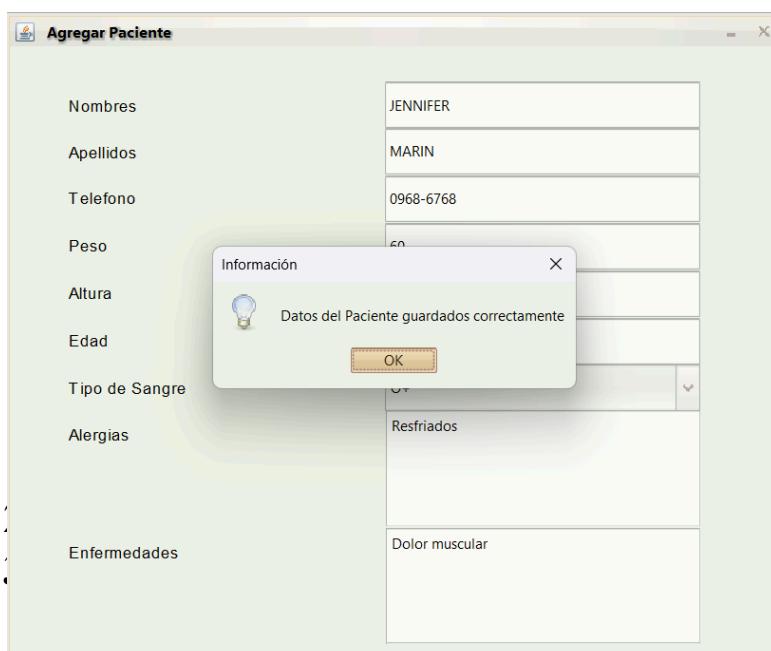
PRUEBAS (CASOS DE PRUEBAS)

PRUEBA	DESCRIPCIÓN	ENTRADAS	RESULTADO ESPERADO	ESTADO
1	Validar campos del paciente ingresando datos	Campos con datos ingresados	Guarda los datos ingresados	PASADO
2	Validar campos del paciente ingresando datos	Campos vacíos	Detecta el error en que cada campo se encuentra vacío	PASADO

Prueba 1:



Prueba 2:

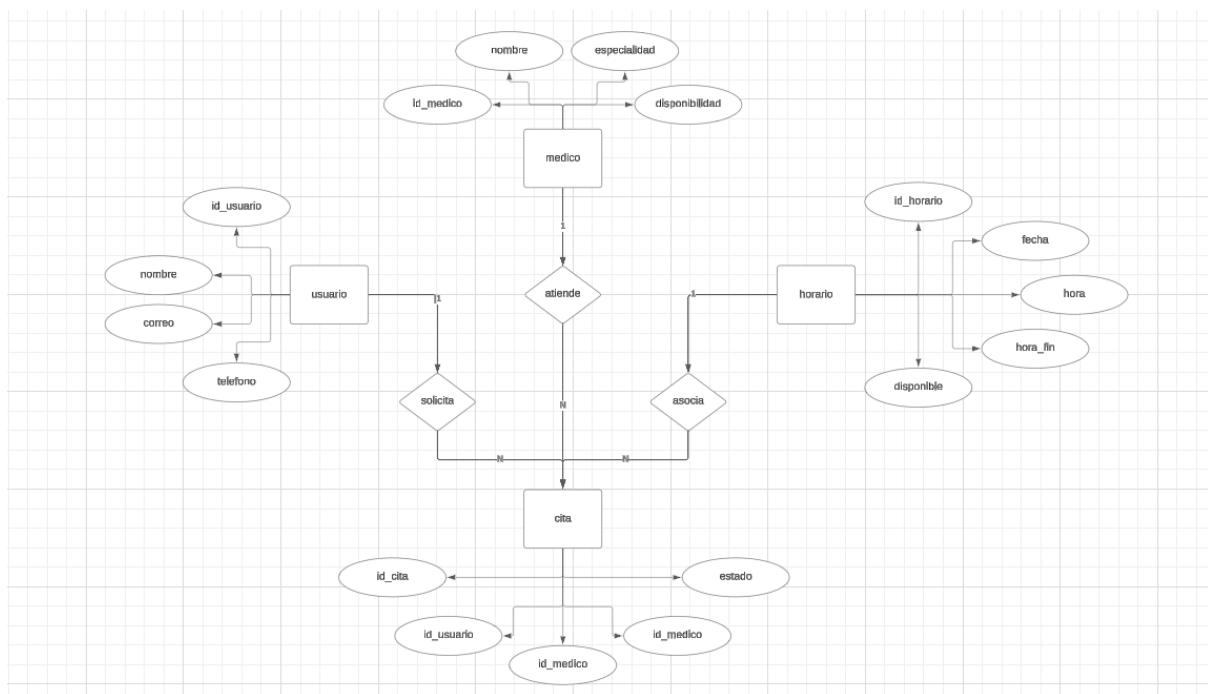


Historia de Usuario	Análisis	Puntos Estimados
Ver disponibilidad del médico	El sistema necesita verificar la disponibilidad del médico en tiempo real para asignar horarios de consulta sin conflictos.	3
	Esto permitirá a los pacientes escoger horarios que se adapten a sus necesidades.	2

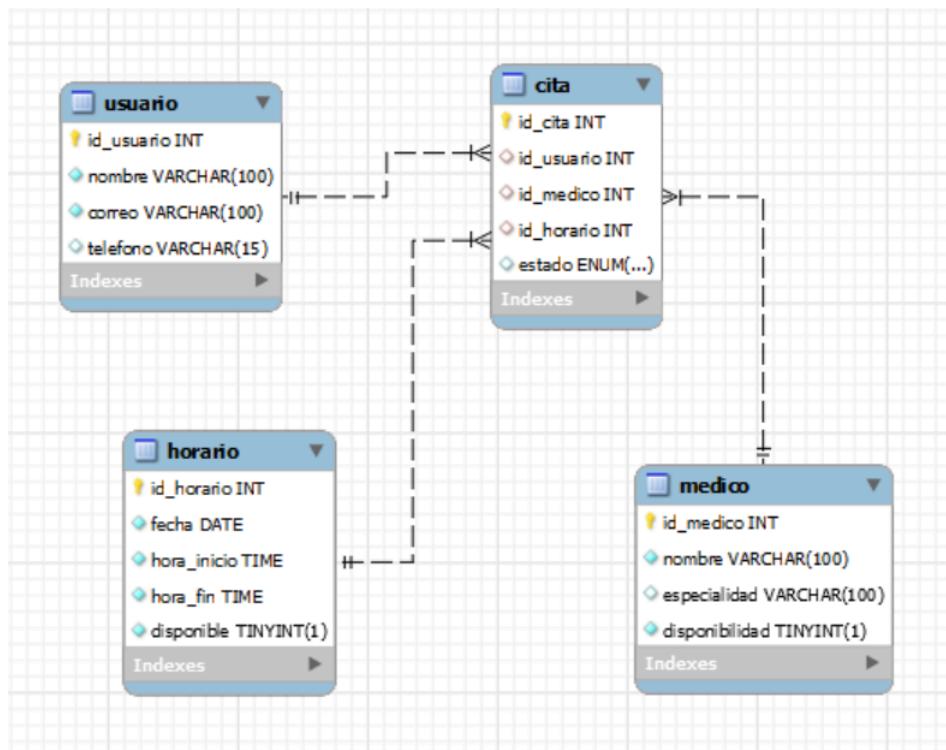


HISTORIA DE USUARIO	
Numero: 3	Usuario: Sistema
Nombre de Historia: Verificar disponibilidad del médico	
Prioridad: Alta	Riesgo de desarrollo: Bajo
Puntos asignados: 3	Iteración asignada: 2
Programador responsable: Yeudiel	
Descripción: El sistema necesita verificar la disponibilidad del médico en tiempo real para asignar horarios de consulta sin conflictos. Esto permitirá a los pacientes escoger horarios que se adapten a sus necesidades.	
Observaciones: Implementar lógica para actualizar la disponibilidad en caso de cambios o cancelaciones de citas.	

MODELO ENTIDAD RELACIÓN



UML



TARJETA CRC

Verificación de disponibilidad del médico			
Responsabilidades		Colaboraciones	
Datos:	Acciones:	Clases:	Multiplicidad:
Disponibilidad del médico	- Verificar disponibilidad en tiempo real. - Asignar horarios sin conflictos.	- Agenda - Paciente - Notificaciones - Citas	1..1
Horarios de consulta	- Permitir a los pacientes escoger horarios.		
Estado de las citas	- Actualizar disponibilidad en caso de cambios o cancelaciones de citas.		

DISEÑO

Se puede observar el paciente, la especialidad del doctor y el doctor, así como su horario de disponibilidad y los días que este labora

Hora	Estado
8:00 A.M	Disponible
9:00 A.M	Disponible
10:00 A.M	Disponible
11:00 A.M	Disponible
12:00 P.M	Disponible
1:00 P.M	No Disponible
2:00 P.M	No Disponible
3:00 P.M	No Disponible
4:00 P.M	No Disponible
5:00 P.M	No Disponible

Si escogemos un dia que el doctor no labora, automáticamente saltará la alerta y los horarios aparecerán como “no disponible”

Agregar Cita

Paciente	Usuariopr Usuarioapellido																																										
Especialidad	especial																																										
Medico	Dogtor Dogtor																																										
Días disponibles	<input type="checkbox"/> L <input type="checkbox"/> M <input type="checkbox"/> X <input type="checkbox"/> J <input type="checkbox"/> V <input type="checkbox"/> S <input type="checkbox"/> D																																										
Seleccione la fecha de la cita en los días disponibles del medico noviembre <input type="button" value="▼"/> <input type="button" value="▲"/> <table border="1" style="margin-left: 10px;"> <tr> <th>dom</th> <th>lun</th> <th>mar</th> <th>miér</th> <th>jue</th> <th>viér</th> <th>sab</th> </tr> <tr> <td>44</td> <td>3</td> <td>4</td> <td>5</td> <td>6</td> <td>7</td> <td>8</td> </tr> <tr> <td>45</td> <td>10</td> <td>11</td> <td>12</td> <td>13</td> <td>14</td> <td>15</td> </tr> <tr> <td>46</td> <td>17</td> <td>18</td> <td>19</td> <td>20</td> <td>21</td> <td>22</td> </tr> <tr> <td>47</td> <td>24</td> <td>25</td> <td>26</td> <td>27</td> <td>28</td> <td>29</td> </tr> <tr> <td>48</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>30</td> </tr> </table>		dom	lun	mar	miér	jue	viér	sab	44	3	4	5	6	7	8	45	10	11	12	13	14	15	46	17	18	19	20	21	22	47	24	25	26	27	28	29	48						30
dom	lun	mar	miér	jue	viér	sab																																					
44	3	4	5	6	7	8																																					
45	10	11	12	13	14	15																																					
46	17	18	19	20	21	22																																					
47	24	25	26	27	28	29																																					
48						30																																					

Seleccione la hora de la cita en la tabla de horario

Hora	Estado
8:00 A.M	No Disponible
9:00 A.M	No Disponible
A.M	No Disponible
A.M	No Disponible
P.M	No Disponible
1:00 P.M	No Disponible
2:00 P.M	No Disponible
3:00 P.M	No Disponible
4:00 P.M	No Disponible
5:00 P.M	No Disponible

Día no disponible

El médico no está disponible en el día seleccionado.

DESARROLLO

(capturas del código)

En esta parte del código guardamos la información/fecha y hora seleccionada para agendar la cita.

```
public void Guardar() {  
  
    int cmbPac = cmbPaciente.getSelectedIndex();  
    int cmbMed = cmbMedico.getSelectedIndex();  
    int ID_Paciente = ID_Pac[cmbPac];  
    int ID_Medico = ID_Med[cmbMed];  
  
    Date Fecha = jCalendar1.getDate();  
  
    int fila = jTable1.getSelectedRow();  
  
    if (cmbMed == 0 || cmbPac == 0 || Fecha == null) {  
        JOptionPane.showMessageDialog(this, "Complete todos los campos y seleccione correctamente",  
        "Complete", JOptionPane.ERROR_MESSAGE);  
        return;  
    }  
    if (fila < 0) {  
        JOptionPane.showMessageDialog(this, "Seleccione la hora de la cita",  
        "Seleccione", JOptionPane.ERROR_MESSAGE);  
    } else {  
  
        if (DiaDisponible()) {  
  
            String Estado = model.getValueAt(fila, 1).toString();  
            String Hora = model.getValueAt(fila, 0).toString();  
        }  
    }  
}
```

Y aquí creamos el calendario cual la información del doctor y su horario se ve reflejada para saber que dia y hora estará disponible para poder atender un paciente

```
private void formInternalFrameOpened(javax.swing.event.InternalFrameEvent evt) {  
  
    Date Hoy = new Date();  
  
    jCalendar1.setMinSelectableDate(Hoy);  
    jCalendar1.setDate(Hoy);  
  
    flag = true;  
  
    String[] Header = {"Hora", "Estado"};  
    String[] Horas = {"8:00 A.M", "9:00 A.M", "10:00 A.M", "11:00 A.M", "12:00 P.M", "1:00 P.M",  
    "2:00 P.M", "3:00 P.M", "4:00 P.M", "5:00 P.M"};  
  
    model.setColumnIdentifiers(Header);  
    model.setRowCount(10);  
  
    for (int k = 0; k < 10; k++) {  
        String hr = Horas[k];  
        model.setValueAt(hr, k, 0);  
    }  
  
    jTable1.setModel(model);  
    jTable1.setModel(model);  
  
    TableColumnModel columnModel = jTable1.getColumnModel();  
    columnModel.getColumn(0).setPreferredWidth(130);  
    columnModel.getColumn(1).setPreferredWidth(210);  
}
```

PRUEBAS (CASOS DE PRUEBAS)

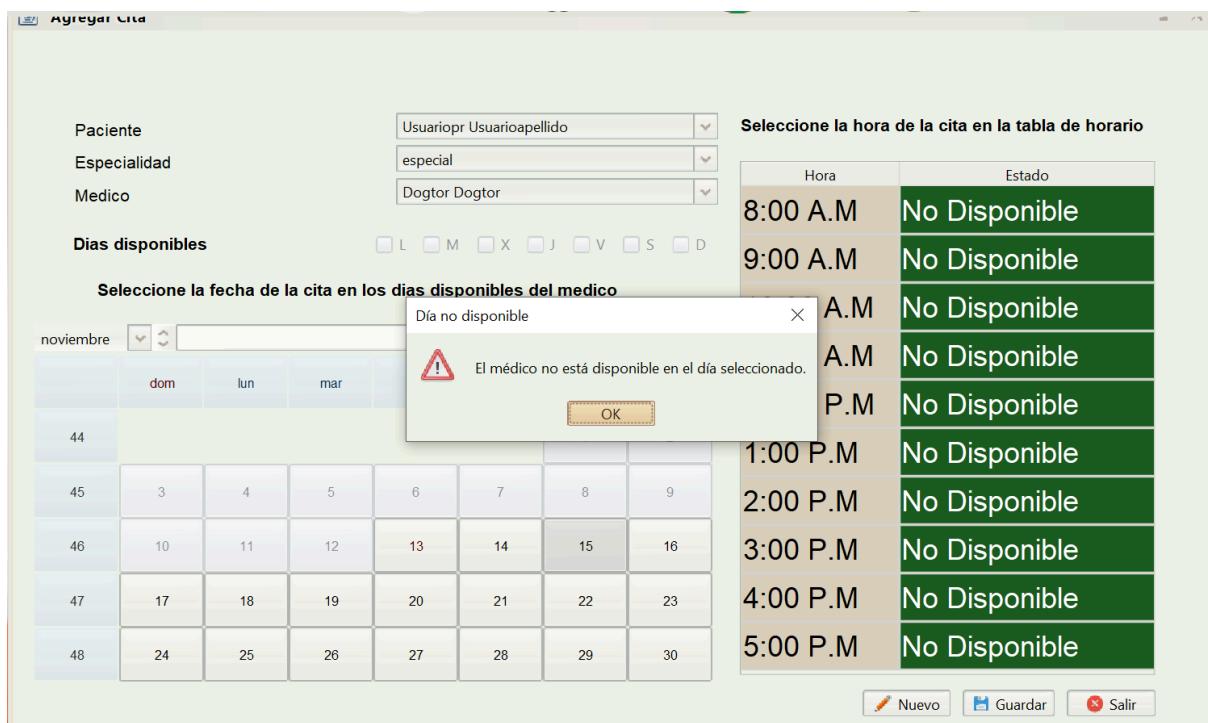
PRUEBA	DESCRIPCIÓN	ENTRADAS	RESULTADO ESPERADO	ESTADO
1	Se registró la cita con el doctor, especialidad, fecha y hora	Selección de Paciente Especialidad del doctor Médico	Se guarda correctamente la cita con los datos del doctor y paciente	ERROR no guarda
2	Se registró la cita con el doctor, especialidad, fecha y hora	No seleccionar alguna de estas opciones: Selección de Paciente Especialidad del doctor Médico Horario	No se guarda la cita con los datos del doctor y paciente	PASADO

Prueba 1: Confirmación de “cita error”

The screenshot shows the 'Agregar Cita' (Add Appointment) window. On the left, there are input fields for 'Paciente' (User), 'Especialidad' (Specialization), and 'Medico' (Doctor). Below these are checkboxes for 'Días disponibles' (Available days) and a section to 'Seleccionar la fecha de la cita en los días disponibles del médico' (Select the appointment date from the doctor's available days). A calendar grid for November 2024 shows dates from 44 to 50. On the right, a table titled 'Seleccione la hora de la cita en la tabla de horario' (Select the appointment time from the schedule table) lists times from 8:00 A.M. to 5:00 P.M., with availability status (Disponible or No Disponible).

Hora	Estado
8:00 A.M.	Disponible
9:00 A.M.	Disponible
10:00 A.M.	Disponible
11:00 A.M.	Disponible
12:00 P.M.	Disponible
1:00 P.M.	No Disponible
2:00 P.M.	No Disponible
3:00 P.M.	No Disponible
4:00 P.M.	No Disponible
5:00 P.M.	No Disponible

Prueba 2: Error al intentar guardar una cita en un horario no disponible



Historia de Usuario

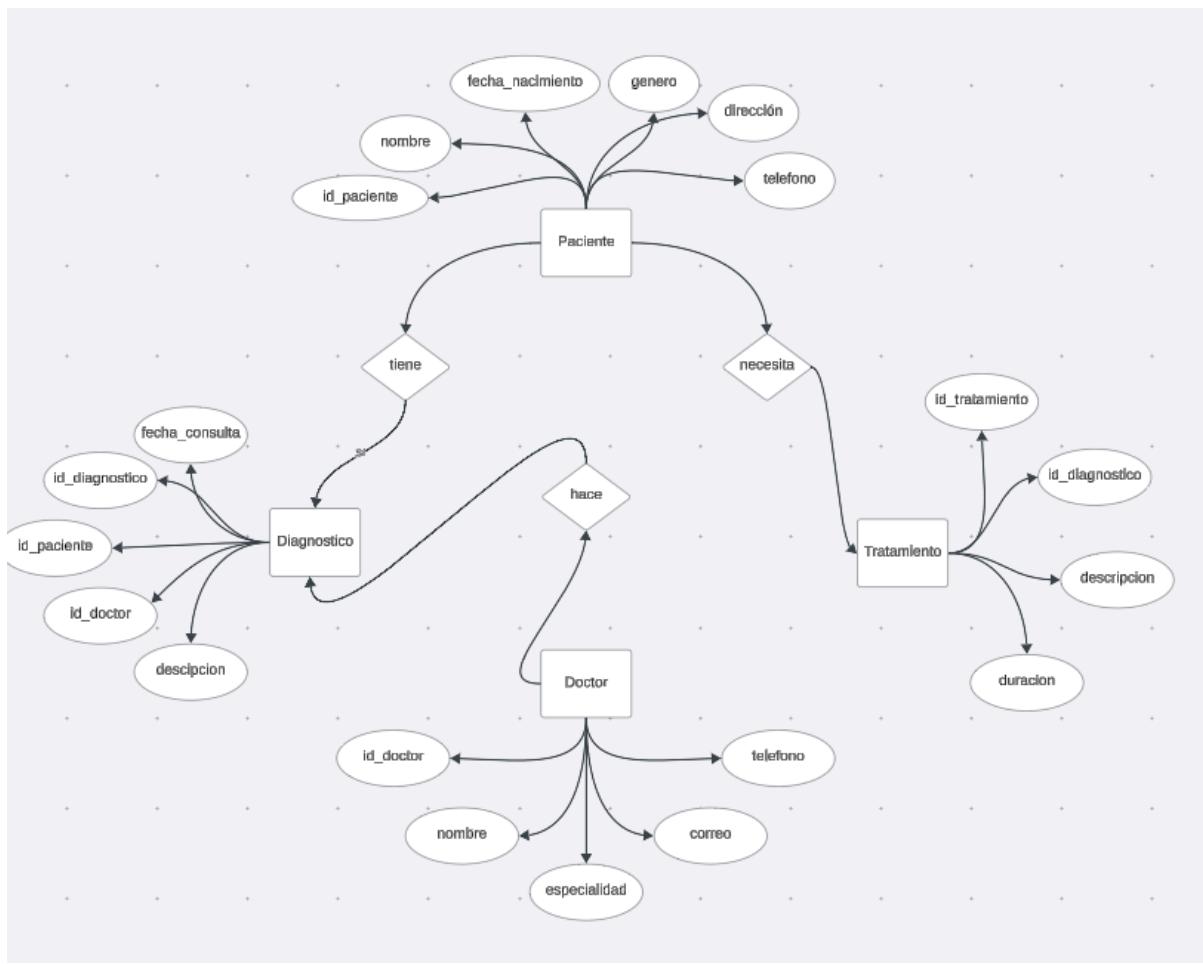
Análisis

**Puntos
Estimados**

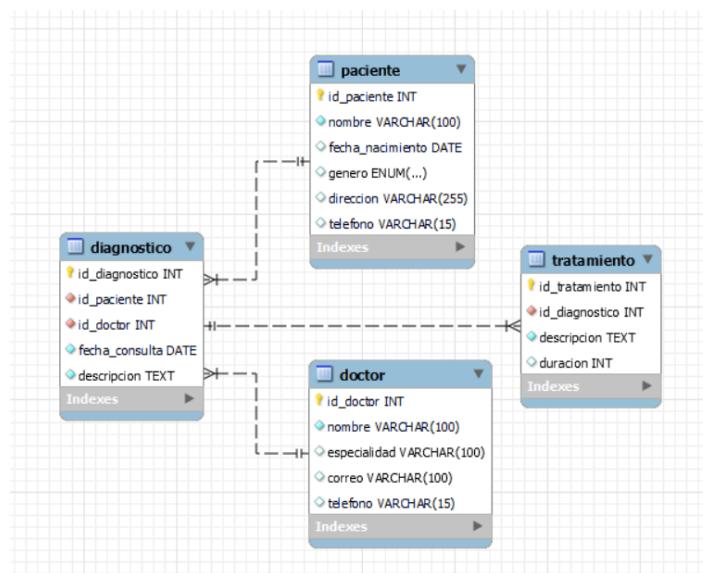
Realizar la consulta médica y registrar diagnóstico	Hacer que la consulta se realice correctamente	3
	El sistema registra el diagnóstico	5

HISTORIA DE USUARIO	
Numero: 4	Usuario: Medico
Nombre de Historia: Realizar la consulta médica y registrar diagnóstico	
Prioridad: Alta	Riesgo de desarrollo: Medio
Puntos asignados: 5	Iteración asignada: 2
Programador responsable: Bolívar	
Descripción: El médico necesita una interfaz para registrar el diagnóstico y tratamiento del paciente durante la consulta. Esta información se guarda en el expediente del paciente y debe ser accesible en futuras consultas.	
Observaciones: Facilitar la búsqueda de historial médico del paciente dentro de la interfaz.	

E-R



UML



TARJETA CRC

Verificación de disponibilidad del médico			
Responsabilidades		Colaboraciones	
Datos:	Acciones:	Clases:	Multiplicidad:
<ul style="list-style-type: none"> ● Diagnóstico ● Tratamiento ● Historial médico del paciente 	<ul style="list-style-type: none"> - Registrar diagnóstico durante la consulta. - Guardar información en el expediente médico del paciente. - Facilitar la búsqueda del historial médico en consultas futuras. 	<ul style="list-style-type: none"> - Doctor - Paciente - Diagnóstico - Historial Medico 	1..1

DISEÑO

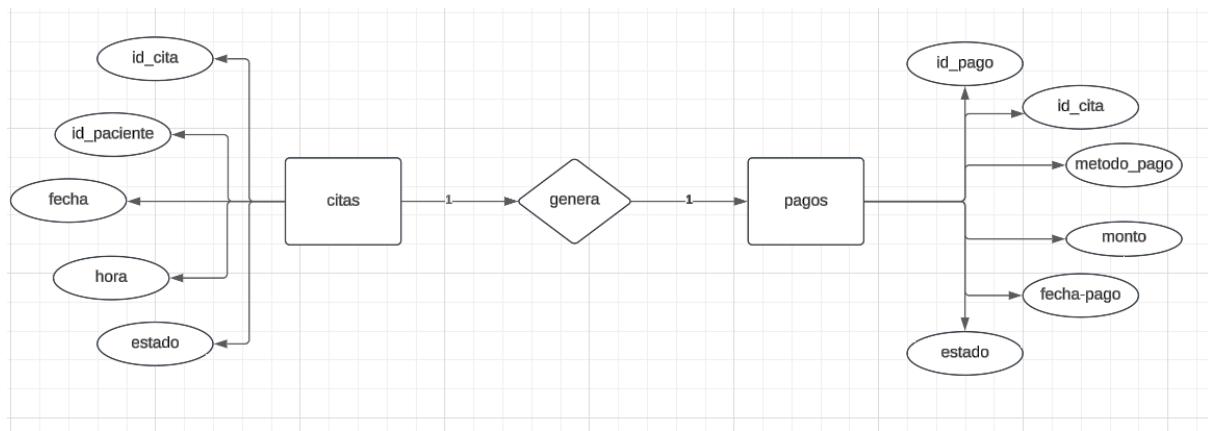
3RA ITERACIÓN

Historia de Usuario	Análisis	Puntos Estimados
Confirmar y procesar el pago de la consulta	Hacer que el sistema permita al paciente confirmar y realizar el pago de manera segura a través de la plataforma	3
	Actualizar el estado de la cita como “pagada” una vez completado el pago	5
	Obtener una notificación de confirmación del pago	2

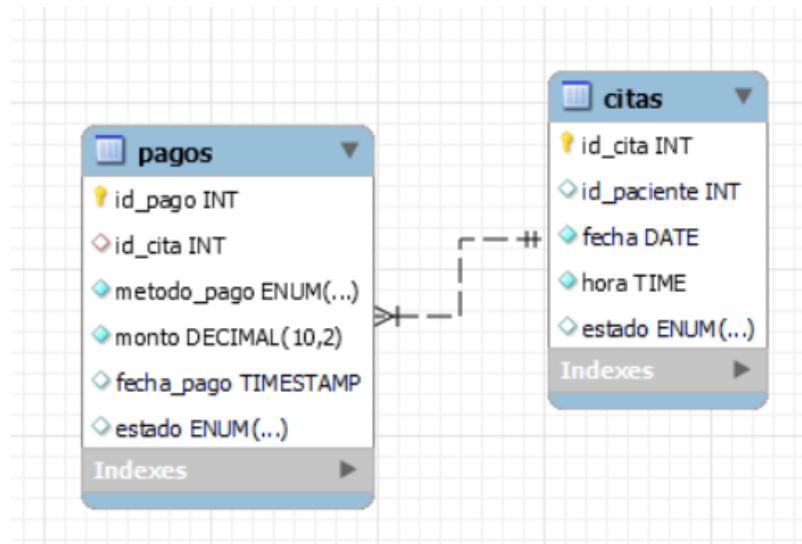


HISTORIA DE USUARIO	
Número: 5	Usuario: Paciente
Nombre de Historia: Confirmar y procesar el pago de la consulta	
Prioridad: Alta	Riesgo de desarrollo: Bajo
Puntos asignados: 3	Iteración asignada: 3
Programador responsable: Bolívar	
Descripción: El paciente debe confirmar y realizar el pago de la consulta de manera segura a través de la plataforma. Una vez completado el pago, el sistema debe actualizar el estado de la cita como “pagada” y enviar una confirmación al paciente.	
Observaciones: Asegurar métodos de pago seguros y notificaciones de confirmación pospago.	

MODELO ENTIDAD RELACIÓN



MODELO RELACIONAL



TARJETA CRC

Recepción y almacenamiento de productos			
Responsabilidades		Colaboraciones	
Datos:	Acciones:	Clases:	Multiplicidad:
Información del paciente Detalles de pago	<ul style="list-style-type: none"> - Actualizar el estado de la cita como pagada “activo” una vez completado el pago - Permitir a la caja confirmar y realizar el pago de manera segura a través del sistema 	Sistemas de pago	1..1

Diseño

Pantalla de detalles de pago: Muestra una tabla con información detallada sobre un pago específico, incluyendo el código, servicio, precio, cantidad y estado. Los datos son presentados de forma clara y organizada para facilitar su revisión



The screenshot shows a Windows application window titled "Ver Pago". At the top left is a small icon of a computer monitor. To the right of the title bar are standard window control buttons (minimize, maximize, close). Below the title bar is a search bar labeled "Buscar:" with an empty input field. The main area is a data grid with the following columns and data:

No.	Fecha	Cliente	Usuario	Total	Estado
1	16-nov-2024	Bolivare Garcia Oso...	yeudiel2	180.0	Cancelado
2	17-nov-2024	Alma M	yeudiel2	180.0	Cancelado
3	30-nov-2024	unmo uno	jenni	180.0	Cancelado
3	30-nov-2024	unmo uno	jenni	180.0	Cancelado
4	07-dic-2024	Bolivare Garcia Oso...	yeudiel2	180.0	Activo

DESARROLLO

Lógica para cargar los detalles de pago: Realiza una consulta a la base de datos utilizando el ID del pago. Los resultados se procesan y se llenan en un modelo de tabla no editable, garantizando que los datos sean confiables y consistentes para su visualización en el frontend.

```
150 if (Fila >= 0) {
151     try {
152         // Obtener el ID del pago y el estado actual
153         int ID = Integer.parseInt(model.getValueAt(Fila, 0).toString());
154         String Estado = model.getValueAt(Fila, Col).toString().trim();
155
156         // Verificar si el estado es 'Activo'
157         if (Estado.equalsIgnoreCase("Activo")) {
158             // Cancelar el pago
159             Pago.Cancelar_Pago(ID);
160             JOptionPane.showMessageDialog(this, "El pago se ha cancelado correctamente.", "Exito", JOptionPane.INFORMATION_MESSAGE);
161         } else if (Estado.equalsIgnoreCase("Cancelado")) {
162             JOptionPane.showMessageDialog(this, "Ya ha cancelado este pago anteriormente.", "Cancelado", JOptionPane.INFORMATION_MESSAGE);
163         } else {
164             JOptionPane.showMessageDialog(this, "Estado desconocido: " + Estado, "Error", JOptionPane.ERROR_MESSAGE);
165         }
166
167         // Recargar los datos
168         CargarDatos();
169
170     } catch (NumberFormatException e) {
171         JOptionPane.showMessageDialog(this, "Error al procesar el ID del pago: " + e.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
172     } catch (Exception ex) {
173         JOptionPane.showMessageDialog(this, "Error al cancelar el pago: " + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
174     }
175 } else {
176     // No se seleccionó ninguna fila
177     JOptionPane.showMessageDialog(this, "Debe seleccionar el pago a cancelar.", "Selección", JOptionPane.ERROR_MESSAGE);
}
```

El backend implementa la lógica para extraer datos desde una base de datos y organizarlos en un modelo de tabla que el frontend puede mostrar. El flujo es el siguiente:

1. Definición de la estructura de datos:
Se especifican las columnas que compondrán la tabla, incluyendo campos como código, número, servicio, precio, cantidad, total y estado del pago. Esto asegura que los datos extraídos tengan un formato coherente y estructurado.
2. Consulta a la base de datos:
Se realiza una consulta SQL para obtener todos los detalles relacionados con un pago específico, identificado por su ID. La consulta se ejecuta sobre una vista o tabla de la base de datos que contiene los detalles del pago.
3. Procesamiento de los datos:
Cada fila del resultado obtenido es procesada:
 - Se extraen valores como el código del servicio, el nombre del servicio, el precio unitario, la cantidad, y el importe total.
 - El estado del detalle, representado como un valor booleano en la base de datos, se convierte en texto legible ("Activo" o "Cancelado").
4. Carga en el modelo de datos:
Los datos procesados se organizan en filas y se agregan al modelo de tabla. Este modelo es no editable para garantizar la integridad de la información.
5. Resultado final:
El modelo de datos completo se asigna a una tabla visual en el frontend, permitiendo al usuario ver los detalles del pago de manera clara y estructurada.

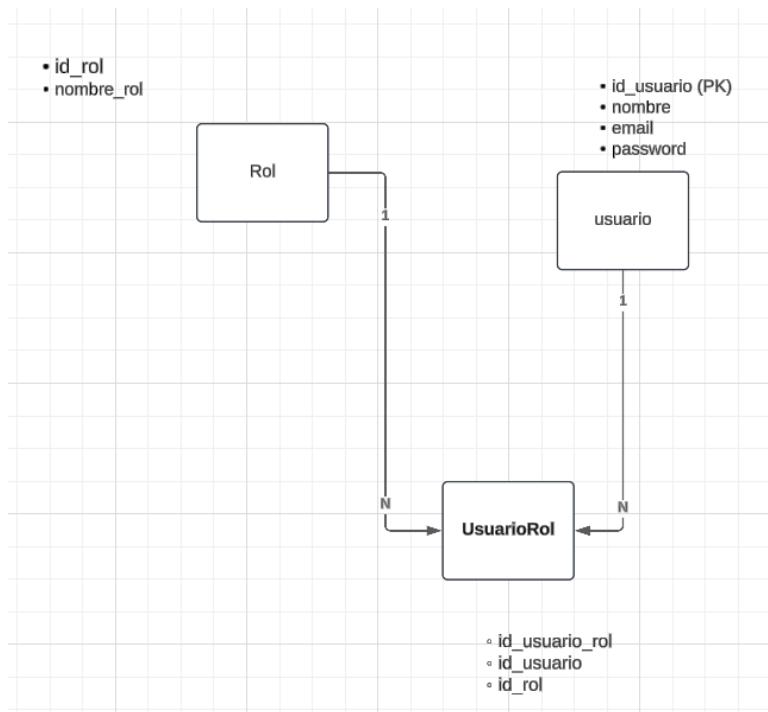
```
public void CargarDatos(int ID_Pago) {  
  
    String[] Header = {"Codigo", "No.", "Servicio", "Precio", "Cantidad", "Total", "Estado"};  
    model.setColumnIdentifiers(Header);  
    this.jTable1.setModel(model);  
    String[] Datos = new String[7];  
  
    try {  
        resultado = Conexion.consulta("Select * from Detalle_PagoV "  
            + "Where ID_Pago = " + ID_Pago);  
        while (resultado.next()) {  
  
            Datos[0] = String.valueOf(resultado.getInt(1));  
            Datos[1] = String.valueOf(resultado.getInt(2));  
            Datos[2] = resultado.getString(3);  
            Datos[3] = String.valueOf(resultado.getDouble(4));  
            Datos[4] = String.valueOf(resultado.getInt(5));  
            Datos[5] = String.valueOf(resultado.getDouble(6));  
            boolean Estado = resultado.getBoolean(7);  
            String Estate = "Cancelado";  
            if (Estado) {  
                Estate = "Activo";  
            }  
            Datos[6] = Estate;  
  
            model.addRow(Datos);  
        }  
    } catch (Exception e) {  
        System.out.println(e);  
    }  
}
```

Historia de usuario 6

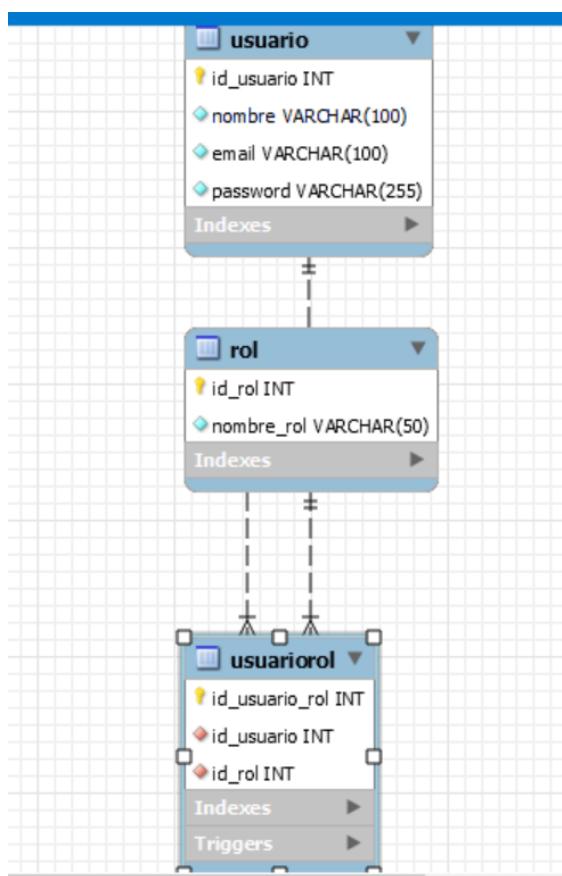
Historia de Usuario	Análisis	Puntos Estimados
Crear y asignar roles de usuario	Diseñar una interfaz para que el administrador pueda asignar roles como médico, paciente o administrativo	3
	Implementar validaciones para asegurar que cada usuario tenga los permisos correctos.	2
	Probar exhaustivamente la funcionalidad para evitar accesos no autorizados.	1

HISTORIA DE USUARIO	
Número: 6	Usuario: Administrador
Nombre de Historia: Crear y asignar roles de usuario en el sistema	
Prioridad: Media	Riesgo de desarrollo: Medio
Puntos asignados: 4	Iteración asignada: 3
Programador responsable: Yeudiel	
Descripción: El administrador necesita la capacidad de crear nuevos usuarios y asignarles roles específicos dentro del sistema, tales como médico, paciente o administrativo, con los permisos correspondientes.	
Observaciones: Asegurar que los permisos estén claramente definidos para evitar accesos no autorizados.	

MODELO ENTIDAD RELACIÓN



MODELO RELACIONAL

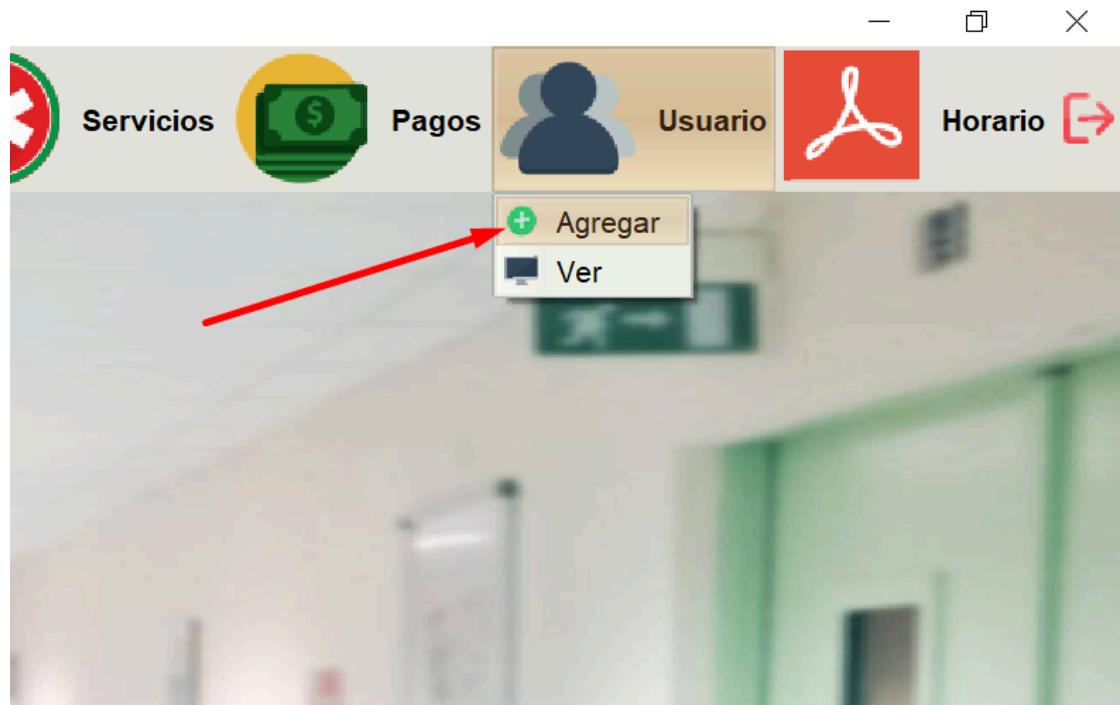


FASE DISEÑO TARJETAS CRC

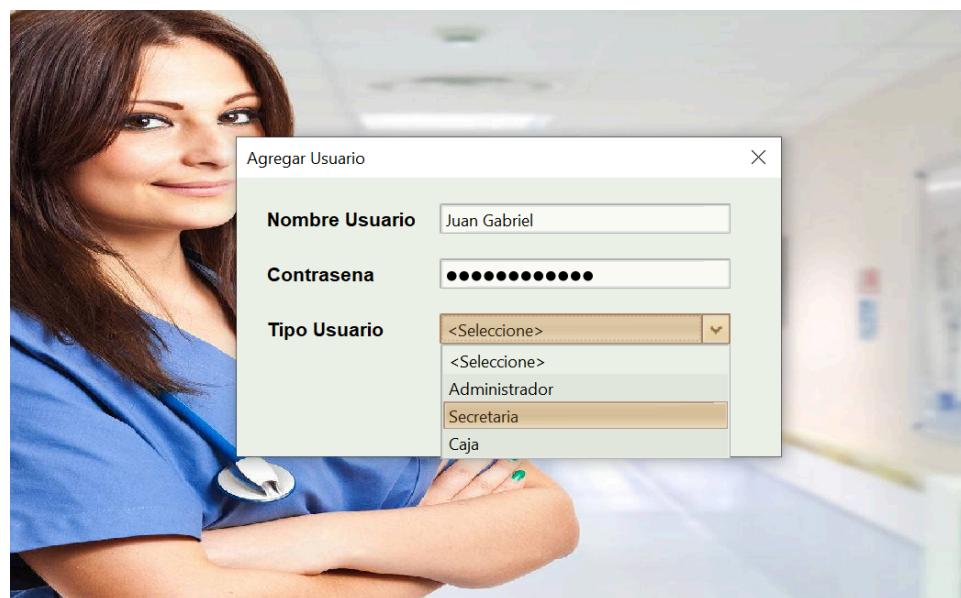
Rol de Usuario			
Responsabilidades		Colaboraciones	
Datos:	Acciones:	Clases:	Multiplicidad:
<p>Datos:</p> <ul style="list-style-type: none"> • Información del usuario (nombre, correo, rol asignado). • Permisos asociados al rol. • Estado de activación del usuario. 	<p>Acciones:</p> <ul style="list-style-type: none"> • Crear nuevos usuarios. • Asignar roles específicos (administrativo, Secretaria o Caja). • Gestionar permisos asociados a cada rol. • Validar que cada usuario tenga acceso restringido según su rol. • 	<p>Clases:</p> <p>Sistema de Usuarios. Roles. Permisos.</p>	<p>Multiplicidad:</p> <p>1..1</p>

DISEÑO

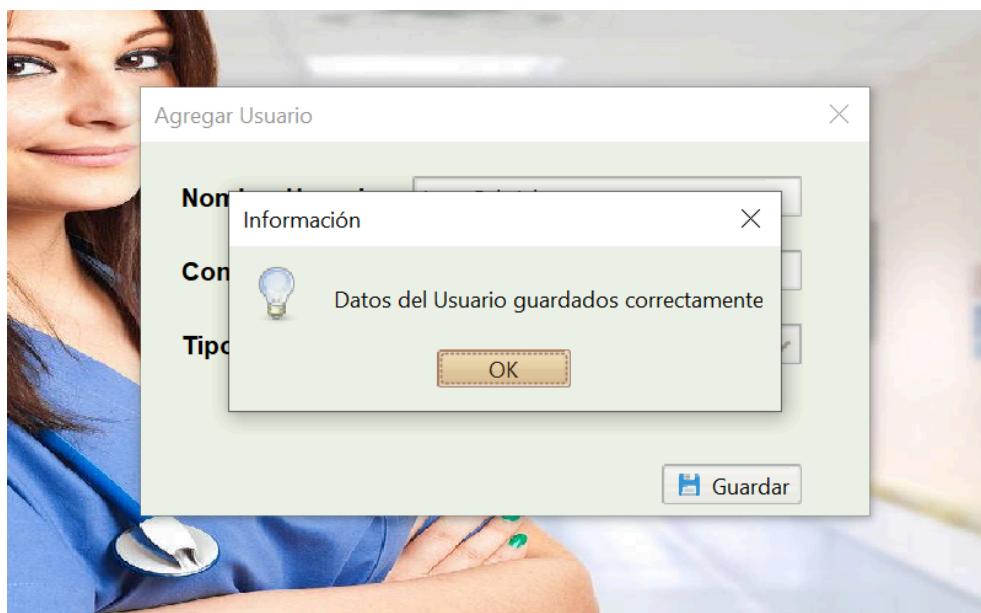
Haz clic en la opción 'Agregar Usuario' desde el menú principal para iniciar el proceso de creación de un nuevo usuario.



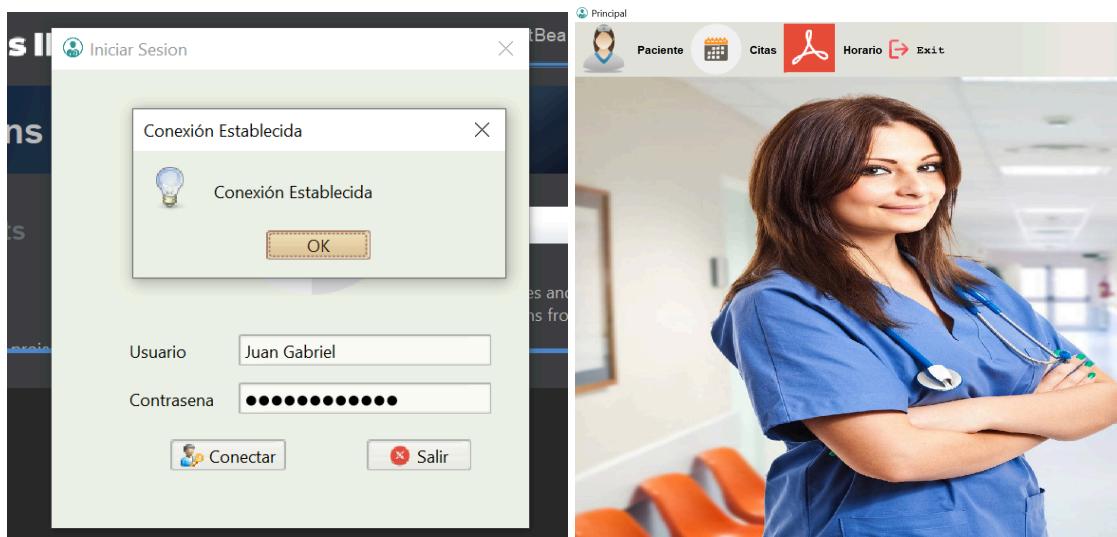
Rellena los campos requeridos, como el nombre del usuario, correo electrónico, y selecciona el rol correspondiente: Administrativo, Secretario, o Caja



Verifica que toda la información esté correcta. Una vez completado, haz clic en 'Guardar' para registrar al nuevo usuario, el usuario ha sido guardado exitosamente.



Inicia sesión con las credenciales del usuario creado. A continuación, se mostrará la interfaz con los permisos específicos asignados al rol de Secretaria.



DESARROLLO

Esta función valida los campos ingresados en el formulario de creación de usuarios. Si los campos de nombre, contraseña y rol están completos, se agrega un nuevo usuario al sistema mediante el método `Usuario.Agregar_Usuario`. Si falta algún campo, se muestra un mensaje de error al usuario. La función también asigna un rol específico en función de la selección del usuario.

```
private String NombreUsuario;
public void setNombreUsuario(String NombreUsuario) {
    this.NombreUsuario = NombreUsuario;
    this.txtUser.setText(NombreUsuario);
}
public void Guardar() {
    String Nombre = txtUser.getText().trim();
    String Pass = txtPass.getText();

    int Rol = cmbTipo.getSelectedIndex();

    if("".equals(Nombre)||"".equals(Pass) || Rol == 0){
        JOptionPane.showMessageDialog(this, "Ingrese y seleccione todos los campos",
            "Ingrese y seleccione", JOptionPane.ERROR_MESSAGE);
        return;
    }
    String Role = (String) cmbTipo.getSelectedItem();
    if(Rol == 1){
        Role = "Admin";
    }
    Usuario.Agregar_Usuario(Nombre, Pass, Role);
    this.dispose();
}
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    Guardar();
}
```

Esta función configura la visibilidad de las diferentes opciones del sistema según el rol del usuario que ha iniciado sesión. Dependiendo del rol (Caja, Médico, Secretaria, Admin), se

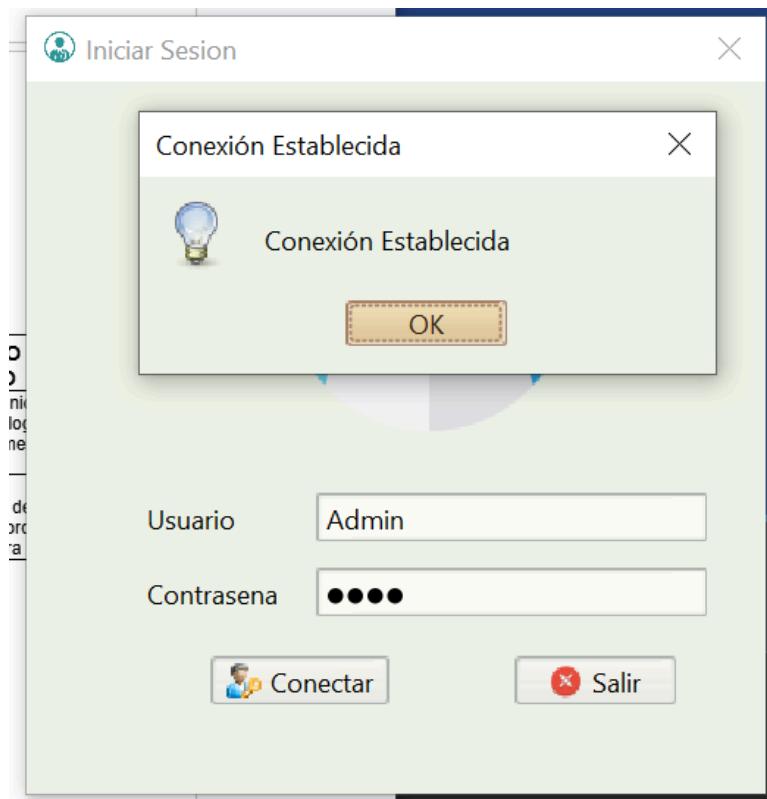
activan o desactivan los componentes de la interfaz gráfica correspondientes, garantizando que cada usuario tenga acceso solo a las funciones permitidas.

```
public void Iniciar(String Rol) {  
  
    if ("Caja".equals(Rol)) {  
        Pago.setVisible(true);  
    }  
    if ("Medico".equals(Rol)) {  
        Consulta.setVisible(true);  
        Paciente.setVisible(true);  
        AgregarPaciente.setVisible(false);  
        VerConsulta.setVisible(false);  
    }  
    if ("Secretaria".equals(Rol)) {  
        Paciente.setVisible(true);  
        Cita.setVisible(true);  
        Reporte.setVisible(true);  
    }  
    if ("Admin".equals(Rol)) {  
        Especialidad.setVisible(true);  
        Medico.setVisible(true);  
        Paciente.setVisible(true);  
        Cita.setVisible(true);  
        Consulta.setVisible(true);  
        Servicio.setVisible(true);  
        Pago.setVisible(true);  
        Usuario.setVisible(true);  
        Reporte.setVisible(true);  
    }  
}
```

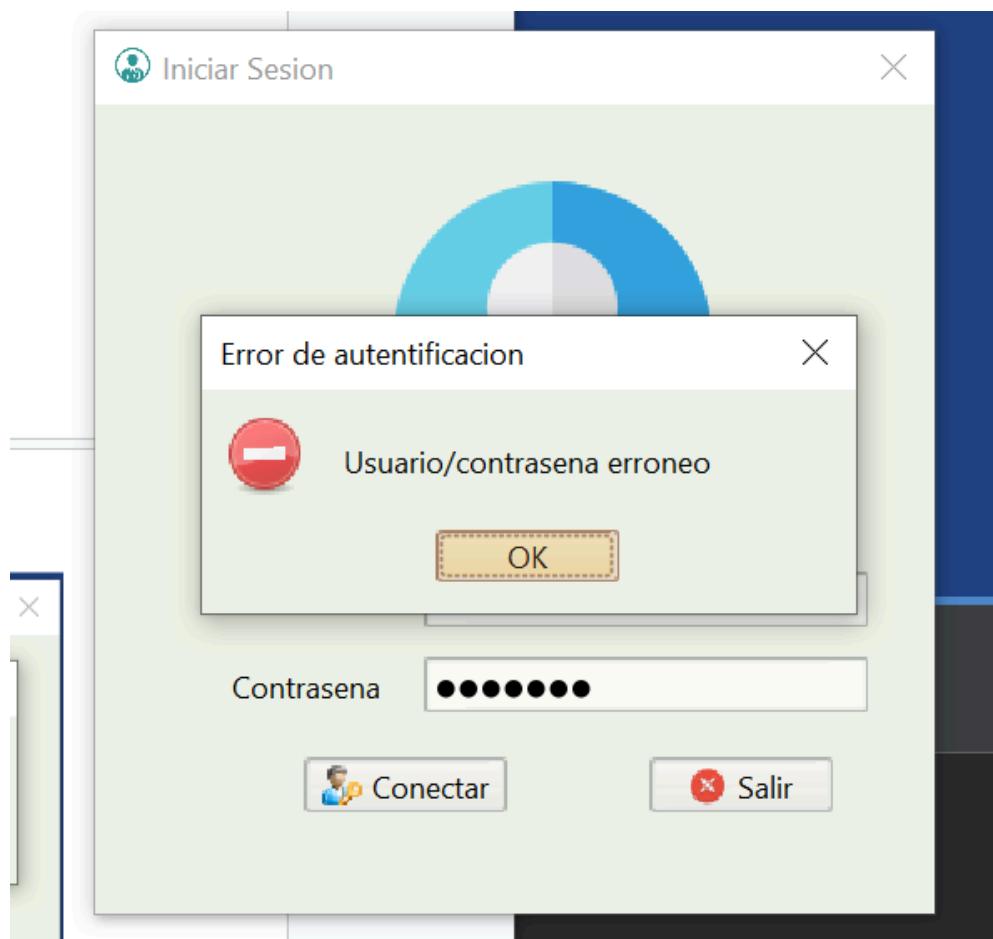
PRUEBAS (CASOS DE PRUEBAS)

PRUEBA	DESCRIPCIÓN	ENTRADAS	RESULTADO ESPERADO	ESTADO
1	Inicio de login exitoso con su correo valido password	usuario: admin contraseña: 1234	Se inicia correctamente el login y se dirige al menú principal	PASADO
2	Inicio de correo no válida o de lo contrario password no válido	usuario: bolívar contraseña: 123456	Detecta el error de correo y password como se espera	PASADO

Prueba 1:



Prueba 2:



2.7 FASE DE PRODUCCIÓN

Objetivo

Mostrar el sistema de administración de citas médicas, personal y usuarios de la clínica en un entorno real para asegurar que cumple con las expectativas de los doctores y administradores.

Actividades

- **Desarrollo del código:**
Implementación de las funcionalidades clave, como la gestión de médicos, pacientes y citas, conforme a los requerimientos establecidos.
- **Integración continua:**
Uso de herramientas para garantizar que los módulos del sistema, como el generador de citas y la base de datos relacional, funcionen correctamente en conjunto.
- **Pruebas unitarias:**
Verificación de componentes individuales, como el registro de citas y el manejo de roles administrativos.
- **Pruebas de integración:**
Validación de la interacción entre módulos, asegurando que los doctores y administradores puedan operar fluidamente.
- **Optimización del rendimiento:**
Ajustes en el sistema para garantizar tiempos de respuesta rápidos y evitar problemas relacionados con la carga del entorno.
- **Revisión del código:**
Validación del código por otros desarrolladores para mantener estándares de calidad y coherencia.

Despliegue del Código

El software se implementa en los equipos y servidores de la clínica, asegurando que el entorno de producción esté listo para el uso cotidiano.

Pruebas de Producción

- **Pruebas de carga:** Evaluar el comportamiento del sistema bajo condiciones de uso intensivo.
- **Pruebas de seguridad:** Garantizar que los datos de los pacientes y médicos estén protegidos.
- **Pruebas de funcionalidad:** Confirmar que los procesos administrativos y de citas se ejecuten correctamente en producción.

Plan de Soporte y Mantenimiento

Se diseña un plan para atender incidencias, aplicar actualizaciones y garantizar la continuidad operativa del sistema. Esto incluye monitoreo regular y canales para soporte técnico.

Entregables

- Sistema desplegado y funcional.
- Manual de usuario para doctores y administradores.
- Documentación técnica para futuros desarrolladores.
- Plan de soporte y mantenimiento operativo.

2.8 FASE DE MANTENIMIENTO

Objetivo

Asegurar la operatividad del sistema de administración de la clínica, adaptándolo a nuevos requerimientos y manteniéndolo eficiente, seguro y confiable para los usuarios.

Actividades

1. **Corrección de errores:** Solución de problemas identificados por los administradores y doctores durante el uso diario.
2. **Actualizaciones y parches:** Inclusión de mejoras, como nuevas funcionalidades (reportes estadísticos o notificaciones automáticas) y ajustes de seguridad.
3. **Mantenimiento preventivo:** Revisiones periódicas para evitar errores futuros y garantizar estabilidad.
4. **Soporte técnico:** Atención a dudas y problemas reportados por los usuarios del sistema.
5. **Monitoreo y análisis:** Seguimiento del rendimiento del sistema para identificar áreas de mejora o necesidades de optimización.

2.9 FASE DE MUERTE

Objetivo

Retirar de manera adecuada el sistema de administración de la clínica cuando ya no sea funcional o haya sido reemplazado por una solución más moderna.

Actividades

1. **Evaluación de la obsolescencia:**
Determinar si el sistema sigue cumpliendo con las necesidades de la clínica.
2. **Planificación del desmantelamiento:** Definir un cronograma y los recursos necesarios para retirar el sistema de manera ordenada.
3. **Migración de datos:** Transferir la información de pacientes, médicos y citas a una nueva solución tecnológica, asegurando la integridad de los datos.
4. **Desinstalación y eliminación:** Eliminar de manera segura los componentes del sistema en los servidores y equipos de la clínica.
5. **Comunicación y cierre:** Notificar a los usuarios sobre el cierre del sistema y documentar las lecciones aprendidas para futuros proyectos.

Entregables

- Plan de desmantelamiento completo.
- Datos migrados exitosamente a la nueva solución.

- Informe final del cierre del sistema.

CONCLUSIÓN

Durante el desarrollo de este proyecto, mantuvimos una estrecha colaboración con la empresa para comprender a fondo sus procesos. Esto nos permitió realizar un análisis exhaustivo y desarrollar un software que representara de manera precisa estas operaciones. Creemos firmemente que nuestra solución será beneficiosa para la empresa al agilizar procesos, mejorar su eficacia y reducir los tiempos relacionados con la gestión del almacén y la creación de órdenes y requisiciones.

Para la creación del software, optamos por el IDE Apache NetBeans y la base de datos MySQL WorkBench debido a nuestra experiencia previa con estas herramientas. Gracias a la metodología Extreme Programming (XP), logramos avances rápidos y adaptaciones ágiles a las necesidades específicas de la empresa. La automatización implementada ha simplificado el seguimiento del inventario, reducido discrepancias y optimizado el proceso de reabastecimiento.

Este proyecto no solo ha cumplido con los objetivos establecidos, sino que también ha sentado las bases para futuras mejoras y desarrollos en la empresa. La colaboración continua con el equipo permitirá seguir refinando el software para adaptarse a las necesidades cambiantes del negocio. Estamos convencidos de que nuestra solución contribuirá significativamente al éxito y la eficiencia del Estudiante en el futuro.

FUENTES DE INFORMACIÓN

- Beck, K. (2002). *Desarrollo guiado por pruebas: Con ejemplos*. Addison-Wesley.
- Chacon, S., & Straub, B. (2014). *Pro Git* (2^a ed.). Apress.
- Farley, D., & Humble, J. (2010). *Entrega continua: Liberación confiable de software a través de automatización de construcción, pruebas y despliegue*. Addison-Wesley.
- Kim, G., Behr, K., & Spafford, G. (2013). *El proyecto Phoenix: Una novela sobre TI, DevOps y cómo ayudar a tu empresa a ganar*. IT Revolution Press.
- Kruchten, P. (2004). *El proceso unificado de Rational: Una introducción* (3^a ed.). Addison-Wesley.
- Loukides, M. (2011). *Ética en la ingeniería de software*. O'Reilly Media.
- Martin, R. C. (2003). *Desarrollo ágil de software, principios, patrones y prácticas*. Prentice Hall.
- Newman, S. (2015). *Construcción de microservicios: Diseño de sistemas de grano fino*. O'Reilly Media.