

# UNIVERSIDAD AUTÓNOMA DE AGUASCALIENTES

Maestría en ciencias

Fernando Sánchez Ruiz

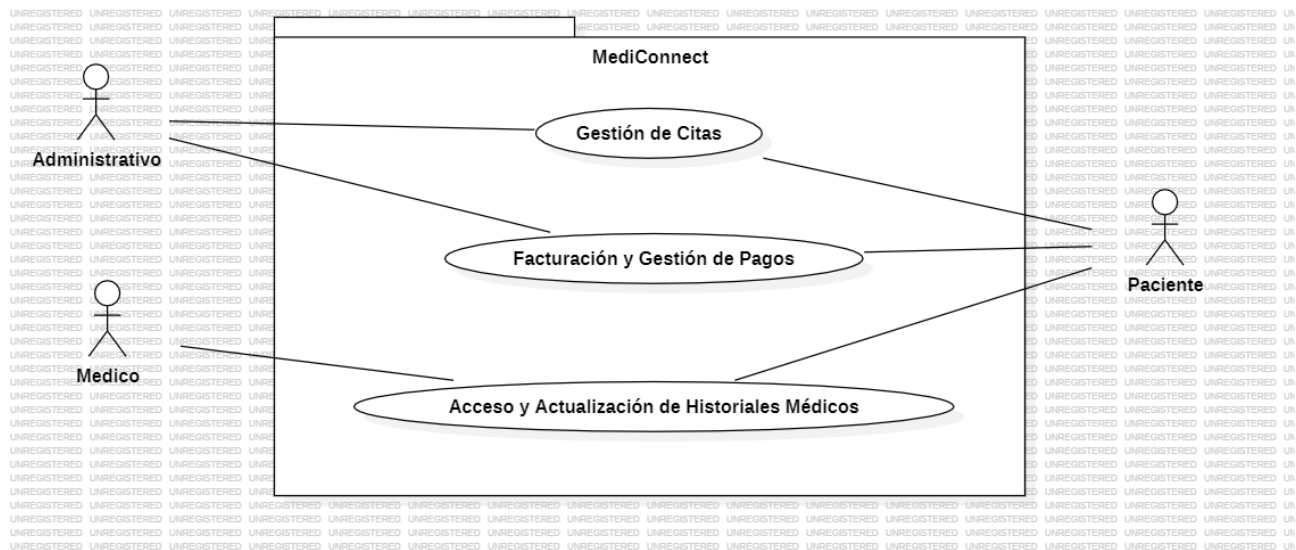
## MediConnect

DESARROLLO DE SOFTWARE

## Planteamiento del Problema

Un equipo de desarrollo de software está encargado de crear una solución integral llamada "MediConnect" para abordar los desafíos de la gestión hospitalaria moderna. Este sistema buscará facilitar la gestión eficiente de citas médicas, permitiendo a los pacientes reservar, modificar o cancelar citas de manera conveniente y ágil, lo cual es esencial para garantizar el acceso oportuno a la atención médica. Además, "MediConnect" proporcionará a los médicos acceso rápido y seguro a los historiales médicos completos de los pacientes, facilitando la toma de decisiones informadas sobre tratamientos y cuidados. Por otro lado, el sistema simplificará y automatizará los procesos de facturación y gestión de pagos para el personal administrativo, eliminando los errores y retrasos asociados con los métodos manuales e ineficientes.

## Diagrama de Casos de uso



## Descripción de casos de uso

### Caso de uso: Gestion de citas

Actor: Paciente

Descripción: Inicia sesión en el sistema para buscar y reservar una cita médica. Puede elegir el especialista, la fecha y la hora según la disponibilidad. Además, el paciente tiene la opción de modificar o cancelar una cita existente.

Actor: Administrativo

Descripción: Recibe notificaciones de nuevas citas, modificaciones o cancelaciones. Puede también ingresar al sistema para ajustar la agenda de los médicos según sea necesario, registrar nuevas citas o modificar las existentes por petición telefónica o presencial de los pacientes.

### **Caso de uso: Facturación y Gestión de Pagos**

Actor: Administrativo

Descripcion: Genera facturas basadas en los servicios médicos proporcionados y las registra en el sistema. Envía estas facturas a los pacientes a través del sistema o por correo electrónico. Además, gestiona los pagos recibidos, actualizando el estado de las facturas en el sistema.

Actor: Paciente

Descripcion: Recibe la factura y realiza el pago a través de la plataforma online. Puede seleccionar diferentes métodos de pago disponibles (tarjeta de crédito, transferencia bancaria, etc.). Una vez realizado el pago, el sistema actualiza su estado de factura a "Pagado".

### **Caso de uso: Acceso y Actualización de Historiales Médicos**

Actor: Médico

Descripcion: Accede al historial médico del paciente para revisar información previa antes de una cita. Durante o después de la consulta, el médico actualiza el historial médico con nuevos diagnósticos, tratamientos o recomendaciones.

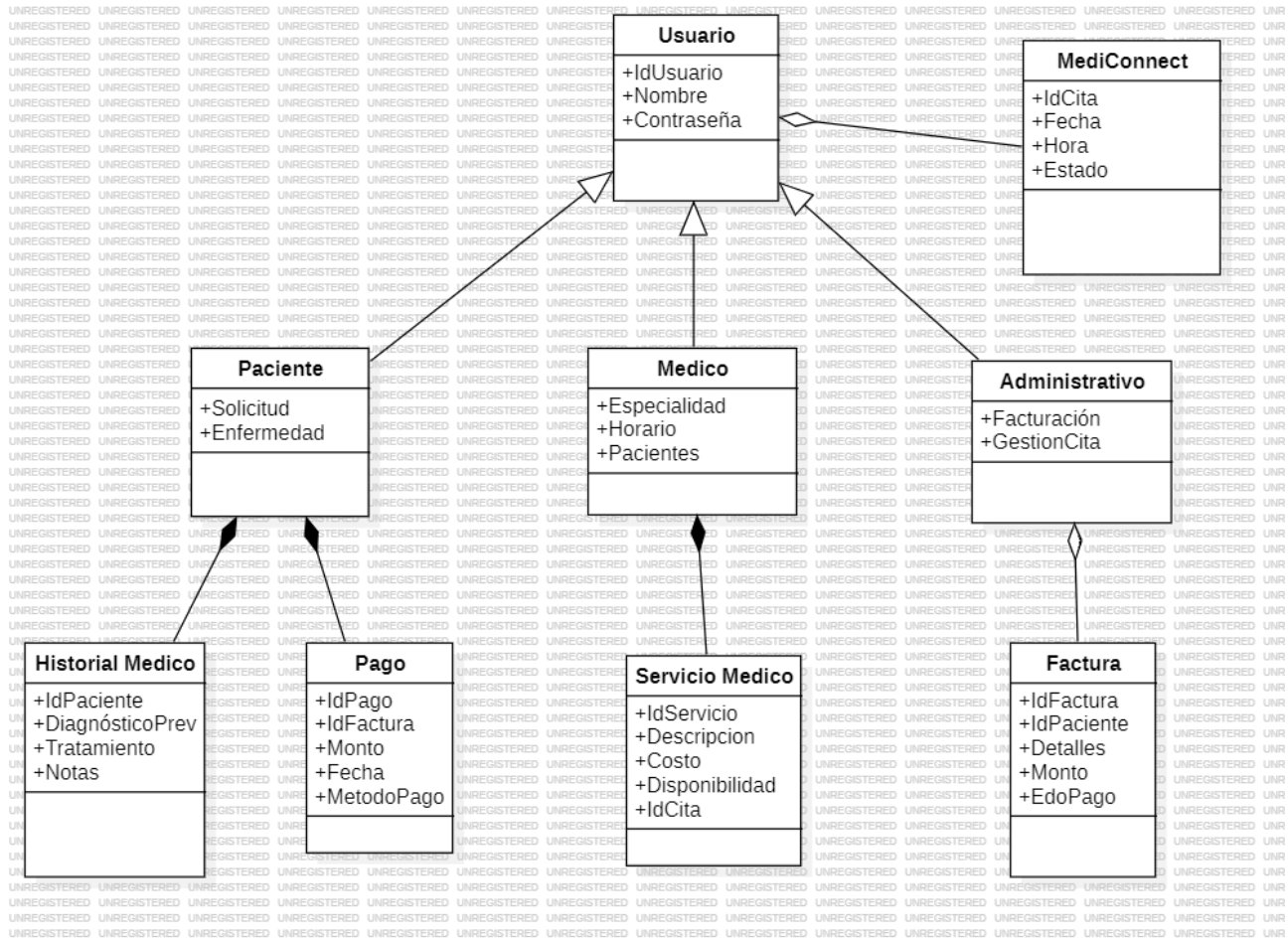
Actor: Paciente

Descripcion: Tiene la capacidad de acceder a su propio historial médico a través del sistema, donde puede ver diagnósticos previos, tratamientos actuales, y recomendaciones médicas. Esto facilita una mayor transparencia y comprensión de su propia salud.

### **Identificación de clases candidatas**

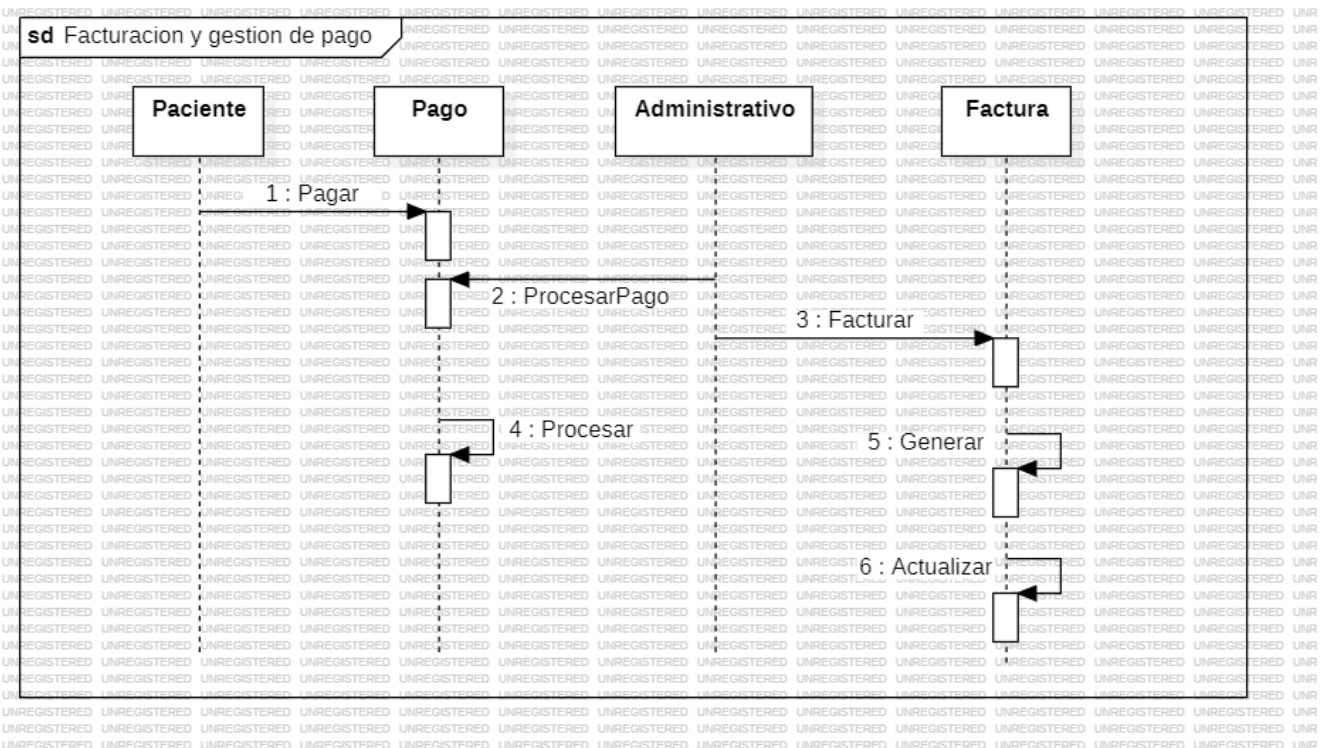
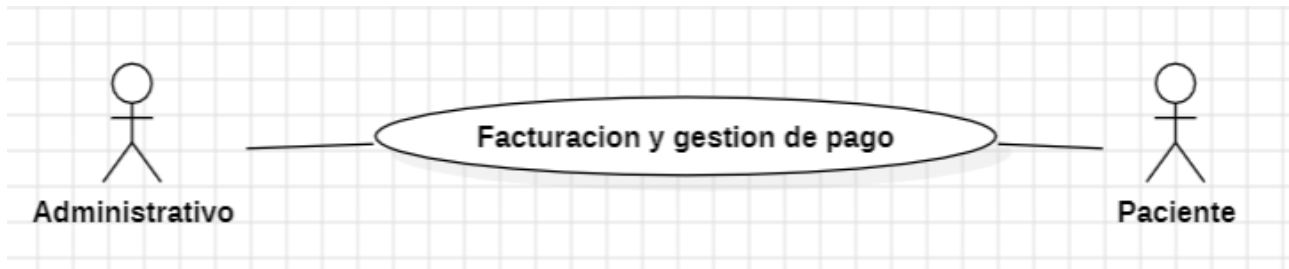
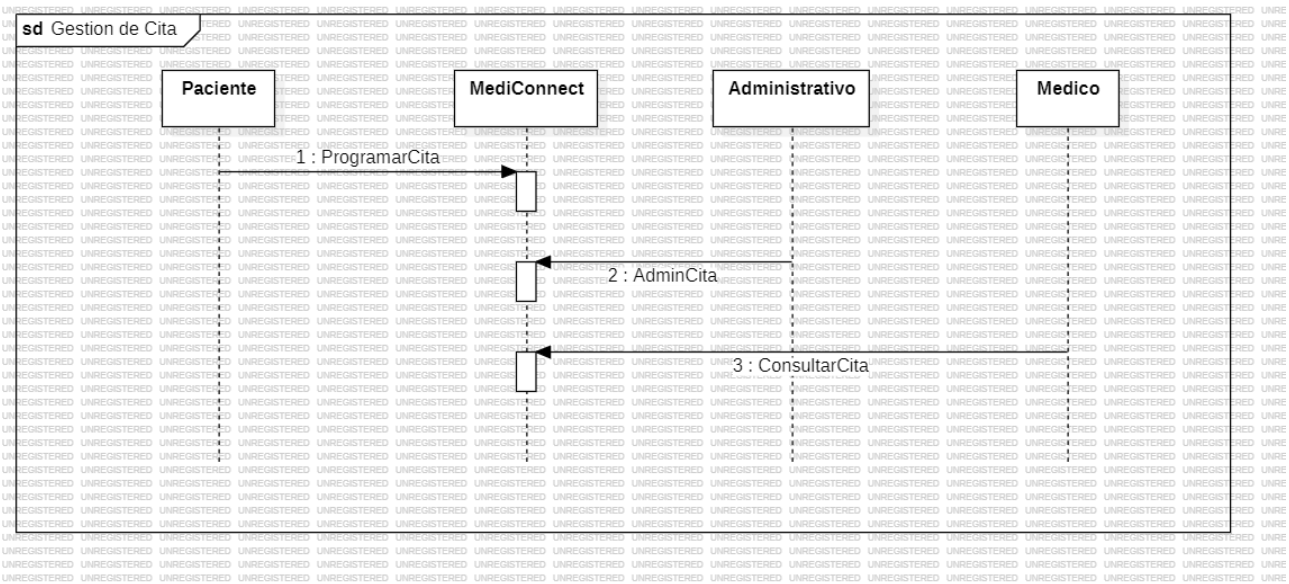
- Usuario
- Paciente
- Medico
- Administrativo
- MediConnect
- Medico
- Historial Medico
- Factura
- Pago
- Servicio Medico
- Calendario

## Modelo Conceptual

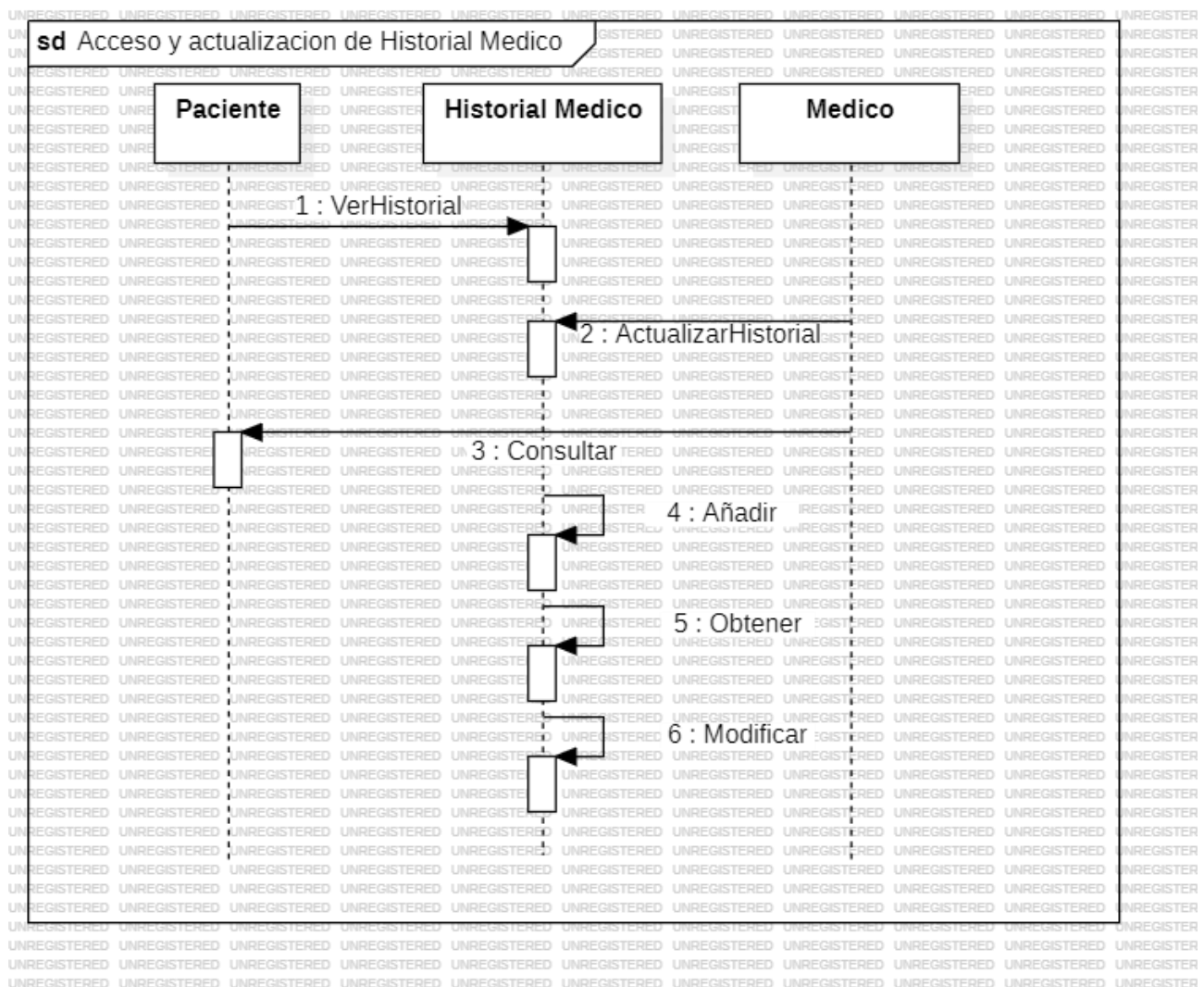
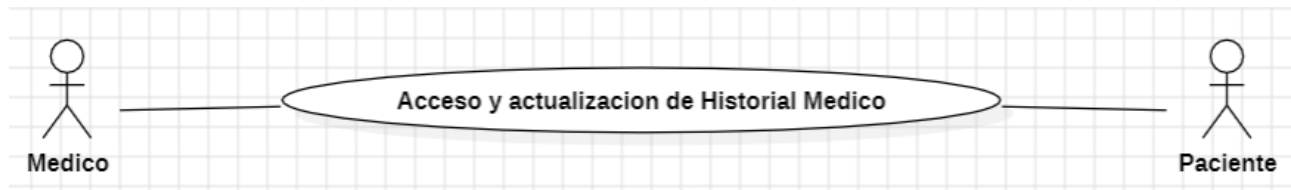


## Diagrama de Uso

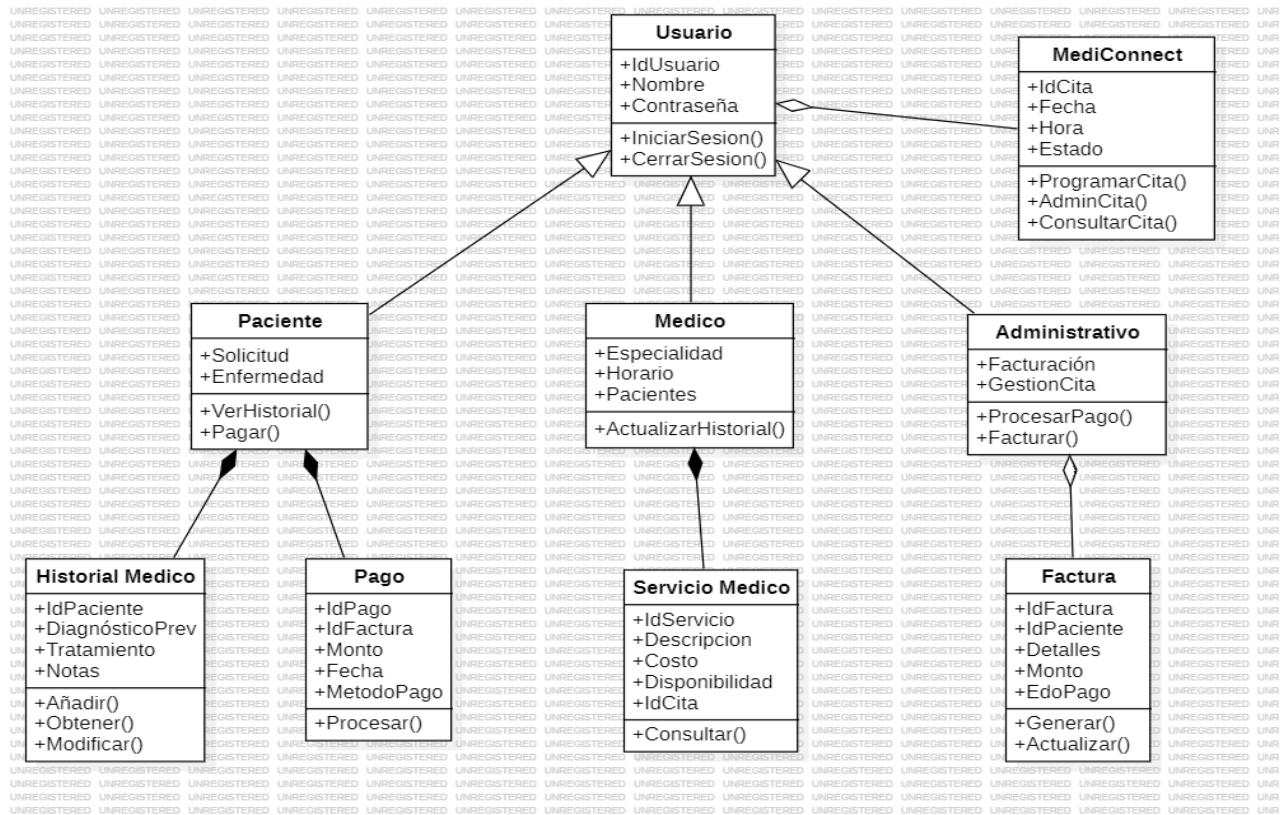




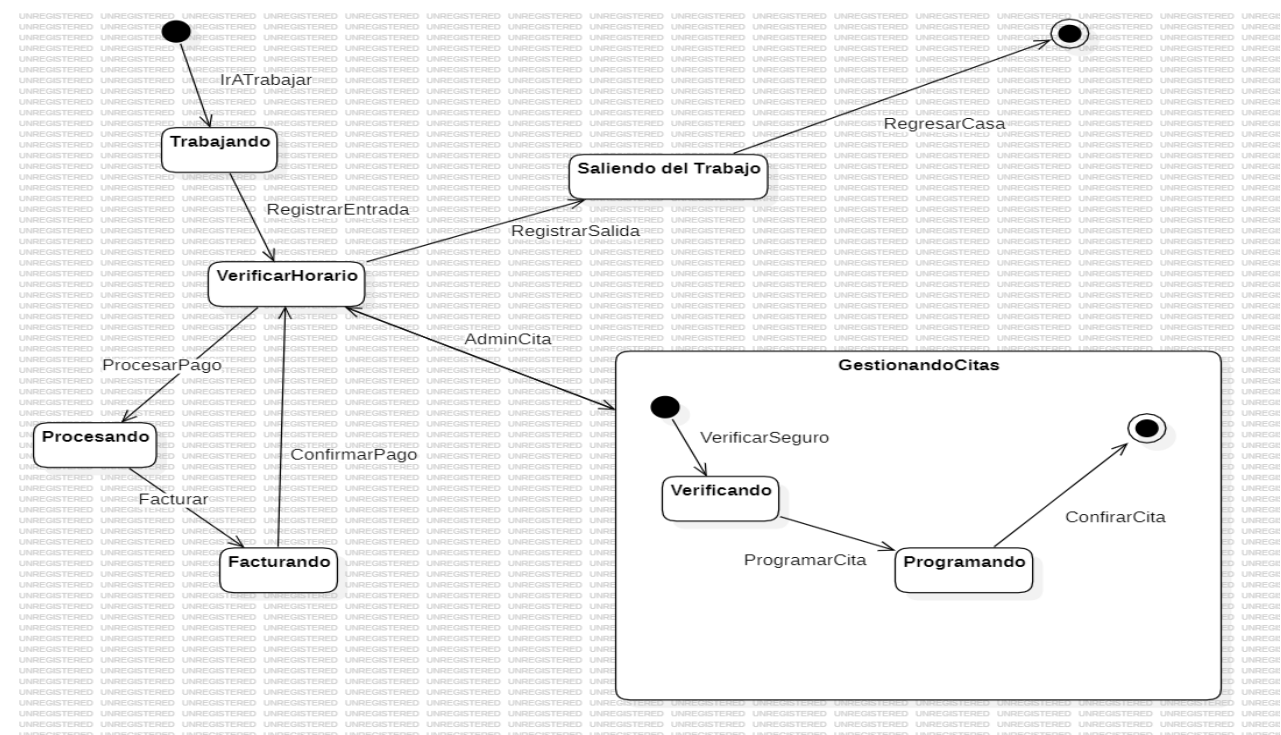




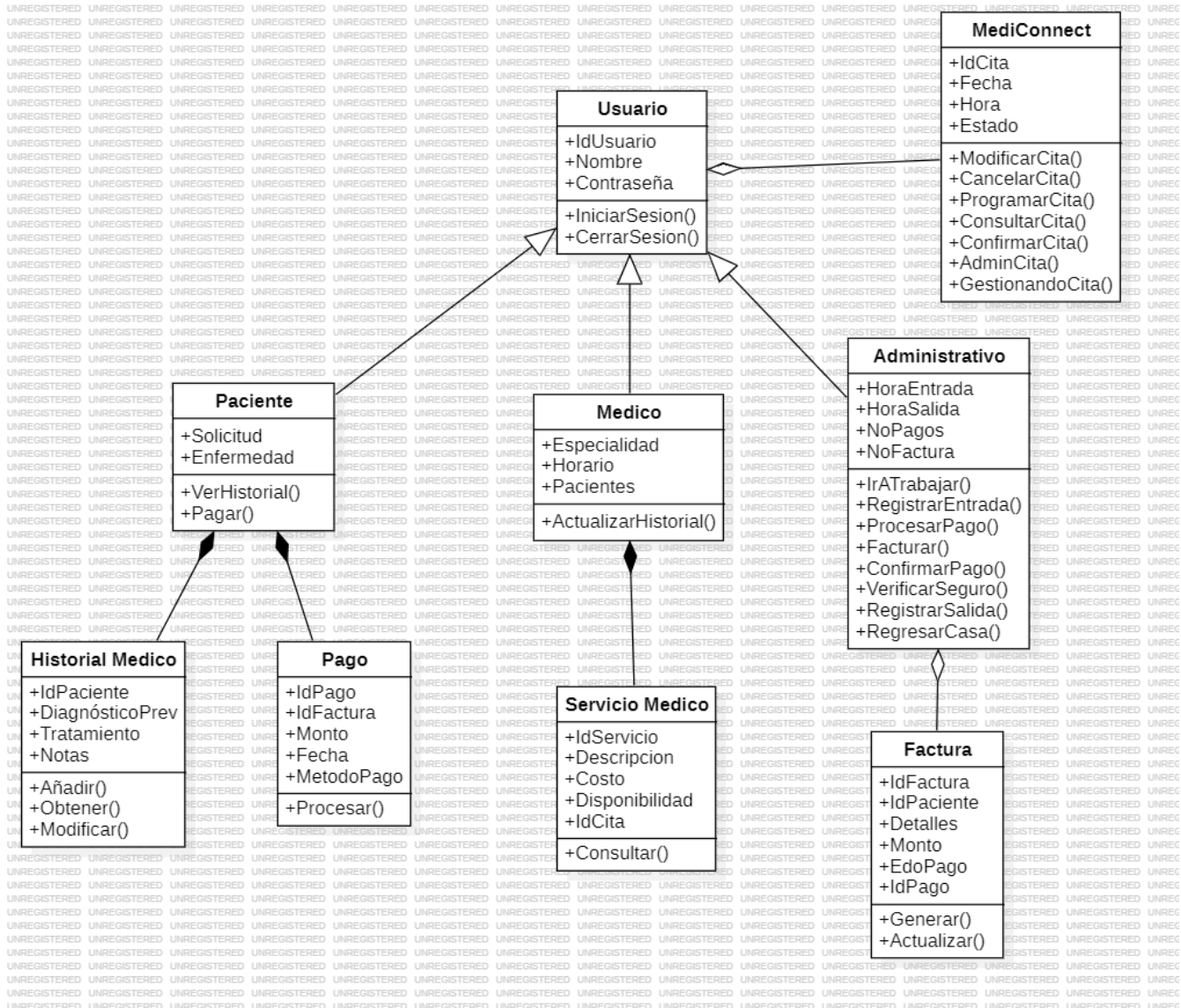
## Diagrama de Clases



## Diagrama de Estados de Administrativo



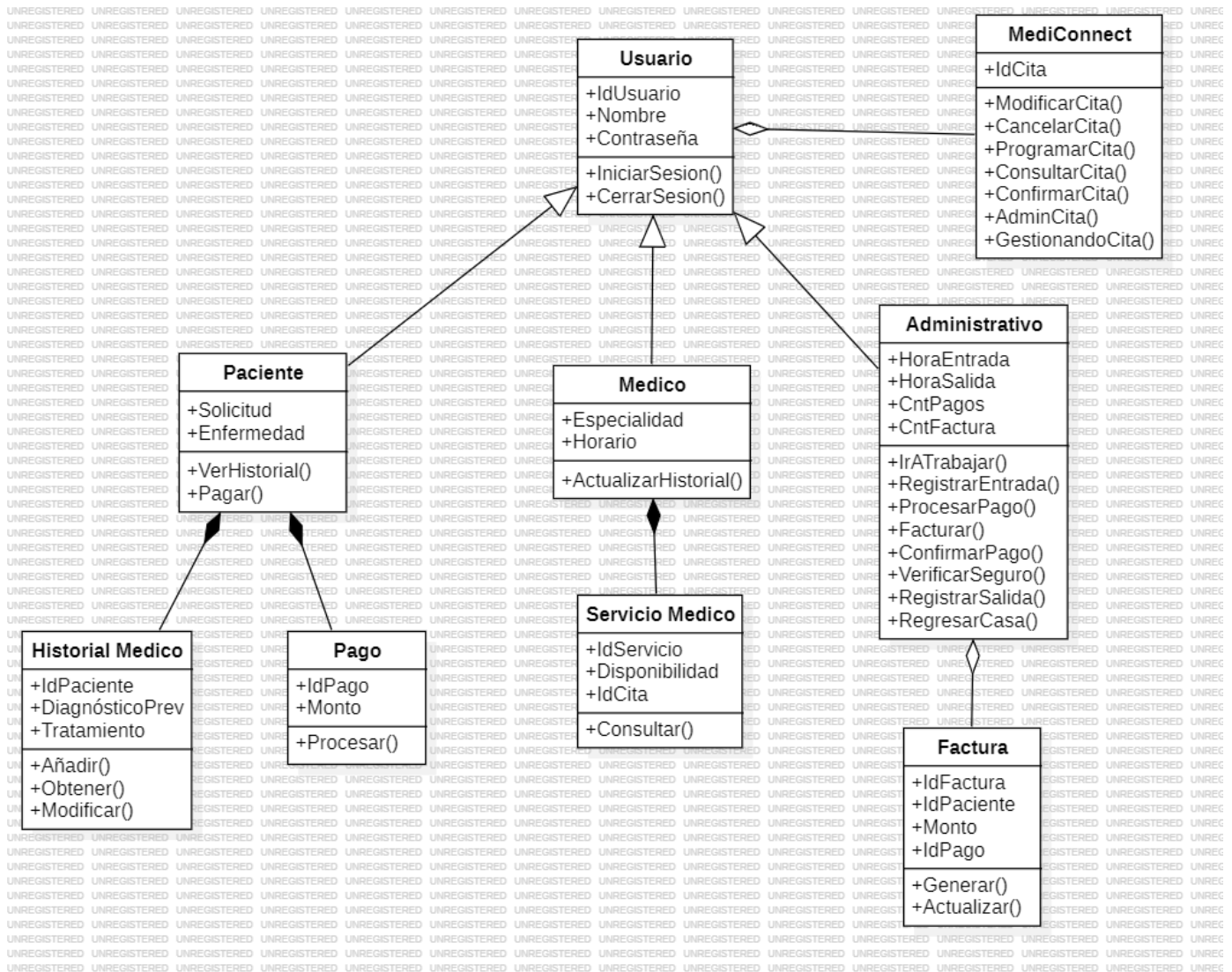
## Diagrama de Clases Modificado



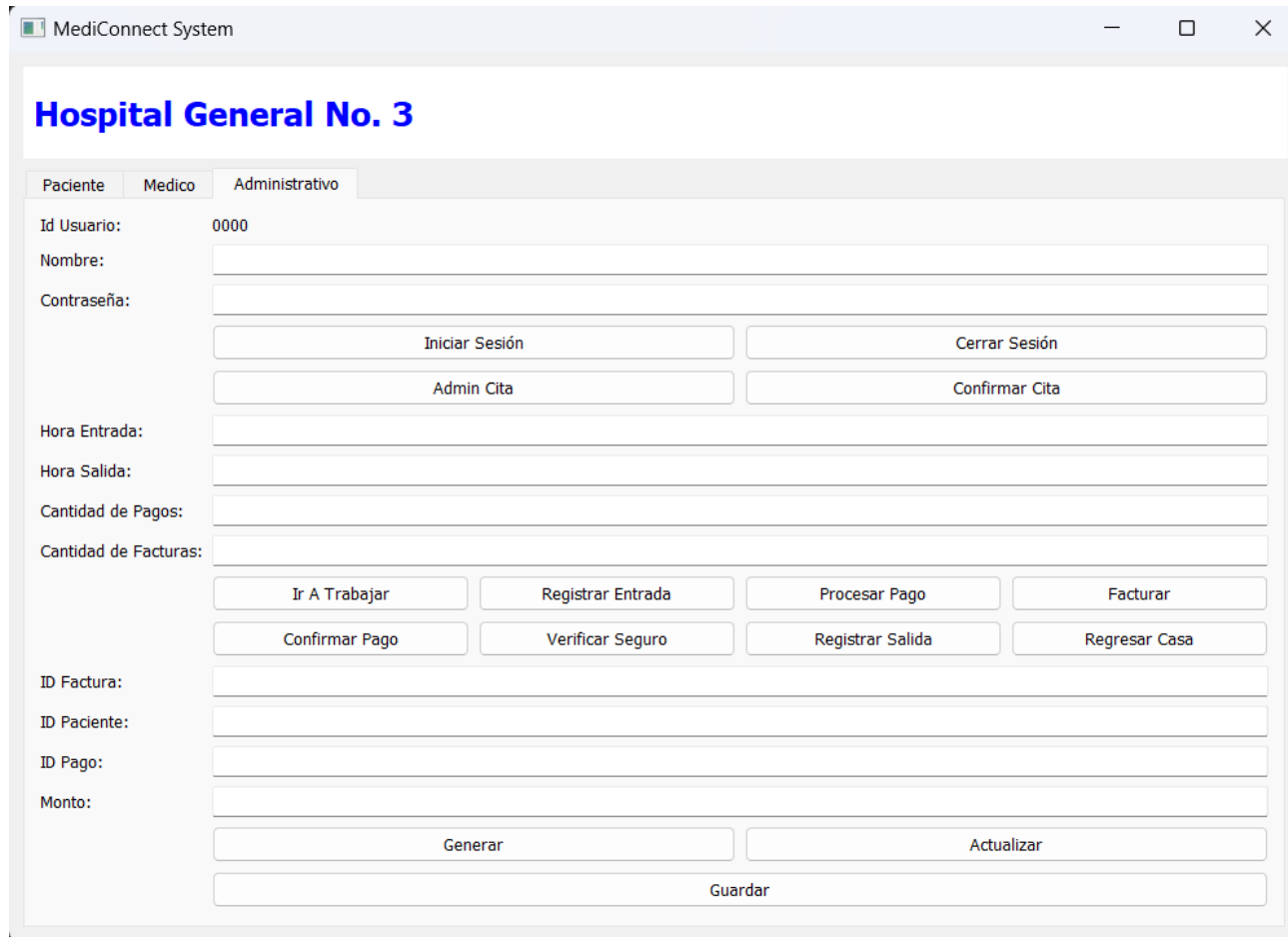


## Diagrama de Clases Simplificado para Ejecución

En la optimización del proyecto, se decidió reducir el número de atributos debido a su extensión, manteniendo así la mayor cantidad de funciones posibles. Este enfoque permitió simplificar la estructura del proyecto sin sacrificar su funcionalidad esencial, asegurando una gestión más eficiente de los recursos y una mejora en la claridad del código.



## GUI (Interfaz de Usuario)



MediConnect System

### Hospital General No. 3

Paciente Medico Administrativo

Id Usuario: 0000

Nombre:

Contraseña:

Iniciar Sesión Cerrar Sesión

Admin Cita Confirmar Cita

Hora Entrada:

Hora Salida:

Cantidad de Pagos:

Cantidad de Facturas:

Ir A Trabajar Registrar Entrada Procesar Pago Facturar

Confirmar Pago Verificar Seguro Registrar Salida Regresar Casa

ID Factura:

ID Paciente:

ID Pago:

Monto:

Generar Actualizar

Guardar

La interfaz del programa se centra en un sistema de gestión hospitalaria denominado "MediConnect", específicamente para "Hospital General No. 3". Ofrece una navegación dividida en tres categorías principales: Paciente, Médico y Administrativo. En la sección visible, correspondiente a Administrativo, se observan campos para ingresar el ID de usuario (preestablecido en '0000'), nombre y contraseña, seguidos de botones para iniciar sesión, administrar citas, confirmar citas y cerrar sesión. Además, incluye áreas para registrar la hora de entrada y salida, la cantidad de pagos y facturas, con botones para realizar diversas operaciones como ir a trabajar, registrar entrada y salida, procesar pagos, facturar, entre otros. Al final, se proporcionan campos para ingresar ID de factura, paciente y pago, junto con un monto, acompañados por botones para generar y actualizar información. El diseño es utilitario y centrado en la funcionalidad, con una estética simple y directa.

Debido a limitaciones de tiempo, no fue posible completar las tres pestañas planificadas y se decidió desarrollar únicamente la pestaña "Administrativo". Esta sección refleja la implementación de métodos específicos del diagrama de clases previamente diseñado para la categoría administrativo del sistema Mediconnect.

MediConnect System

### Hospital General No. 3

PacienteMedicoAdministrativo

Id Usuario:

6815

Nombre:

Fernando

Contraseña:

Perro

Iniciar Sesión

Cerrar Sesión

Admin Cita

Confirmar Cita

Hora Entrada:

12:30

Hora Salida:

6:30

Cantidad de Pagos:

4

Cantidad de Facturas:

15

Ir A Trabajar

Registrar Entrada

Procesar Pago

Facturar

Confirmar Pago

Verificar Seguro

Registrar Salida

Regresar Casa

ID Factura:

1212

ID Paciente:

478

ID Pago:

1234

Monto:

650

Generar

Actualizar

Guardar

En el ejemplo de ejecución del sistema, el usuario debe completar todos los campos requeridos y, tras pulsar el botón de guardar, se asigna automáticamente un ID de usuario aleatorio. Este ID se muestra en la interfaz, lo que a su vez activa los botones correspondientes a las funcionalidades detalladas en el diagrama de clases para la sección "Administrativo". Después presionamos el botón deseado, en nuestro ejemplo presionamos RegistrarEntrada, y se despliega el método seleccionado, en este caso solo es una ventana emergente, sin haber generado una base de datos.

Mensaje de Inicio de Sesión

El trabajador Fernando entro a trabajar a 12:30

OK

Página 11  
de 14

Av. Universidad No. 940, Ciudad Universitaria, C.P. 20131, Aguascalientes, Ags

## Código Fuente Presentable

En esta sección, he de comentar que el programa es muy extenso, entonces dare un ejemplo rápido de un complemento, la herencia de un método grande y un poco del main, sin entrar en el desarrollo de la interfaz.

Factura:

```
class Factura():
    def __init__(self, IdFactura, IdPaciente, IdPago, Monto):
        self.IdFactura = IdFactura
        self.IdPaciente = IdPaciente
        self.Monto = Monto
        self.IdPago = IdPago
    def Generar(self):
        return f"La factura {self.IdFactura} se esta generando\nMonto:
{self.Monto}\nPaciente: {self.IdPaciente}"
    def Actualizar(self):
        return f"La factura {self.IdFactura} ligada al pago {self.IdPago}\nEsta
actualizada"
```

Administrativo:

```
class Administrativo(Usuarios):
    def __init__(self, Nombre, Contraseña, IdCita, #Usuarios
        HoraEntrada, HoraSalida, CntPagos, CntFactura, #Administrativo
        IdFactura, IdPaciente, IdPago, Monto): #Factura
        Usuarios.__init__(self, Nombre, Contraseña, IdCita)
        self.HoraEntrada = HoraEntrada
        self.HoraSalida = HoraSalida
        self.CntPagos = CntPagos
        self.CntFactura = CntFactura
        self.NoFactura = Factura(IdFactura, IdPaciente, IdPago, Monto)
    def __getattr__(self, attr):
        try:
            return getattr(self.mediconnect, attr)
        except AttributeError:
            pass # Si no se encuentra en mediconnect, intenta con ServicioMedico
        try:
            return getattr(self.NoFactura, attr)
        except AttributeError:
            pass
        raise AttributeError(f"'{type(self).__name__}' object has no attribute
'{attr}'")
```



```
def IrAtrabajar(self):
    return f"El trabajador {self.Nombre} esta dirigiendose a trabajar"
def RegistrarEntrada(self):
    return f"El trabajador {self.Nombre} entro a trabajar a {self.HoraEntrada}"
def ProcesarPago (self):
    return f"El pago {self.IdPago} esta en proceso"
def Facturar (self):
    return f"Se han generado {self.CntFactura}"
def ConfirmarPago(self):
    return f"Se han confirmado {self.CntPagos}"
def VerificarSeguro(self):
    return f"El seguro esta activo"
def RegistrarSalida(self):
    return f"El trabajador {self.Nombre} salio del trabajo a {self.HoraSalida}"
def RegresarCasa(self):
    return f"El trabajador {self.Nombre} regreso a casa"
```

Implementación de los métodos en los botones deseados:

```
self.IniciarSesion.clicked.connect(lambda:
self.MostrarMensaje(self.Ad.IniciarSesion(f"Mediconnect.User.{self.IDuser.text()}")
))
    self.CerrarSesion.clicked.connect(lambda:
self.MostrarMensaje(self.Ad.CerrarSesion(f"Mediconnect.User.{self.IDuser.text()}")
)
    self.AdminCita.clicked.connect(lambda:
self.MostrarMensaje(self.Ad.AdminCita()))
    self.ConfirmarCita.clicked.connect(lambda:
self.MostrarMensaje(self.Ad.ConfirmarCita()))

    self.IrAtrabajar.clicked.connect(lambda:
self.MostrarMensaje(self.Ad.IrAtrabajar()))
    self.RegistrarEntrada.clicked.connect(lambda:
self.MostrarMensaje(self.Ad.RegistrarEntrada()))
    self.ProcesarPago.clicked.connect(lambda:
self.MostrarMensaje(self.Ad.ProcesarPago()))
    self.Facturar.clicked.connect(lambda:
self.MostrarMensaje(self.Ad.Facturar()))

    self.ConfirmarPago.clicked.connect(lambda:
self.MostrarMensaje(self.Ad.ConfirmarPago()))
    self.VerificarSeguro.clicked.connect(lambda:
self.MostrarMensaje(self.Ad.VerificarSeguro()))
    self.RegistrarSalida.clicked.connect(lambda:
```

## Diagrama de clases FINAL

El diagrama de clases de la aplicación "MediConnect" ha recibido una actualización final incorporando una clase específica para la interfaz de usuario (MediConnectApp). Esta clase es crucial para la creación de componentes visuales como botones, áreas de texto y etiquetas, que corresponden a los atributos de las clases definidas en el sistema. El sistema se organiza en varias clases principales: Usuario, Paciente, Médico, Administrativo, Servicio Médico, Historial Médico, Pago y Factura.

Cada clase posee atributos y métodos que permiten gestionar las operaciones correspondientes a su rol dentro de la aplicación. Por ejemplo, la clase "Paciente" maneja solicitudes, enfermedades, visualización de historiales médicos y pagos, y algunos métodos provenientes de la clase MediConnect.

