Forma

Descripción generada automáticamente con confianza baja

**Jorge Corvalan Cornejo**

**TALLER DE APLICACIONES JAVA**

Aplicación de Registro de Atenciones Médicas

Actividad Evaluativa E.P.E 2

Contenido

[**Instrucción 1**](#_Toc168886170)

[**Desarrollo 2**](#_Toc168886171)

[**Requerimientos de la Aplicación 2**](#_Toc168886172)

[**Diseño de la Base de Datos 3**](#_Toc168886173)

[**Explicación de las Tablas 3**](#_Toc168886174)

[**Diagrama de la base de datos 4**](#_Toc168886175)

[**Desarrollo de la Aplicación 4**](#_Toc168886176)

[**Estructura del Proyecto 5**](#_Toc168886177)

[**Descripción de las Clases 5**](#_Toc168886178)

[**DAO (Data Access Object) 6**](#_Toc168886179)

[**Models 6**](#_Toc168886180)

[**GUI (Graphical User Interface) 6**](#_Toc168886181)

[**MainFrame 6**](#_Toc168886182)

[**Funcionamiento de la Aplicación 7**](#_Toc168886183)

[**Capturas de pantalla de la Aplicación 7**](#_Toc168886184)

[**Ejemplo de Uso: Panel de Pacientes 10**](#_Toc168886185)

[**Instrucciones para la Ejecución de la Aplicación 12**](#_Toc168886186)

[**Requisitos Previos 12**](#_Toc168886187)

[**Pasos para Ejecutar la Aplicación 12**](#_Toc168886188)

[**Disponibilidad del Código Fuente 12**](#_Toc168886189)

[**Conclusión 13**](#_Toc168886190)

[**Glosario 14**](#_Toc168886191)

[**Bibliografía 15**](#_Toc168886192)

# Instrucción

El presente informe describe el desarrollo de una aplicación Java destinada a una institución de salud, cuyo objetivo principal es registrar y gestionar las atenciones médicas de pacientes en el área de atención cerrada. La aplicación permite realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar) sobre una base de datos MySQL que incluye tablas para médicos, pacientes, especialidades médicas, diagnósticos médicos e ingresos. La interfaz gráfica, desarrollada con Swing, facilita la interacción del usuario con la base de datos, permitiendo una gestión eficiente y organizada de la información.

El desarrollo de esta aplicación se basa en la necesidad de mantener un registro detallado y accesible de las atenciones médicas, consolidando información clave como el paciente atendido, el médico responsable, la especialidad médica, el diagnóstico, y las fechas de ingreso y alta. La estructura modular del proyecto y el uso de patrones de diseño como DAO (Data Access Object) garantizan la mantenibilidad y escalabilidad de la aplicación, permitiendo futuras ampliaciones y adaptaciones a nuevos requerimientos.

# Desarrollo

## Requerimientos de la Aplicación

Para una institución de salud, se requiere:

* Registrar las atenciones de cada paciente en su área de atención cerrada.
* Definir un conjunto de tablas en una Base de Datos relacional MySQL para el registro de médicos, pacientes, especialidades médicas y diagnósticos médicos.
* Consolidar la información en una entidad (Ingresos) que incluya paciente, médico, especialidad médica, diagnóstico, fecha de ingreso y fecha de alta.
* Permitir la administración de los datos mediante una interfaz gráfica que ofrezca funcionalidades para guardar, actualizar y eliminar registros.

## Diseño de la Base de Datos

La base de datos MySQL consta de las siguientes tablas:

1. CREATE DATABASE IF NOT EXISTS InstitucionSalud;

2.

3. USE InstitucionSalud;

4.

5. CREATE TABLE IF NOT EXISTS Especialidades (

6. id INT AUTO\_INCREMENT PRIMARY KEY,

7. nombre VARCHAR(100) NOT NULL

8. );

9.

10. CREATE TABLE IF NOT EXISTS Medicos (

11. id INT AUTO\_INCREMENT PRIMARY KEY,

12. nombre VARCHAR(100) NOT NULL,

13. especialidad\_id INT,

14. FOREIGN KEY (especialidad\_id) REFERENCES Especialidades(id)

15. );

16.

17. CREATE TABLE IF NOT EXISTS Pacientes (

18. id INT AUTO\_INCREMENT PRIMARY KEY,

19. nombre VARCHAR(100) NOT NULL,

20. fecha\_nacimiento DATE NOT NULL

21. );

22.

23. CREATE TABLE IF NOT EXISTS Diagnosticos (

24. id INT AUTO\_INCREMENT PRIMARY KEY,

25. nombre VARCHAR(100) NOT NULL

26. );

27.

28. CREATE TABLE IF NOT EXISTS Ingresos (

29. id INT AUTO\_INCREMENT PRIMARY KEY,

30. paciente\_id INT,

31. medico\_id INT,

32. especialidad\_id INT,

33. diagnostico\_id INT,

34. fecha\_ingreso DATE,

35. fecha\_alta DATE,

36. FOREIGN KEY (paciente\_id) REFERENCES Pacientes(id),

37. FOREIGN KEY (medico\_id) REFERENCES Medicos(id),

38. FOREIGN KEY (especialidad\_id) REFERENCES Especialidades(id),

39. FOREIGN KEY (diagnostico\_id) REFERENCES Diagnosticos(id)

40. );

41.

42.

### Explicación de las Tablas

* **Especialidades**: Contiene la información de las especialidades médicas disponibles.
* **Medicos**: Registra los datos de los médicos, incluyendo su especialidad.
* **Pacientes**: Almacena la información de los pacientes.
* **Diagnosticos**: Guarda los distintos diagnósticos médicos.
* **Ingresos**: Consolida la información de las atenciones médicas, vinculando pacientes, médicos, especialidades y diagnósticos con las fechas de ingreso y alta.

### Diagrama de la base de datos

Diagrama

Descripción generada automáticamente

## Desarrollo de la Aplicación

El desarrollo de la aplicación Java se realizó utilizando las siguientes tecnologías y bibliotecas:

* **Java SE 8**: Para la programación en Java.
* **Swing**: Para la creación de la interfaz gráfica de usuario.
* **JDBC**: Para la conexión y manipulación de la base de datos MySQL.

## Estructura del Proyecto

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamenteEl proyecto está organizado en los siguientes paquetes y clases:

## Descripción de las Clases

### DAO (Data Access Object)

Cada clase DAO contiene métodos para realizar operaciones CRUD sobre las tablas de la base de datos correspondientes. Por ejemplo, MedicoDAO tiene métodos como getAll(), insert(), update() y delete() para manejar los registros de médicos.

### Models

Las clases en el paquete models representan las entidades de la base de datos. Cada clase tiene atributos que corresponden a las columnas de las tablas y métodos getter y setter para acceder y modificar estos atributos.

### GUI (Graphical User Interface)

Cada clase en el paquete gui representa un panel que permite la interacción con una tabla específica de la base de datos. Los paneles incluyen formularios para ingresar datos y botones para agregar, actualizar y eliminar registros. También incluyen una tabla que muestra los datos actuales de la base de datos.

### MainFrame

MainFrame es la clase principal que crea la ventana principal de la aplicación. Contiene una barra de menús que permite al usuario seleccionar diferentes paneles para trabajar con las distintas tablas de la base de datos.

## Funcionamiento de la Aplicación

El usuario puede seleccionar diferentes tablas de la base de datos a través de la barra de menús. Cada panel permite al usuario:

* **Agregar**: Ingresar nuevos registros a la base de datos.
* **Actualizar**: Modificar registros existentes.
* **Eliminar**: Borrar registros seleccionados.
* **Ver**: Visualizar los registros actuales en una tabla.

### Capturas de pantalla de la Aplicación

* Interfaz de usuario gráfica, Aplicación

  Descripción generada automáticamentePanel de Especialidades
* **Tabla

  Descripción generada automáticamentePanel de Médicos**
* **Tabla

  Descripción generada automáticamentePanel de Pacientes**
* **Interfaz de usuario gráfica, Aplicación

  Descripción generada automáticamentePanel de Diagnósticos**
* **Tabla

  Descripción generada automáticamentePanel de Ingresos**

### Ejemplo de Uso: Panel de Pacientes

* Tabla

  Descripción generada automáticamente**Agregar Paciente**: El usuario ingresa el nombre y la fecha de nacimiento del paciente en los campos correspondientes y presiona el botón "Agregar".
* Tabla

  Descripción generada automáticamente**Actualizar Paciente**: El usuario selecciona un paciente de la tabla, modifica los campos y presiona "Actualizar".
* Tabla

  Descripción generada automáticamenteTabla

  Descripción generada automáticamente**Eliminar Paciente**: El usuario selecciona un paciente de la tabla y presiona "Eliminar".

## Instrucciones para la Ejecución de la Aplicación

### Requisitos Previos

* Tener instalado Java SE Development Kit (JDK).
* Tener instalado MySQL y crear la base de datos utilizando el script SQL proporcionado.

### Pasos para Ejecutar la Aplicación

1. **Configurar la Base de Datos**: Ejecutar el script SQL para crear las tablas y la base de datos.
2. **Configurar el Proyecto**: Importar el proyecto Java en el IDE de su preferencia.
3. **Configurar la Conexión a la Base de Datos**: Modificar los parámetros de conexión en las clases DAO para que apunten a su instancia de MySQL.
4. **Compilar y Ejecutar la Aplicación**: Compilar y ejecutar la aplicación desde el IDE.

### Disponibilidad del Código Fuente

El código fuente completo de esta aplicación está disponible en GitHub. Puede ser accedido y clonado desde el siguiente repositorio:

[GitHub - Institución Salud](https://github.com/CookeV/institucionsalud)

En este repositorio también se encuentran los archivos necesarios para crear la base de datos, así como una copia de este informe. Esto permite al docente y a cualquier otra persona interesada examinar el código, realizar pruebas y verificar la implementación de la aplicación de manera detallada.

# Conclusión

La aplicación desarrollada cumple con los objetivos planteados, proporcionando una herramienta robusta y fácil de usar para la gestión de atenciones médicas en una institución de salud. La interfaz gráfica intuitiva y las funcionalidades CRUD aseguran un manejo eficiente de los datos, permitiendo a los usuarios registrar, actualizar y consultar información de manera ágil. La implementación de una estructura modular y el uso de patrones de diseño como DAO (Data Access Object) facilitan el mantenimiento y la escalabilidad del sistema, permitiendo futuras ampliaciones y adaptaciones a nuevos requerimientos.

El proceso de desarrollo de esta aplicación ha sido una experiencia de aprendizaje significativa. Se adquirieron conocimientos sobre la importancia de un diseño de base de datos bien estructurado y cómo manejar eficientemente las relaciones entre tablas en una aplicación real. Además, la integración de la lógica de la aplicación con una interfaz gráfica de usuario proporcionó una comprensión valiosa sobre el diseño de interfaces intuitivas que mejoren la experiencia del usuario. La experiencia también reforzó la importancia de escribir código modular y mantenible, esencial para cualquier proyecto de software escalable. Estos aprendizajes son fundamentales para futuros proyectos y para el crecimiento profesional en el campo del desarrollo de software.

# Glosario

|  |  |
| --- | --- |
| Término | Definición |
| CRUD | Acrónimo de Crear, Leer, Actualizar y Eliminar, que describe las operaciones básicas de una base de datos. |
| DAO (Data Access Object) | Patrón de diseño que proporciona una abstracción de acceso a una base de datos u otro mecanismo de persistencia. |
| JDBC (Java Database Connectivity) | API en Java que define como un cliente puede acceder a una base de datos. |
| MySQL | Sistema de gestión de bases de datos relacional basado en SQL. |
| Swing | Biblioteca de componentes de interfaz gráfica para Java que permite la creación de aplicaciones con GUI. |
| IDE (Integrated Development Environment) | Entorno de desarrollo integrado que proporciona herramientas para el desarrollo de software, como edición de código, depuración y compilación. |
| Base de Datos Relacional | Sistema de base de datos que organiza los datos en tablas relacionadas entre sí mediante llaves primarias y foráneas. |
| Interfaz Gráfica de Usuario (GUI) | Interfaz que permite a los usuarios interactuar con una aplicación a través de componentes gráficos como botones, formularios y tablas. |
| Script SQL | Conjunto de instrucciones SQL que se ejecutan en una base de datos para realizar tareas como creación de tablas o inserción de datos. |
| Modularidad | Principio de diseño que divide un sistema en partes más pequeñas y manejables, facilitando el mantenimiento y escalabilidad del software. |
| Escalabilidad | Capacidad de un sistema de manejar un incremento en la carga de trabajo o la ampliación de sus funcionalidades sin comprometer su desempeño. |
| GitHub | Plataforma de desarrollo colaborativo y control de versiones basada en Git, utilizada para alojar y compartir proyectos de software. |
| Repositorio | Espacio de almacenamiento centralizado donde se guardan y gestionan los archivos de un proyecto de software. |
| Compilación | Proceso de traducir el código fuente escrito en un lenguaje de programación a un formato ejecutable por una computadora. |

# Bibliografía

* **Oracle. JDBC Technology. Recuperado de** [**https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/**](https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/)
* **MySQL. MySQL :: MySQL 8.0 Reference Manual. Recuperado de** [**https://dev.mysql.com/doc/refman/8.0/en/**](https://dev.mysql.com/doc/refman/8.0/en/)
* **Oracle. The Java™ Tutorials. Recuperado de** [**https://docs.oracle.com/javase/tutorial/**](https://docs.oracle.com/javase/tutorial/)
* **Oracle. Java Platform, Standard Edition 8 API Specification. Recuperado de** [**https://docs.oracle.com/javase/8/docs/api/**](https://docs.oracle.com/javase/8/docs/api/)
* **GitHub. About GitHub. Recuperado de** [**https://github.com/about**](https://github.com/about)
* **Stack Overflow. Stack Overflow - Where Developers Learn, Share, & Build Careers. Recuperado de** [**https://stackoverflow.com/**](https://stackoverflow.com/)