Validaciones

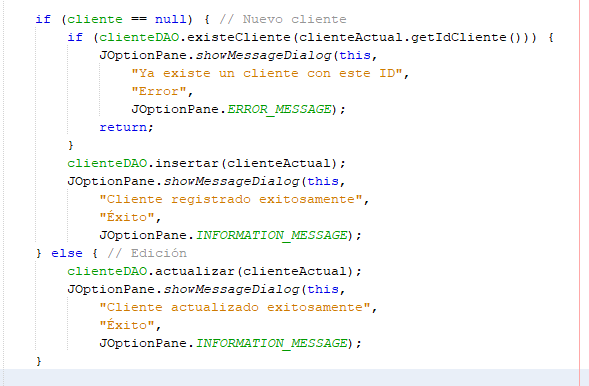
Interfaz de usuario gráfica, Sitio web

El contenido generado por IA puede ser incorrecto.

Este método es crucial para garantizar que los datos ingresados en el formulario de películas sean válidos antes de guardarlos en la base de datos.

Texto

El contenido generado por IA puede ser incorrecto.

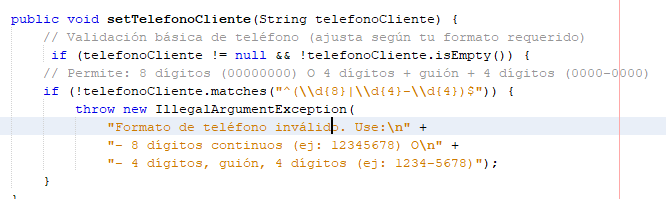
Este método realiza validaciones esenciales para el formulario de clientes, asegurando que los datos ingresados cumplan con los requisitos básicos antes de ser procesados.

**es una validación de existencia previa** que actúa como **guardia de integridad de datos** antes de realizar operaciones en la base de datos.

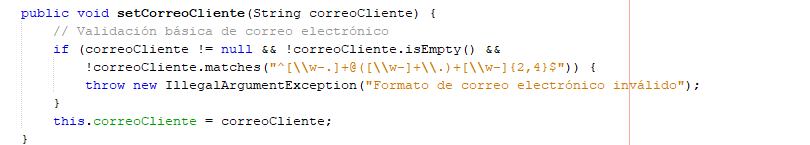
Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.

verifica si el parámetro id no es null y cumple con un formato específico: debe ser una cadena compuesta exactamente por 13 dígitos numéricos. La expresión regular "^\\d{13}$" asegura que el contenido tenga solo números (\\d), exactamente 13 caracteres ({13}), desde el inicio (^) hasta el final ($) de la cadena.



valida que el número de teléfono ingresado tenga un formato aceptado antes de establecerlo. Primero, verifica que el valor no sea null ni esté vacío. Luego, usa una expresión regular para asegurarse de que el teléfono tenga **exactamente 8 dígitos seguidos** (12345678) o **4 dígitos, un guión y otros 4 dígitos** (1234-5678). Si no cumple con alguno de estos dos formatos, lanza una excepción IllegalArgumentException con un mensaje claro para el usuario sobre los formatos válidos.



valida que el correo electrónico proporcionado tenga un formato correcto antes de asignarlo. Primero, verifica que el valor no sea null ni esté vacío. Luego, usa una expresión regular para confirmar que el correo tenga una estructura válida: una combinación de letras, números o guiones antes del símbolo @, seguido por un dominio con uno o más subdominios separados por puntos y una extensión final de 2 a 4 caracteres.

Explicacion del DAO

El patrón **DAO (Data Access Object)** junto con **JDBC** y una estructura **MVC** organiza el código en capas claras: los **models** definen entidades como Cliente con atributos y getters/setters; el **dao** contiene interfaces (como ClienteDAO) que declaran operaciones CRUD, mientras que **dao/impl** implementa esas interfaces usando JDBC, gestionando conexiones (Connection), consultas parametrizadas (PreparedStatement) y mapeo de resultados (ResultSet). Los **controllers** coordinan la lógica de negocio, validando datos y mediando entre las **views** (interfaz gráfica) y el DAO, asegurando que la vista nunca interactúe directamente con la base de datos. Esta arquitectura garantiza modularidad, seguridad contra SQL injection (gracias a PreparedStatement), y facilidad de mantenimiento, ya que cambios en la BD solo afectan al DAO, sin impactar el resto del sistema.