**Práctica de Aula sobre Pruebas**

**Alumno: Pelayo Iglesias Manzano**

**UO266600 – PL 3 – Equipo 4**

**Índice**

[**1.Referencia al proyecto y equipo de prácticas de laboratorio** 3](#_Toc197432338)

[**2.Descripción de los procesos de negocio** 3](#_Toc197432339)

[**3. Identificación y descripción de los métodos** 3](#_Toc197432340)

[**4.Diseño de las pruebas** 5](#_Toc197432341)

[**5.Código fuente** 6](#_Toc197432342)

[**6.Pantallazo del resultado en JUNIT** 11](#_Toc197432343)

# **1.Referencia al proyecto y equipo de prácticas de laboratorio**

**Proyecto:** Sistema de gestión COIIPA.

**Grupo:** 3\_4 (PL3, equipo 4).

# **2.Descripción de los procesos de negocio**

El proceso de negocio probado en este estudio corresponde a la actualización del estado de una solicitud pericial. Un perito asignado a una pericial puede realizar una de las siguientes acciones:

* Aceptar la pericial: implica el cambio de estado a “Aceptado”.
* Rechazar la pericial: implica el cambio de estado a “Rechazado”, el registro del motivo y la fecha de rechazo.
* Cancelar la pericial: implica el cambio de estado a “Cancelado”, el registro del motivo y la fecha de cancelación.

No se aceptan estados diferentes a “Aceptado”, “Rechazado” o “Cancelado”.

El sistema de gestión de periciales solo permite cambios desde el estado “Asignado”, una pericial rechazada o cancelada no puede ser modificada.

Al rechazar o cancelar una pericial se debe indicar un motivo que no esté vacío.

# **3. Identificación y descripción de los métodos**

El método puesto a prueba es:  
*public boolean actualizarEstado(String numeroReferencia, String nuevoEstado, String motivo, String fecha)*

Este método está en la clase ActualizarPericialModel, del paquete si2024\_pl34.historiaActualizarPericial.

actualizarEstado incluye en su lógica:

* Verificar si el estado actual de la pericial es “Asignado”.
* Comprobar si el nuevo estado es válido.
* Si el estado cambia a “Rechazado” o “Cancelado” se valida que el motivo no este vacío.
* Actualizar los campos motivo\_rechazo y fecha\_rechazo para “Rechazado” y “Cancelado”.
* Retorna true si la operación tuvo éxito o false si no cumple las condiciones.

Código fuente del método:  
*public boolean actualizarEstado(String numeroReferencia, String nuevoEstado, String motivo, String fecha) {*

*String estadoActual = obtenerEstadoActual(numeroReferencia);*

*// Validar transición de estado*

*if (!estadoActual.equals("Asignado") ||*

*(!nuevoEstado.equals("Aceptado") && !nuevoEstado.equals("Rechazado") && !nuevoEstado.equals("Cancelado"))) {*

*return false;*

*}*

*// Validar motivo si es Rechazado o Cancelado*

*if ((nuevoEstado.equals("Rechazado") || nuevoEstado.equals("Cancelado")) &&*

*(motivo == null || motivo.trim().isEmpty())) {*

*return false;*

*}*

*String sql = "UPDATE peritaje SET estado = ?" +*

*(nuevoEstado.equals("Rechazado") || nuevoEstado.equals("Cancelado")*

*? ", motivo\_rechazo = ?, fecha\_rechazo = ?"*

*: "") +*

*" WHERE numero\_referencia = ?";*

*try (Connection conn = db.getConnection();*

*PreparedStatement stmt = conn.prepareStatement(sql)) {*

*stmt.setString(1, nuevoEstado);*

*if (nuevoEstado.equals("Rechazado") || nuevoEstado.equals("Cancelado")) {*

*stmt.setString(2, motivo);*

*stmt.setString(3, fecha);*

*stmt.setString(4, numeroReferencia);*

*} else {*

*stmt.setString(2, numeroReferencia);*

*}*

*return stmt.executeUpdate() > 0;*

*} catch (Exception e) {*

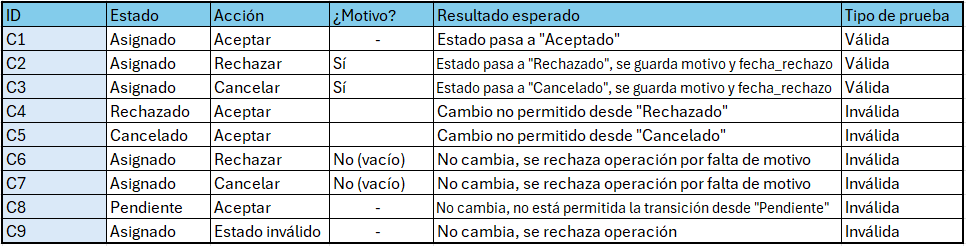
*System.****err****.println("Error al actualizar el estado del peritaje: " + e.getMessage());*

*return false;*

*}*

*}*

# **4.Diseño de las pruebas**

A continuación, se mostrará una tabla con las clases de equivalencia que se han empleado:

# **5.Código fuente**

Código fuente de TestActualizarPericial.java en la rama Pruebas\_UO266600:

*package si2024\_pl34.jdbc.ut;*

*import static org.junit.Assert.\*;*

*import org.junit.After;*

*import org.junit.Before;*

*import org.junit.Test;*

*import si2024\_pl34.historiaActualizarPericial.ActualizarPericialDTO;*

*import si2024\_pl34.historiaActualizarPericial.ActualizarPericialModel;*

*import si2024\_pl34.util.Database;*

*public class TestActualizarPericial {*

*private static Database db = new Database();*

*private ActualizarPericialModel model;*

*@Before*

*public void setUp() {*

*db.createDatabase(true);*

*cargarCasosPrueba();*

*model = new ActualizarPericialModel(db);*

*}*

*@After*

*public void tearDown() {*

*}*

*private void cargarCasosPrueba() {*

*db.executeBatch(new String[]{*

*"DELETE FROM peritaje",*

*"INSERT INTO peritaje (numero\_referencia, estado, fecha\_solicitud) VALUES ('350', 'Asignado', '2025-05-01')", // C1*

*"INSERT INTO peritaje (numero\_referencia, estado, fecha\_solicitud) VALUES ('351', 'Asignado', '2025-05-01')", // C2*

*"INSERT INTO peritaje (numero\_referencia, estado, fecha\_solicitud) VALUES ('352', 'Asignado', '2025-05-01')", // C3*

*"INSERT INTO peritaje (numero\_referencia, estado, fecha\_solicitud) VALUES ('353', 'Rechazado', '2025-05-01')", // C4*

*"INSERT INTO peritaje (numero\_referencia, estado, fecha\_solicitud) VALUES ('354', 'Cancelado', '2025-05-01')", // C5*

*"INSERT INTO peritaje (numero\_referencia, estado, fecha\_solicitud) VALUES ('355', 'Asignado', '2025-05-01')", // C6*

*"INSERT INTO peritaje (numero\_referencia, estado, fecha\_solicitud) VALUES ('356', 'Asignado', '2025-05-01')", // C7*

*"INSERT INTO peritaje (numero\_referencia, estado, fecha\_solicitud) VALUES ('357', 'Pendiente', '2025-05-01')", // C8*

*"INSERT INTO peritaje (numero\_referencia, estado, fecha\_solicitud) VALUES ('358', 'Asignado', '2025-05-01')" // C9*

*});*

*}*

*// C1: Asignado ==> Aceptado*

*@Test public void testC1() {*

*boolean ok = model.actualizarEstado("350", "Aceptado", null, "2025-06-01");*

*assertTrue(ok);*

*ActualizarPericialDTO dto = model.obtenerPericialPorReferencia("350");*

*assertEquals("Aceptado", dto.getEstado());*

*}*

*// C2: Asignado ==> Rechazado (con motivo)*

*@Test public void testC2() {*

*boolean ok = model.actualizarEstado("351", "Rechazado", "Falta de disponibilidad", "2025-06-02");*

*assertTrue(ok);*

*ActualizarPericialDTO dto = model.obtenerPericialPorReferencia("351");*

*assertEquals("Rechazado", dto.getEstado());*

*assertEquals("Falta de disponibilidad", dto.getMotivoRechazo());*

*assertEquals("2025-06-02", dto.getFechaRechazo());*

*}*

*// C3: Asignado ==> Cancelado (con motivo)*

*@Test public void testC3() {*

*boolean ok = model.actualizarEstado("352", "Cancelado", "No se llegó a acuerdo", "2025-06-03");*

*assertTrue(ok);*

*ActualizarPericialDTO dto = model.obtenerPericialPorReferencia("352");*

*assertEquals("Cancelado", dto.getEstado());*

*assertEquals("No se llegó a acuerdo", dto.getMotivoRechazo());*

*assertEquals("2025-06-03", dto.getFechaRechazo());*

*}*

*// C4: Rechazado ==> Aceptado (inválido)*

*@Test public void testC4() {*

*boolean ok = model.actualizarEstado("353", "Aceptado", null, "2025-06-04");*

*assertFalse(ok);*

*ActualizarPericialDTO dto = model.obtenerPericialPorReferencia("353");*

*assertEquals("Rechazado", dto.getEstado());*

*}*

*// C5: Cancelado ==> Aceptado (inválido)*

*@Test public void testC5() {*

*boolean ok = model.actualizarEstado("354", "Aceptado", null, "2025-06-05");*

*assertFalse(ok);*

*ActualizarPericialDTO dto = model.obtenerPericialPorReferencia("354");*

*assertEquals("Cancelado", dto.getEstado());*

*}*

*// C6: Asignado ==> Rechazado (sin motivo, inválido)*

*@Test public void testC6() {*

*boolean ok = model.actualizarEstado("355", "Rechazado", "", "2025-06-06");*

*assertFalse(ok);*

*ActualizarPericialDTO dto = model.obtenerPericialPorReferencia("355");*

*assertEquals("Asignado", dto.getEstado());*

*}*

*// C7: Asignado ==> Cancelado (sin motivo, inválido)*

*@Test public void testC7() {*

*boolean ok = model.actualizarEstado("356", "Cancelado", "", "2025-06-07");*

*assertFalse(ok);*

*ActualizarPericialDTO dto = model.obtenerPericialPorReferencia("356");*

*assertEquals("Asignado", dto.getEstado());*

*}*

*// C8: Pendiente ==> Aceptado (inválido)*

*@Test public void testC8() {*

*boolean ok = model.actualizarEstado("357", "Aceptado", null, "2025-06-08");*

*assertFalse(ok);*

*ActualizarPericialDTO dto = model.obtenerPericialPorReferencia("357");*

*assertEquals("Pendiente", dto.getEstado());*

*}*

*// C9: Asignado ==> Estado inválido*

*@Test public void testC9() {*

*boolean ok = model.actualizarEstado("358", "Suspendido", null, "2025-06-09");*

*assertFalse(ok);*

*ActualizarPericialDTO dto = model.obtenerPericialPorReferencia("358");*

*assertEquals("Asignado", dto.getEstado());*

*}*

*}*

# **6.Pantallazo del resultado en JUNIT**

