




Universidad
Zaragoza

Historia Clínica

Pruebas

Pablo Doñate Navarro
Adnana Dragut
Proyecto Software

Versión: 1.0
Fecha: 16/05/2022

 Universidad Zaragoza	Historia Clínica Pruebas	Pablo Doñate Adnana Dragut
--	---	---

HOJA DE CONTROL

Proyecto	Historia Clínica		
Entregable	Pruebas		
Autor	Pablo Doñate Navarro y Adnana Dragut		
Versión/Edición	1.0	Fecha Versión	16/05/2022
Aprobado por	Sin aprobación	Fecha Aprobación	-
		Nº Total de Páginas	

REGISTRO DE CAMBIOS

Versión	Causa del Cambio	Responsable del Cambio	Fecha del Cambio
1.0	Versión inicial.	Pablo Doñate Navarro Adnana Dragut	16/05/2022

CONTROL DE DISTRIBUCIÓN

Nombre y Apellidos
Pablo Doñate Navarro
Adnana Dragut


 Universidad Zaragoza	Historia Clínica Pruebas	Pablo Doñate Adnana Dragut
--	---	---

TABLA DE CONTENIDOS

Pruebas	4
<i>Pruebas App Cliente</i>	<i>4</i>
<i>Fichero “LoginVistaTest”</i>	<i>4</i>
<i>Fichero “EpisodiosPacienteVistaTest”</i>	<i>6</i>
<i>Fichero “EditarSanitarioVistaTest”</i>	<i>9</i>
<i>Fichero “RecetaElectronicaVistaTest”</i>	<i>11</i>
<i>Pruebas App Servidor</i>	<i>13</i>
<i>Fichero “TestDarAltasPacienteSanitario”</i>	<i>13</i>
<i>Fichero “TestAñadirMedicVacunEpCit”</i>	<i>16</i>
<i>Fichero “TestObtenerHistoriaCompleta”</i>	<i>20</i>



1. Pruebas

En este apartado se detallarán las pruebas realizadas con el conjunto de librerías de JUnit para verificar el funcionamiento de las funcionalidades básicas tanto de la aplicación cliente como de la aplicación servidor.

1.1. Pruebas App Cliente

1.1.1. Fichero “LoginVistaTest”

Este fichero contiene dos métodos de test para comprobar el funcionamiento de la función que se encarga de verificar si la sintaxis de un correo electrónico es correcta, así como un método test para verificar el salto de una excepción que debe suceder en el caso en el que la sintaxis del correo es inválida.

Método setUp()

Este método se utiliza para inicializar algunas variables y componentes necesarios para realizar los test de los métodos correspondientes.

```
/**
 * Inicialización de las variables necesarias para realizar las pruebas.
 */
@Before
public void setUp() {
    this.commsTest = new Comms();
    this.loginVistaTest = new LoginVista(null, commsTest, "1");
    this.loginVistaTest.setVisible(false);
}
```

Método testSintaxisCorreoCorrectaCaso1()

Este método se utiliza para comprobar que el método detecta que la sintaxis del correo introducido es correcta.




```
/**
 * Comprueba el correcto funcionamiento del método "sintaxisCorreoCorrecta()"
 * utilizando para ello un correo válido.
 *
 */
@Test
public void testSintaxisCorreoCorrectaCaso1() {
    System.out.println("Test 1 - sintaxisCorreoCorrecta()");
    String email = "a@gmail.com";
    boolean expectedOutput = true;
    boolean result = loginVistaTest.sintaxisCorreoCorrecta(email);
    assertEquals(expectedOutput, result);
}
```

Método testSintaxisCorreoCorrectaCaso2()

Este método se utiliza para comprobar que el método detecta que la sintaxis del correo introducido es incorrecta.

```
/**
 * Comprueba el correcto funcionamiento del método "sintaxisCorreoCorrecta()"
 * utilizando para ello un correo inválido.
 *
 */
@Test
public void testSintaxisCorreoCorrectaCaso2() {
    System.out.println("Test 2 - sintaxisCorreoCorrecta()");
    String email = "a@.com";
    boolean expectedOutput = false;
    boolean result = loginVistaTest.sintaxisCorreoCorrecta(email);
    assertEquals(expectedOutput, result);
}
```

 Universidad Zaragoza	Historia Clínica Pruebas	Pablo Doñate Adnana Dragut
--	---	---

Método testExpectedException()

Este método se utiliza para comprobar que el método lanza una excepción cuando la sintaxis del correo electrónico es incorrecta.

```
/**
 * Comprueba el correcto funcionamiento de la excepción que debe saltar para
 * notificar un fallo en la sintaxis de un correo electrónico.
 *
 * @throws Exception
 */
@Test
public void testExpectedException() throws Exception {
    System.out.println("Test 1 - testExpectedException()");
    Throwable exception = assertThrows(Exception.class, () -> loginVistaTest.crearJsonUsuarioAVerificar());
    assertEquals(ERROR_SINTAXIS_EMAIL, exception.getMessage());
}
```

Resultado

Tests passed: 100,00 %

All 3 tests passed. (1,71 s)

- hms_tests.LoginVistaTest passed
 - testSintaxisCorreoCorrectaCaso1 passed (1,266 s)
 - testSintaxisCorreoCorrectaCaso2 passed (0,083 s)
 - testExpectedException passed (0,066 s)

Test 1 - sintaxisCorreoCorrecta()
Test 2 - sintaxisCorreoCorrecta()
Test 1 - testExpectedException()

1.1.2. Fichero “EpisodiosPacienteVistaTest”

Este fichero contiene dos métodos de test para comprobar el funcionamiento de algunos métodos básicos de la clase “EpisodiosPacienteVista”.



Método setUp()

Este método se utiliza para inicializar algunas variables y componentes necesarios para realizar los test de los métodos correspondientes.

```
/**
 * Inicialización de las variables necesarias para realizar las pruebas.
 *
 */
@Before
public void setUp() throws ParseException {
    this.episodiosPacienteVistaTest = new EpisodiosPacienteVista();
    this.episodiosPacienteVistaTest.setVisible(false);
    this.diagnostico_input_field = new JTextField("Resfriado común");

    // Inicialización para testCrearJsonEpisodioEditadoCaso1
    String oldstring1 = "2022-05-15 18:30:78.99";
    Date date1 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").parse(oldstring1);

    String oldstring2 = "2022-04-10 18:30:78.99";
    Date date2 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").parse(oldstring2);

    String oldstring3 = "2020-12-01 18:30:78.99";
    Date date3 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").parse(oldstring3);

    episodios.add(new EpisodioDeAtencionDTO(1, date1, "Dolor de Garganta", "Resfriado común"));
    episodios.add(new EpisodioDeAtencionDTO(2, date2, "Fiebre y dolor muscular", "Covid"));
    episodios.add(new EpisodioDeAtencionDTO(3, date3, "Dolor de cabeza y tos", ""));

    String[][] tableData = {{ "1", "2022-05-15", "Dolor de Garganta", "Resfriado común",
                               "2", "2022-04-10", "Fiebre y dolor muscular", "Covid",
                               "3", "2020-12-01", "Dolor de cabeza y tos", "" }};

    String[] tableColumn = {"Identificador", "Fecha", "Motivo/s", "Diagnóstico"};
    tabla_con_episodios = new JTable(tableData, tableColumn);
    episodiosPacienteVistaTest.setEpisodios(episodios);
    episodiosPacienteVistaTest.setDiagnostico_input_field(diagnostico_input_field);
    tabla_con_episodios.addRowSelectionInterval(0, 0);
    episodiosPacienteVistaTest.setTabla_con_episodios(tabla_con_episodios);
}
```

Método testCrearJsonEpisodioEditadoCaso1()

Este método se utiliza para comprobar que el método crea correctamente el json de un episodio al que se le ha añadido un diagnóstico.



```
/**
 * Comprueba el correcto funcionamiento del método "crearJsonEpisodioEditado()".
 *
 */
@Test
public void testCrearJsonEpisodioEditadoCaso1() throws Exception{
    System.out.println("Test 1 - testCrearJsonEpisodioEditadoCaso1()");
    String expectedOutput = "{\"id\":1,\"fecha\":\"May 15, 2022, 6:31:18 PM\", \"motivo\":\"Dolor de Garganta\", \"diagnostico\":\"Resfriado común\"}";
    String result = episodiosPacienteVistaTest.crearJsonEpisodioEditado();
    assertEquals(expectedOutput, result);
}
```

Método testObtenerIndexEpisodioPacienteCaso1()

Este método se utiliza para comprobar que el método obtiene correctamente el índice del episodio correspondiente al identificador pasado por parámetro.

```
/**
 * Comprueba el correcto funcionamiento del método "obtenerIndexEpisodioPaciente()".
 *
 */
@Test
public void testObtenerIndexEpisodioPacienteCaso1(){
    System.out.println("Test 1 - testObtenerIndexEpisodioPacienteCaso1()");
    int expectedOutput = 2;
    int result = episodiosPacienteVistaTest.obtenerIndexEpisodioPaciente(3);
    assertEquals(expectedOutput, result);
}
```

Resultado

Tests passed: 100,00 %

Both tests passed. (0,655 s)

- hms_tests.EpisodiosPacienteVistaTest passed
- testObtenerIndexEpisodioPacienteCaso1 passed (0,389 s)
- testCrearJsonEpisodioEditadoCaso1 passed (0,073 s)

Test 1 - testObtenerIndexEpisodioPacienteCaso1()
idEp 3

Test 1 - testCrearJsonEpisodioEditadoCaso1()



1.1.3. Fichero “EditarSanitarioVistaTest”

Este fichero contiene tres métodos de test para comprobar el funcionamiento de algunos métodos básicos de la clase “EditarSanitarioVista”.

Método setUp()

Este método se utiliza para inicializar algunas variables y componentes necesarios para realizar los test de los métodos correspondientes.

```
/**
 * Inicialización de las variables necesarias para realizar las pruebas.
 */
@Before
public void setUp() {
    this.editarSanitarioTest = new EditarSanitarioVista();
    this.tabla_con_sanitarios = new JTable();
    this._nombre_input_field_test = new JTextField();
    this.editarSanitarioTest.setVisible(false);
    this.sanitarioTest = new SanitarioDTO("Ana", "Test2", "Test2", "112233X", 987654321, "a@gmail.com", "4321", "Médico");

    // Inicialización para testObtenerDniSanitarioSeleccionadoCaso1
    String[][] tableData = {{"Juan", "Test1", "Test1", "123456X", "123456789", "j@gmail.com", "1234", "Médico"},
        {"Ana", "Test2", "Test2", "112233X", "987654321", "a@gmail.com", "4321", "Médico"},
        {"Maria", "Test3", "Test3", "112233X", "111222333", "m@gmail.com", "1111", "Otros"}};

    String[] tableColumn = {"Nombre", "Apellido1", "Apellido2", "DNI", "Teléfono", "Correo", "Puesto"};
    tabla_con_sanitarios = new JTable(tableData, tableColumn);
    editarSanitarioTest.setTabla_con_sanitarios(tabla_con_sanitarios);
}
```

Método testObtenerDniSanitarioSeleccionadoCaso1()

Este método se utiliza para comprobar que el método obtiene correctamente el dni de un sanitario seleccionado de la tabla.

```
/**
 * Comprueba el correcto funcionamiento del método "obtenerDniSanitarioSeleccionado()".
 */
@Test
public void testObtenerDniSanitarioSeleccionadoCaso1() {
    System.out.println("Test 1 - testObtenerDniSanitarioSeleccionadoCaso1()");
    String expectedOutput = "112233X";
    String result = editarSanitarioTest.obtenerDniSanitarioSeleccionado(1);
    assertEquals(expectedOutput, result);
}
```



Método testModificarNombreSanitarioCaso1()

Este método se utiliza para comprobar que el método no modifica el atributo nombre de un sanitario si dicho nombre no ha sido modificado.

```
/**
 * Comprueba el correcto funcionamiento del método "modificarNombreSanitario()",
 * pasándole para ello un nombre sin modificar de un sanitario.
 */
@Test
public void testModificarNombreSanitarioCaso1() {
    System.out.println("Test 1 - testModificarNombreSanitarioCaso1()");
    _nombre_input_field_test.setText("Ana");
    editarSanitarioTest.setNombre_input_field(_nombre_input_field_test);

    String expectedOutput = sanitarioTest.getNombre();
    System.out.println("Nombre sanitario antes de llamar al método de modificar --> " + sanitarioTest.getNombre());
    editarSanitarioTest.modificarNombreSanitario(sanitarioTest);
    System.out.println("Nombre sanitario después de llamar al método de modificar --> " + sanitarioTest.getNombre());
    String result = sanitarioTest.getNombre();
    assertEquals(expectedOutput, result);
}
```

Método testModificarNombreSanitarioCaso2()

Este método se utiliza para comprobar que el método modifica el atributo nombre de un sanitario si dicho nombre ha sido modificado.

```
/**
 * Comprueba el correcto funcionamiento del método "modificarNombreSanitario()",
 * pasándole para ello un nombre modificado de un sanitario.
 */
@Test
public void testModificarNombreSanitarioCaso2() {
    System.out.println("Test 2 - testModificarNombreSanitarioCaso2()");
    _nombre_input_field_test.setText("Ana Maria");
    editarSanitarioTest.setNombre_input_field(_nombre_input_field_test);

    String expectedOutput = "Ana Maria";
    System.out.println("Nombre sanitario antes de llamar al método de modificar --> " + sanitarioTest.getNombre());
    editarSanitarioTest.modificarNombreSanitario(sanitarioTest);
    System.out.println("Nombre sanitario después de llamar al método de modificar --> " + sanitarioTest.getNombre());
    String result = sanitarioTest.getNombre();
    assertEquals(expectedOutput, result);
}
```



Resultados

Tests passed: 100,00 %

All 3 tests passed. (0,592 s)

- hms_tests.EditarSanitarioVistaTest passed
- testModificarNombreSanitarioCaso1 passed (0,41 s)
- testModificarNombreSanitarioCaso2 passed (0,011 s)
- testObtenerDniSanitarioSeleccionadoCaso1 passed (0,011 s)

Test 1 - testModificarNombreSanitarioCaso1()
Nombre sanitario antes de llamar al método de modificar --> Ana
Nombre sanitario después de llamar al método de modificar --> Ana

Test 2 - testModificarNombreSanitarioCaso1()
Nombre sanitario antes de llamar al método de modificar --> Ana
Nombre sanitario después de llamar al método de modificar --> Ana Maria

Test 1 - testObtenerDniSanitarioSeleccionadoCaso1()

1.1.4. *Fichero “RecetaElectronicaVistaTest”*

Este fichero contiene dos métodos de test para comprobar el funcionamiento de algunos métodos básicos de la clase “RecetaElectronicaVista”.

Método setUp()

Este método se utiliza para inicializar algunas variables y componentes necesarios para realizar los test de los métodos correspondientes.

```
/**
 * Inicialización de las variables necesarias para realizar las pruebas.
 *
 */
@Before
public void setUp() {
    this.recetaElectronicaVistaTest = new RecetaElectronicaVista();
    this.recetaElectronicaVistaTest.setVisible(false);
}
```

Método testFechaFinPosteriorAFechaInicioCaso1()

Este método se utiliza para comprobar que el método comprueba correctamente que una fecha fin es anterior a una fecha inicio.



```
/**
 * Comprueba el correcto funcionamiento del método "fechaFinPosteriorAFechaInicio()",
 * pasándole para ello una fecha fin anterior a la fecha de inicio.
 *
 * @throws ParseException
 */
@Test
public void testFechaFinPosteriorAFechaInicioCaso1() throws ParseException{
    System.out.println("Test 1 - testFechaFinPosteriorAFechaInicioCaso1()");

    String oldstring1 = "2022-05-15 18:30:78.99";
    Date date1 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").parse(oldstring1);

    String oldstring2 = "2022-04-10 18:30:78.99";
    Date date2 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").parse(oldstring2);

    boolean expectedOutput = false;
    boolean result = recetaElectronicaVistaTest.fechaFinPosteriorAFechaInicio(date1, date2);
    assertEquals(expectedOutput, result);
}
```

Método testFechaFinPosteriorAFechaInicioCaso2()

Este método se utiliza para comprobar que el método comprueba correctamente que una fecha fin es posterior a una fecha inicio.

```
/**
 * Comprueba el correcto funcionamiento del método "fechaFinPosteriorAFechaInicio()",
 * pasándole para ello una fecha fin posterior a la fecha de inicio.
 *
 * @throws ParseException
 */
@Test
public void testFechaFinPosteriorAFechaInicioCaso2() throws ParseException{
    System.out.println("Test 2 - testFechaFinPosteriorAFechaInicioCaso2()");

    String oldstring1 = "2022-05-15 18:30:78.99";
    Date date1 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").parse(oldstring1);

    String oldstring2 = "2022-07-10 18:30:78.99";
    Date date2 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").parse(oldstring2);

    boolean expectedOutput = true;
    boolean result = recetaElectronicaVistaTest.fechaFinPosteriorAFechaInicio(date1, date2);
    assertEquals(expectedOutput, result);
}
```

Resultados



Universidad
Zaragoza

Historia Clínica
Pruebas

Pablo Doñate
Adnana Dragut

Tests passed: 100,00 %

30th tests passed. (0,39 s)

hms_tests.RecetaElectronicaVistaTest **passed**

Test 1 - testFechaFinPosteriorAFechaInicioCaso1()

Test 2 - testFechaFinPosteriorAFechaInicioCaso2()



1.2. Pruebas App Servidor

1.2.1. Fichero “TestDarAltasPacienteSanitario”

En este fichero de pruebas, se ha llevado a cabo la validación de los métodos encargados de Añadir un nuevo sanitario al sistema, añadir un nuevo paciente, modificar los campos de un sanitario existente, y finalmente, un método de validación de credenciales de acceso.

Para empezar, indicar que se ha implementado un método de conexión a la DB, que es ejecutado al comienzo de cada función de Test.

Método testConnectDB()

```
/**
 * Función que se ejecuta correctamente si se ha podido conectar a la DB.
 * En caso contrario, saltará excepción.
 *
 * @throws SQLException
 */
@Before
public void testConnectDB() throws SQLException {
    database = DataBaseControl.getInstance();
}
```

Método testAñadirNuevoSanitario()

Método que verifica si se ha dado de alta un nuevo sanitario. Para ello, se debe de comprobar tanto que se haya dado de alta un usuario nuevo, como luego su especialización en sanitario.

```
/**
 * Función que verifica si se ha podido dar de alta un nuevo sanitario.
 * Para ello, antes se añadirá en la db como usuario. Luego se le especializará
 * en sanitario.
 *
 * @throws SQLException
 */
@Test
public void testAñadirNuevoSanitario() throws SQLException {
    SanitarioDTO sanitario = new SanitarioDTO("Manolo", "García", "Martínez",
        "18123598X", 625174985, "manologar@gmail.com", "154876", "Médico");

    assertTrue(true == database.addUsuario(sanitario.getDni(), sanitario.getEmail(),
        sanitario.getContraseña()));
    assertTrue(true == database.addSanitario(sanitario));
}
```



Método testEditarSanitarioExistente()

Método que edita algún campo de un sanitario, y lo registra en la base de datos. Finalmente, se verifica que se haya realizado correctamente.

```
/**
 * Función que modifica un sanitario existente.
 * Para ello, en la prueba se ha modificado el primer apellido.
 * Se ha verificado que se modifica correctamente el sanitario.
 *
 * @throws SQLException
 */
@Test
public void testEditarSanitarioExistente() throws SQLException {
    SanitarioDTO sanitarioModif = new SanitarioDTO("Manolo", "Pérez", "Martínez",
        "18123598X", 625174985, "manologar@gmail.com", "154876", "Médico");
    assertTrue(true == database.editarSanitario(sanitarioModif));
}
```

Método testVerificarUsuario()

Método que verifica las credenciales de acceso de un usuario. Para ello, se verifica tanto el acceso de un usuario administrador, como de un sanitario, y de unas credenciales incorrectas.

```
/**
 * Función que verifica si dado un usuario, existe en la DB.
 * Se realizan tres pruebas diferentes:
 * - Usuario encontrado, pero es Sanitario, no admin.
 * - Usuario encontrado, es admin.
 * - Usuario no encontrado.
 *
 * @throws SQLException
 * @throws ClassNotFoundException
 */
@Test
public void testVerificarUsuario() throws SQLException, ClassNotFoundException {
    UsuarioDTO usuarioCorrectoSan = new UsuarioDTO("123456789X", "antlopg@gmail.com",
        "81dc9bdb52d04dc20036dbd8313ed055");
    UsuarioDTO usuarioCorrectoAdm = new UsuarioDTO("987987987A", "admin@hsw.com",
        "81dc9bdb52d04dc20036dbd8313ed055");
    UsuarioDTO usuarioIncorrectoSan = new UsuarioDTO("12345678", "ant@gmail.com",
        "123456789");

    assertTrue("UsuarioDTO(correoElectronico=antlopg@gmail.com, contraseña=81dc9bdb52d04dc20036dbd8313ed055, admin=false)"
        .equals(database.verificarUsuario(usuarioCorrectoSan).toString()));
    assertTrue("UsuarioDTO(correoElectronico=admin@hsw.com, contraseña=81dc9bdb52d04dc20036dbd8313ed055, admin=true)"
        .equals(database.verificarUsuario(usuarioCorrectoAdm).toString()));
    assertTrue(null == database.verificarUsuario(usuarioIncorrectoSan));
}
```




Método testAñadirNuevoPaciente()

Método que añade un nuevo paciente a la DB, y verifica que se ha registrado correctamente.

```
/**
 * Función que verifica si se ha podido añadir un nuevo paciente.
 *
 * @throws SQLException
 */
@Test
public void testAñadirNuevoPaciente() throws SQLException {
    PacienteDTO paciente = new PacienteDTO("256047A", "Alejandro", "Domingo",
        "Navarro", "Ácido clavulánico", 18, 1.87, 78.53);
    assertTrue(true == database.addPaciente(paciente));
}
```

Resultado



 Universidad Zaragoza	Historia Clínica Pruebas	Pablo Doñate Adnana Dragut
--	---	---

1.2.2. Fichero “TestAñadirMedicVacunEpCit”

En este fichero de pruebas, se ha llevado a cabo la validación de los métodos encargados de añadir un episodio a un paciente, añadir una cita a un paciente, añadir medicamentos a la receta de un paciente, y finalmente, un método de asignar vacunas a un paciente.

Para empezar, indicar que se ha implementado un método de conexión a la DB, que es ejecutado al comienzo de cada función de Test.

Método testAñadirEpisodioPaciente()

Método que verifica que se ha podido asignar un nuevo episodio de atención a un paciente.

```
/**
 * Función que verifica la asignación de un nuevo episodio de atención
 * al paciente creado en la función de test anterior.
 *
 * @throws SQLException
 * @throws ParseException
 */
@Test
public void testAñadirEpisodioPaciente() throws SQLException, ParseException {
    String string_date = "2022-05-20";
    SimpleDateFormat f = new SimpleDateFormat("yyyy-MM-dd");

    Date fecha = f.parse(string_date);

    EpisodioAtencionDTO episodio = new EpisodioAtencionDTO(10, fecha,
        "Alejandro ha acudido en consecuencia de dolor de cabeza.",
        "El paciente ha sido positivo en COVID-19");
    assertTrue(true == database.nuevoEpisodio(episodio, "256047A"));
}
```



Método testAñadirCitaPaciente()

Método que verifica que se ha podido asignar una nueva cita a un paciente.

```
/**
 * Función que verifica la asignación de una nueva cita
 * al paciente creado en la función de test anterior con el
 * sanitario creado en otra función de test.
 *
 * @throws SQLException
 * @throws ParseException
 */
@Test
public void testAñadirCitaPaciente() throws SQLException, ParseException {
    String string_date = "2022-05-20";
    String string_hora = "13:15";
    SimpleDateFormat f = new SimpleDateFormat("yyyy-MM-dd");
    SimpleDateFormat h = new SimpleDateFormat("HH:mm");

    Date fecha = f.parse(string_date);
    Date hora = h.parse(string_hora);

    UbicacionDTO ubicacion = new UbicacionDTO("Teruel",
        "Hospital Obispo Polanco", "14A");
    CitaDTO cita = new CitaDTO(10, "18123598X", ubicacion, fecha, hora,
        "El paciente tiene cita con su médico de cabecera.");

    assertTrue(true == database.nuevaCita(cita, "256047A"));
}
```



Método testAñadirMedicamentoPaciente()

Método que verifica que se ha podido añadir un nuevo medicamento a la receta de un paciente.

```
/**
 * Función que verifica la inserción de un medicamento
 * en la receta del paciente creado en una función de test anterior.
 *
 * @throws SQLException
 * @throws ParseException
 */
@Test
public void testAñadirMedicamentoPaciente() throws SQLException, ParseException {
    String string_date_1 = "2022-05-20";
    String string_date_2 = "2022-05-27";
    SimpleDateFormat f = new SimpleDateFormat("yyyy-MM-dd");

    Date fechaInicio = f.parse(string_date_1);
    Date fechaFin = f.parse(string_date_2);

    MedicamentoDTO medicamento = new MedicamentoDTO(1, "Paracetamol");
    MedicamentoPacienteDTO medicamentoPaciente =
        new MedicamentoPacienteDTO(10, medicamento, fechaInicio, fechaFin);
    assertTrue(true == database.añadirMedicamentoAPaciente(medicamentoPaciente, "256047A"));
}
```

Método testAñadirVacunaPaciente()

Método que verifica que se ha podido asignar una nueva vacuna a la historia de un paciente.

```
/**
 * Función que verifica la inserción de una vacuna
 * en la historia del paciente creado en una función de test anterior.
 *
 * @throws SQLException
 * @throws ParseException
 */
@Test
public void testAñadirVacunaPaciente() throws SQLException, ParseException {
    String string_date = "2022-05-20";
    SimpleDateFormat f = new SimpleDateFormat("yyyy-MM-dd");

    Date fecha = f.parse(string_date);

    VacunaDTO vacuna = new VacunaDTO(1, "Hepatitis B");
    VacunaPacienteDTO vacunaPaciente = new VacunaPacienteDTO(10, vacuna, fecha);
    assertTrue(true == database.añadirVacunaAPaciente(vacunaPaciente, "256047A"));
}
```



Resultado

Tests passed: 100,00 %

All 4 tests passed. (0,699 s)

- ✓ TestAñadirMedicVacunEpCit passed
- ✓ testAñadirEpisodioPaciente passed (0,568 s)
- ✓ testAñadirCitaPaciente passed (0,016 s)
- ✓ testAñadirMedicamentoPaciente passed (0,019 s)
- ✓ testAñadirVacunaPaciente passed (0,027 s)



1.2.3. *Fichero “TestObtenerHistoriaCompleta”*

En este fichero de pruebas, se ha llevado a cabo la validación del método asociado a obtener la historia completa de un paciente. Es decir, episodios de atención, vacunas y medicamentos.

Para empezar, indicar que se ha implementado un método de conexión a la DB, que es ejecutado al comienzo de la función de Test.

Método testObtenerHistoriaCompletaPaciente()

Método que verifica que se obtiene la historia completa de un paciente.

```
/**
 * Función que obtiene la historia completa de un paciente.
 * Para ello, se compara la historia que yo considero que debo de obtener, con
 * la historia devuelta por la DB.
 * La historia que creo yo,
 *
 * @throws SQLException
 */
@Test
public void testObtenerHistoriaCompletaPaciente() throws SQLException, ParseException {
    Gson gson = new Gson();
    SimpleDateFormat f = new SimpleDateFormat("yyyy-MM-dd");

    String string_date_ep = "2022-05-20";
    String string_date_med_1 = "2022-05-20";
    String string_date_med_2 = "2022-05-27";
    String string_date_vac = "2022-05-20";

    Date fechaEp = f.parse(string_date_ep);
    Date fechaInicio = f.parse(string_date_med_1);
    Date fechaFin = f.parse(string_date_med_2);
    Date fechaVac = f.parse(string_date_vac);

    EpisodioAtencionDTO[] episodios = new EpisodioAtencionDTO[] {
        {new EpisodioAtencionDTO(10, fechaEp, "Alejandro ha acudido en consecuencia de dolor de cabeza.",
            "El paciente ha sido positivo en COVID-19")};
    };

    MedicamentoDTO medicamento = new MedicamentoDTO(1, "Paracetamol");
    MedicamentoPacienteDTO[] medicamentos = new MedicamentoPacienteDTO[] {
        {new MedicamentoPacienteDTO(10, medicamento, fechaInicio, fechaFin)};
    };

    VacunaDTO vacuna = new VacunaDTO(1, "Hepatitis B");
    VacunaPacienteDTO[] vacunas = new VacunaPacienteDTO[] {
        {new VacunaPacienteDTO(10, vacuna, fechaVac)};
    };

    HistoriaPacienteDTO historia = new HistoriaPacienteDTO(Arrays.asList(episodios),
        Arrays.asList(medicamentos), Arrays.asList(vacunas));
    HistoriaPacienteDTO historiaRecibida =
        gson.fromJson(database.obtenerHistoriaPaciente("256047A"), HistoriaPacienteDTO.class);

    assertTrue(historia.toJson().equals(historiaRecibida.toJson()));
}
```



Resultado

Tests passed: 100,00 %

The test passed. (0,718 s)

- ✓ TestObtenerHistoriaCompleta **passed**
- ✓ testObtenerHistoriaCompletaPaciente **passed** (0,649 s)