

UNIVERSIDAD DE LAS FUERZAS ARMADAS-ESPE SEDE SANTO DOMINGO

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN - DCCO-SS

CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

PERIODO : Noviembre 2023 – Marzo 2024

ASIGNATURA : Metodología de desarrollo de Software

TEMA : Tarea 4

NOMBRES : Aldo Saula, Esteban Larco

SEMESTRE : Tercero.

DOCENTE : Ing. Javier Cevallos Farías.

FECHA DE ENTREGA : 07/02/2024

SANTO DOMINGO - ECUADOR

2024

INDICE

1. Introducción	1
2. Sistemas de Objetivos	2
Objetivo General:	2
Objetivos Específicos:	2
3. Desarrollo	2
MenAdminTest	2
4. Conclusiones	6
5. Recomendaciones	7
6. Bibliografía/ Referencias	7

1. Introducción

En la gestión clínica, el rendimiento y la solidez del software son factores clave para garantizar un flujo de trabajo eficiente y servicios médicos de alta calidad. En este contexto, las pruebas, especialmente las pruebas de categorías, desempeñan un papel clave en las pruebas y verificación del sistema. SISSA Digital. (2023).

Los test de las clases son una parte importante del proceso de desarrollo de software porque permiten evaluar el comportamiento individual de las categorías en un sistema. Es un conjunto de clases (framework) que permite que las clases Java se ejecuten de manera controlada para que se pueda evaluar el comportamiento de cada método en la categoría para ver si funciona según lo previsto. En un proyecto de gestión clínica donde interactúan múltiples categorías para gestionar pacientes, citas, registros médicos y otros aspectos críticos, se vuelve aún más importante realizar una amplia gama de pruebas. Peña, J. M. F. (2005).

En este informe, analizamos la importancia de los test en las clases de proyectos de gestión clínica y destacamos cómo estas pruebas pueden ayudar a detectar errores tempranamente, mejorar la calidad del código y optimizar el rendimiento del sistema. Además, examinaremos las mejores prácticas en el diseño y ejecución de pruebas de clases efectivas y los beneficios que aportan al desarrollo y mantenimiento de software en el contexto específico de la gestión clínica.

2. Sistemas de Objetivos

Objetivo General:

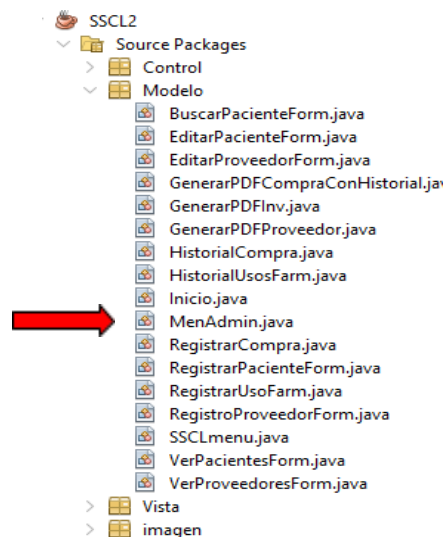
Evaluar y garantizar la funcionalidad, fiabilidad y eficiencia de las clases en el sistema de gestión de clínica mediante test unitarios, contribuyendo a la mejora continua de la calidad del software.

Objetivos Específicos:

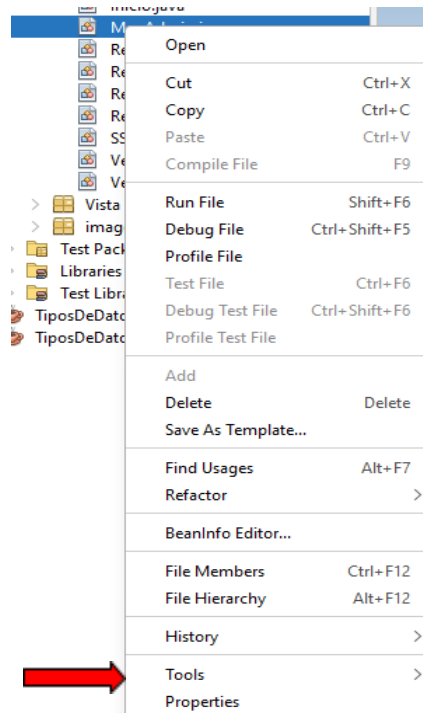
- Verificar la correcta implementación de las funciones y métodos dentro de cada clase del sistema, asegurando que cumplan con los requisitos funcionales y los estándares de diseño establecidos
- Identificar y corregir posibles vulnerabilidades, errores lógicos y problemas de rendimiento en las clases del sistema de gestión de clínica, con el fin de mejorar la estabilidad y la eficacia del software.

3. Desarrollo

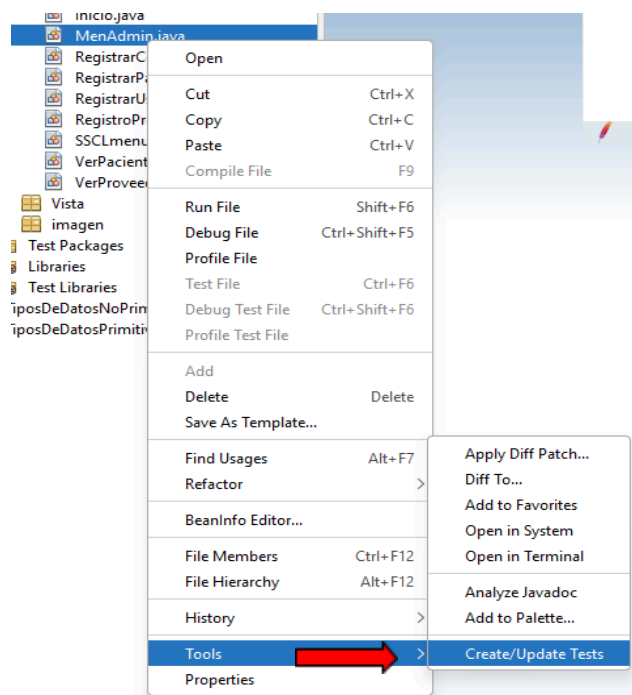
MenAdminTest



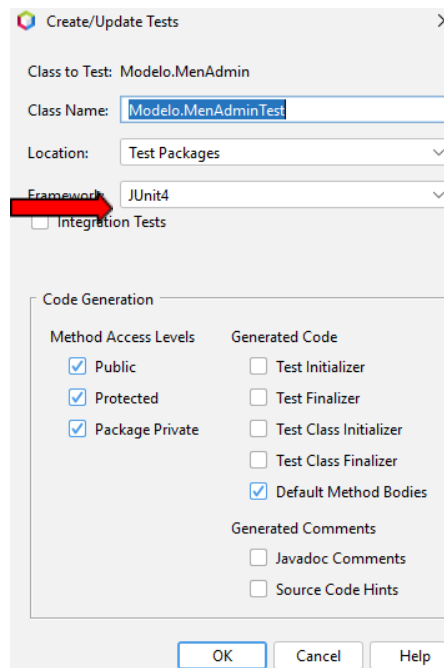
- Nos dirigimos a la clase a la cual se le va realizar el test unitario, nos colocamos encima de ella presionamos clic derecho y se desplegará múltiples opciones.



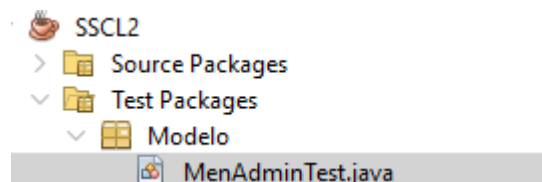
- Nos dirigimos a la opción de tools y se nos desplegará las siguientes opciones



- Elegimos la opción CreateUpdateTests y se nos desplegará el siguiente formulario



- En framework elegimos la opción “JUnit4” y marcamos las casillas que se ven en captura y le damos a “OK”



- En Test Packages se nos creará un paquete y una clase que es donde se hará el testeo

```

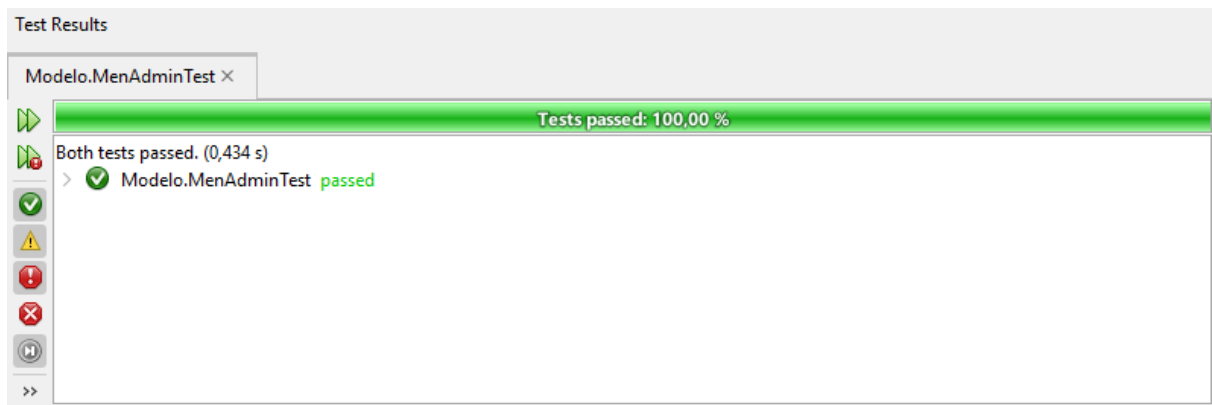
1 package Modelo;
2
3 import org.junit.Test;
4 import static org.junit.Assert.*;
5
6 public class MenAdminTest {
7
8     @Test
9     public void testVerificarCredenciales_CorrectCredentials_ReturnsTrue() {
10         // Se crea una instancia de la clase MenAdmin para realizar las pruebas.
11         MenAdmin menAdmin = new MenAdmin();
12
13         // Se llama al método verificarCredenciales con credenciales correctas y se guarda el resultado.
14         boolean result = menAdmin.verificarCredenciales("JavierCevallos", "129062023");
15
16         // Se verifica si el resultado es verdadero.
17         assertTrue(result);
18     }
19
20     @Test
21     public void testVerificarCredenciales_IncorrectCredentials_ReturnsFalse() {
22         // Se crea una instancia de la clase MenAdmin para realizar las pruebas.
23         MenAdmin menAdmin = new MenAdmin();
24
25         // Se llama al método verificarCredenciales con credenciales incorrectas y se guarda el resultado.
26         boolean result = menAdmin.verificarCredenciales("AldoSaula", "12345");
27
28         // Se verifica si el resultado es falso.
29         assertFalse(result);
30     }
31 }

```

- La clase MenAdminTest es una clase de prueba que contiene métodos de prueba para probar la funcionalidad de la clase MenAdmin
- Se importa la clase org.junit.Test que es necesaria para usar las anotaciones de prueba de JUnit.
- Los métodos de prueba deben ser anotados con @Test para que JUnit los reconozca como métodos de prueba.
- En el primer método de prueba (testVerificarCredenciales_CorrectCredentials_ReturnsTrue), se crea una instancia de MenAdmin, se llama al método verificarCredenciales con credenciales correctas y se verifica si devuelve verdadero (true).
- En el segundo método de prueba (testVerificarCredenciales_IncorrectCredentials_ReturnsFalse), se crea otra

instancia de MenAdmin, se llama al método verificarCredenciales con credenciales incorrectas y se verifica si devuelve falso (false).

- Estos métodos de prueba proporcionan una forma de verificar el comportamiento del método verificarCredenciales de la clase MenAdmin en diferentes casos de entrada.



- Test pasado al 100%

4. Conclusiones

Las pruebas unitarias son una herramienta invaluable para verificar el comportamiento esperado de los métodos en una clase. En este caso, las pruebas han demostrado la capacidad del método verificarCredenciales de la clase MenAdmin para retornar resultados coherentes según las credenciales proporcionadas.

La implementación de pruebas unitarias proporciona una mayor confianza en el funcionamiento del código y facilita la detección temprana de posibles errores o fallos en la lógica de programación. En este ejemplo, las pruebas han permitido identificar el comportamiento correcto e incorrecto del método verificarCredenciales, lo que contribuye a la robustez del sistema.

5. Recomendaciones

Es importante mantener un conjunto completo de pruebas unitarias que cubran diferentes casos de uso y situaciones de entrada para garantizar una cobertura adecuada del código. Esto incluye no solo casos de éxito, como credenciales correctas, sino también casos de fallo, como credenciales incorrectas o valores atípicos.

Es esencial mantener las pruebas unitarias actualizadas y ejecutarlas regularmente durante todo el ciclo de desarrollo del software. Esto garantiza que cualquier cambio realizado en el código sea validado automáticamente, lo que facilita la detección temprana de posibles problemas y contribuye a la estabilidad y mantenibilidad del sistema a lo largo del tiempo.

6. Bibliografía/ Referencias

SISSA Digital. (2023). Software para hospitales: desarrollo de software a medida como herramienta clave. LinkedIn.com.
<https://es.linkedin.com/pulse/software-para-hospitales-desarrollo-de-medida-como-herramienta-fopr/>

Peña, J. M. F. (2005). Pruebas de unidad utilizando JUnit. Wwww.uv.mx.
<https://www.uv.mx/personal/jfernandez/files/2010/07/Uso-JUnit.pdf>