Proyecto Certificador Desarrollo Software 3

Ficha de seguimiento N°7: Desarrollo de pruebas de calidad del software y cierre del proyecto.

Indicador de logro N°4
Realiza pruebas de calidad para verificar el ciclo de vida del software. seleccionadas.

idat

Contenido

1. Objetivo: 3
2. Pasos: 3
3. Instrumentos / Herramientas: 3
4. Desarrollo 4

idat

Ficha de seguimiento 7: Desarrollo de pruebas de calidad del software y cierre del proyecto.

En esta ficha encontrarás los pasos para construir/trabajar esta parte de tu proyecto. Tienes un formato o plantilla para completar los datos y resultados de la actividad de esta ficha. Tu profesor/a revisará lo avanzado, y te brindará una retroalimentación.

Objetivo:

Se espera que el estudiante realice pruebas de calidad para verificar el cumplimiento del ciclo de vida del software, aplicando las diversas técnicas y herramientas seleccionadas; para ello, de acuerdo con el enfoque de gestión de proyectos elegido, se realizarán esfuerzos en paralelo: por un lado, la Gestión del Proyecto (Fase Ejecución y Cierre) y por el otro el Desarrollo del Software en lo que compete a las Pruebas Unitarias e Integrales.

1. Pasos:

Paso 1: Gestión del Proyecto (1-A Fase Ejecución y 1-B Cierre)
Paso 2: Pruebas Unitarias
Paso 3: Pruebas Integrales
Paso 4: Elaboración de las conclusiones y recomendaciones.

*Nota: Paso 1 es paralelo a los pasos 2 y 3

2. Instrumentos / Herramientas:

Manejo básico de las siguientes herramientas:

- ✓ Procesador de texto MS-WORD o Google Docs
- ✓ Presentaciones MS-POWEP POINT o Google Docs
- ✓ Hoja de Cálculo MS-EXCEL o Google Docs
- ✓ Base de Datos MS-SQL SERVER
- ✓ Base de Datos MySQL
- ✓ Base de Datos PostgreSQL
- ✓ Lenguaje de Programación PHP
- ✓ Lenguaje de Programación JAVA
- ✓ Lenguaje de Programación VISUAL STUDIO .NET (C# o VB)
- ✓ Lenguaje de Programación PYTHON
- ✓ Lenguaje de Programación KOTLIN

3. Desarrollo

Paso 1: Gestión del Proyecto (1 – A: Fase de Ejecución)

La Fase de Ejecución es donde se concretan las tareas planificadas para alcanzar los objetivos del proyecto. Se desarrollan las actividades, se gestionan los recursos y se controla el avance. Esta fase es independiente del enfoque de gestión elegido (predictivo o ágil). En resumen: se materializa el plan del proyecto, se gestionan recursos y actividades, se controla el avance y se realizan ajustes.

Paso 2: Pruebas Unitarias

Las pruebas unitarias constituyen un proceso crítico en el desarrollo de sistemas de información, en el que cada componente o unidad del software se verifica de manera individual para asegurarse de que funciona correctamente de acuerdo con las especificaciones. Este paso es fundamental tanto en enfoques predictivos como en ágiles, aunque la implementación puede variar. En métodos predictivos, las pruebas unitarias se realizan después de completar todos los componentes individuales, siguiendo un plan de pruebas detallado. En enfoques ágiles, estas pruebas se ejecutan continuamente a medida que cada función o módulo se desarrolla, facilitando la identificación temprana de errores y la integración constante. Las pruebas unitarias no solo ayudan a validar la funcionalidad y el rendimiento de los componentes, sino que también garantizan que futuras modificaciones no introduzcan nuevos errores, contribuyendo a la calidad y sostenibilidad del software.

Ejemplo de Pruebas Unitarias:

ejemplo de cómo podrías implementar pruebas unitarias en un proyecto de desarrollo de software, utilizando Python y la biblioteca unittest, que es un marco de prueba estándar en Python:

Supongamos que tienes una función en un módulo de Python que calcula el IGV (Impuesto General a las Ventas) de un producto dado su precio base. La función podría lucir así:

Código de la función

python Copy code

```python
def calcular_igv(precio_base, tasa_igv=0.18):
    """Calcula el IGV de un producto dado un precio base y una tasa de IGV.
    Por defecto, la tasa de IGV es del 18%."""
    return precio_base * tasa_igv
```

Pruebas unitarias con `unittest`

```python
import unittest
from tu_modulo import calcular_igv  # Asume que la función está en 'tu_modulo.py'

class TestCalcularIGV(unittest.TestCase):

    def test_calculo_igv_estandar(self):
        """Test que verifica que el IGV se calcula correctamente con la tasa estándar
        precio = 100  # Precio base del producto
        resultado_esperado = 18.0  # Resultado esperado (100 * 0.18)
        self.assertAlmostEqual(calcular_igv(precio), resultado_esperado, places=2)

    def test_calculo_igv_personalizado(self):
        """Test que verifica que el IGV se calcula correctamente con una tasa persona
        precio = 200
        tasa_igv = 0.20  # Una tasa personalizada del 20%
        resultado_esperado = 40.0  # Resultado esperado (200 * 0.20)
        self.assertAlmostEqual(calcular_igv(precio, tasa_igv), resultado_esperado, pl

    def test_calculo_igv_cero(self):
        """Test que verifica que el IGV es cero cuando el precio base es cero."""
        precio = 0
        resultado_esperado = 0.0
        self.assertEqual(calcular_igv(precio), resultado_esperado)

if __name__ == '__main__':
    unittest.main()
```

Nota: Las imágenes son sólo referenciales, dependerá del tipo de pruebas realizadas y la(s) herramientas de pruebas empleadas.

Las pruebas unitarias siguen verificando que la función maneje correctamente diferentes entradas y que calcule el impuesto adecuadamente bajo diferentes escenarios. Esto es esencial para asegurar la precisión financiera en el manejo de impuestos dentro de aplicaciones de software.

Paso 3: Pruebas Integrales

Una vez completadas las pruebas unitarias, el proyecto avanza hacia las pruebas integrales. Esta etapa verifica que todos los componentes del sistema funcionen en conjunto de manera efectiva y conforme a los requisitos globales del proyecto. En la gestión predictiva, las pruebas integrales se planifican meticulosamente una vez que todas las unidades están testadas y ensambladas. En el enfoque ágil, la integración se realiza de manera incremental, adaptándose a los ciclos iterativos de entrega, lo que permite identificar desajustes y conflictos entre módulos de forma precoz. Estas pruebas son vitales para asegurar la cohesión y la interoperabilidad del sistema, así como para validar los flujos de trabajo y procesos transversales que involucran múltiples componentes del sistema, asegurando que el producto final sea robusto y funcionalmente integrado.

Escenario para pruebas integrales:

Supongamos que estás desarrollando un sistema de facturación que incluye, además del cálculo del IGV, funcionalidades para gestionar clientes y emitir facturas. Los módulos relevantes podrían ser:

- Módulo de Clientes: gestiona la información del cliente.
- Módulo de Productos: maneja información de productos y precios.
- Módulo de Facturación: integra los datos de clientes y productos para emitir facturas, incluyendo el cálculo del IGV.

Ejemplo de pruebas integrales:

Aquí te muestro cómo podrías estructurar una prueba integral usando Python para asegurar que los módulos interactúan correctamente:

```
import unittest
from sistema_facturacion import Cliente, Producto, generar_factura

class TestIntegracionSistemaFacturacion(unittest.TestCase):

    def setUp(self):
        """Configura los objetos necesarios para las pruebas."""
        # Crear un cliente
        self.cliente = Cliente(nombre="Empresa XYZ", ruc="20123456789")

        # Crear productos
        self.producto1 = Producto(nombre="Laptop", precio=3000)
        self.producto2 = Producto(nombre="Mouse", precio=50)

        # Tasas de IGV
        self.tasa_igv = 0.18

    def test_emitir_factura(self):
        """Verifica que la factura se genera correctamente con todos los componentes integrados."""
        # Generar factura
        factura = generar_factura(cliente=self.cliente, productos=[self.producto1, self.producto2], tasa_igv=self.tasa_igv)

        # Verificar la información del cliente en la factura
        self.assertEqual(factura.cliente.nombre, "Empresa XYZ")

        # Verificar que los productos están listados en la factura
        self.assertIn(self.producto1, factura.productos)
        self.assertIn(self.producto2, factura.productos)

        # Verificar el cálculo total de la factura incluyendo el IGV
        total_sin_igv = self.producto1.precio + self.producto2.precio
        total_con_igv = total_sin_igv + (total_sin_igv * self.tasa_igv)
```

```
        self.assertAlmostEqual(factura.total, total_con_igv)

if __name__ == '__main__':
    unittest.main()
```

Puntos clave de las pruebas integrales:

- ✓ SetUp: Configura objetos de prueba, como instancias de cliente y productos que serán usados en varias pruebas.
- ✓ Test de Integración: Verifica que los módulos interactúan como se espera. Por ejemplo, asegura que el módulo de facturación recoja correctamente los datos del cliente y del producto, aplique el IGV, y calcule el total correcto.
- ✓ Verificaciones: Comprueba que todos los elementos están presentes y correctos, como los datos del cliente en la factura, la lista de productos, y el cálculo total que incluye el IGV.

Este tipo de prueba es crucial para detectar problemas en las interfaces entre módulos, asegurando que el sistema completo funcione armoniosamente antes de ser puesto en producción.

Paso 4: Elaboración de las conclusiones y recomendaciones.

El último paso del proceso de seguimiento de un proyecto de desarrollo de sistemas de información es la elaboración de conclusiones y recomendaciones. Mínimo tres (3) y Máximo cinco (5).

Esta etapa consiste en evaluar la ejecución del proyecto en su totalidad, identificar los logros alcanzados y los desafíos enfrentados, y formular recomendaciones para futuras iniciativas.

Es importante analizar tanto los aspectos técnicos del proyecto como la gestión del mismo, incluyendo la eficacia de las metodologías utilizadas, la gestión de recursos y la comunicación entre los equipos.

Las conclusiones se basan en hechos ocurridos durante el Proyecto, deben proporcionar una visión clara del rendimiento del proyecto frente a los objetivos originales, mientras que las recomendaciones deben enfocarse en mejorar prácticas, optimizar procesos y prevenir problemas recurrentes. Lo ideal es que para cada conclusión le corresponda una recomendación

Este documento final actúa como un insumo crucial para la mejora continua y la toma de decisiones estratégicas en proyectos futuros.

Formato de Presentación 1 – Una sola Tabla de Conclusiones y Recomendaciones:

Conclusiones	**Recomendaciones**
1.- Conclusión 1	**1.-** Recomendación 1
2.- Conclusión 2	**2.-** Recomendación 3

3.- Conclusión 3	**3.-** Recomendación 3

Ejemplo de Uso de Formato 1:

Conclusiones	Recomendaciones
1.- La integración continua de componentes durante el desarrollo agilizó la detección y corrección de errores, pero generó retrasos cuando los módulos no estaban adecuadamente alineados con los requerimientos del sistema.	**1.-** Implementar sesiones de revisión de requisitos más frecuentes y detalladas entre los desarrolladores y los stakeholders para asegurar una alineación constante y evitar desviaciones en los requerimientos durante las fases de integración.
2.- Las pruebas unitarias demostraron ser eficaces para asegurar la funcionalidad de componentes individuales, pero las pruebas de rendimiento no se planificaron con la misma anticipación, resultando en problemas de escalabilidad al final del proyecto.	**2.-** Desarrollar un plan de pruebas de rendimiento desde las primeras etapas del proyecto, integrándolo con las pruebas unitarias para asegurar tanto la funcionalidad como la escalabilidad del sistema desde el inicio.
3.- La comunicación entre el equipo de desarrollo y los stakeholders fue eficiente en las etapas iniciales del proyecto, pero decayó hacia la fase de cierre, afectando la claridad y el ajuste final del producto entregado.	**3.-** Establecer un protocolo de comunicación continua que incluya reuniones regulares y actualizaciones de estado a lo largo de todo el proyecto, enfocándose especialmente en mantener el mismo nivel de interacción durante la fase de cierre.

Formato de Presentación 2 – Tabla de Conclusiones separada de la Tabla de Recomendaciones:

Conclusiones
1.- Conclusión 1
2.- Conclusión 2
3.- Conclusión 3

Recomendaciones
1.- Recomendación 1
2.- Recomendación 3
3.- Recomendación 3

Ejemplo de Uso de Formato 2:

Conclusiones
1.- La integración continua de componentes durante el desarrollo agilizó la detección y corrección de errores, pero generó retrasos cuando los módulos no estaban adecuadamente alineados con los requerimientos del sistema.

2.- Las pruebas unitarias demostraron ser eficaces para asegurar la funcionalidad de componentes individuales, pero las pruebas de rendimiento no se planificaron con la misma anticipación, resultando en problemas de escalabilidad al final del proyecto.
3.- La comunicación entre el equipo de desarrollo y los stakeholders fue eficiente en las etapas iniciales del proyecto, pero decayó hacia la fase de cierre, afectando la claridad y el ajuste final del producto entregado.

Recomendaciones
1.- Implementar sesiones de revisión de requisitos más frecuentes y detalladas entre los desarrolladores y los stakeholders para asegurar una alineación constante y evitar desviaciones en los requerimientos durante las fases de integración.
2.- Desarrollar un plan de pruebas de rendimiento desde las primeras etapas del proyecto, integrándolo con las pruebas unitarias para asegurar tanto la funcionalidad como la escalabilidad del sistema desde el inicio.
3.- Establecer un protocolo de comunicación continua que incluya reuniones regulares y actualizaciones de estado a lo largo de todo el proyecto, enfocándose especialmente en mantener el mismo nivel de interacción durante la fase de cierre.

Paso 1: Gestión del Proyecto (1 – B: Fase de Cierre)

La fase de cierre del proyecto es crucial para asegurar que todos los aspectos del proyecto se concluyan de manera ordenada y eficiente, y que el proyecto entregue los resultados esperados. Una Guía de Cierre del Proyecto debe incluir los siguientes elementos:

1.- Revisión del Proyecto: Evaluación exhaustiva de los objetivos alcanzados en comparación con los objetivos planificados. Incluye la revisión de todas las entregas y la confirmación de que todas las especificaciones del proyecto han sido satisfechas.

2.- Documentación: Asegurar que toda la documentación del proyecto, como manuales de usuario, documentación técnica y registros de cambios, esté completa y archivada adecuadamente.

3.- Liberación de Recursos: Proceso de desmovilización de los recursos del proyecto, incluyendo la liberación de personal, la finalización de contratos con proveedores y la devolución de equipos o recursos alquilados.

4.- Evaluación de Desempeño: Realizar evaluaciones de desempeño del equipo de proyecto, identificando fortalezas y áreas de mejora para futuras iniciativas.

5.- Lecciones Aprendidas: Compilación de lecciones aprendidas durante el proyecto para mejorar procesos en futuros proyectos. Este documento debe ser accesible para otros equipos y proyectos futuros.

Ejemplo de Formato (Artefacto) de Cierre:

Documento de Lecciones Aprendidas – Proyecto "X"
Fecha: [Fecha de cierre del proyecto; formato: dd/mm/aaaa] **Participantes:** [Nombres y Apellidos de los miembros del Equipo de proyecto, incluyendo gestores, desarrolladores y analistas] **Objetivo del Documento:** [proporcionar un recurso valioso que detalle las experiencias, tanto positivas como negativas, surgidas durante la ejecución del proyecto para optimizar la gestión de proyectos futuros]. Contenido: **1.- Planificación y Gestión**: [Reflexiones sobre la efectividad de las estrategias de planificación utilizadas, incluyendo ajustes de tiempo y presupuesto. **2.- Tecnología y Herramientas:** [Evaluación de las herramientas y tecnologías empleadas, con recomendaciones sobre lo que funcionó bien y lo que podría mejorarse]. **3.- Colaboración y Comunicación:** [Observaciones sobre la comunicación interna y colaboración entre los miembros del equipo y con partes interesadas externas. **4.- Solución de Problemas:** [Descripción de los desafíos enfrentados y cómo se resolvieron, proporcionando insights para evitar o manejar situaciones similares en el futuro]. **Conclusión del Documento:** Este documento, al ser compartido y revisado en la reunión final de cierre, ofrece una oportunidad valiosa para que todos los miembros del equipo contribuyan con sus experiencias y aprendizajes, asegurando que los conocimientos adquiridos se aprovechen para mejorar la gestión de proyectos futuros en la organización.

Ejemplo de Uso de Formato (Artefacto) de Cierre:

Documento de Lecciones Aprendidas –
Proyecto "Desarrollo del Sistema de Matrícula Escolar"

Fecha: 12/12/2024
Participantes: Juan Pérez (Gestor del Proyecto), Ana Gómez (Desarrolladora Principal), Carlos Ruiz (Analista de Sistemas), Elena Torres (Desarrolladora), Luis Molina (Analista de Pruebas)
Objetivo del Documento: Proporcionar un recurso valioso que detalle las experiencias, tanto positivas como negativas, surgidas durante la ejecución del proyecto para optimizar la gestión de proyectos futuros.

Contenido:

1.- Planificación y Gestión:

- ✓ Reflexiones: La metodología ágil adoptada permitió una gran flexibilidad y adaptación ante los cambios, pero se observó que las estimaciones iniciales de tiempo y recursos fueron subestimadas, lo que llevó a ajustes de tiempo y presupuesto a mitad del proyecto.
- ✓ Mejoras Sugeridas: Se recomienda realizar una planificación más detallada en las fases iniciales, con estimaciones de tiempo y recursos más realistas y provisiones para contingencias. Aumentar la frecuencia de las reuniones de revisión para ajustar las estimaciones y el alcance del proyecto según sea necesario.

2.- Tecnología y Herramientas:

- ✓ Evaluación: El uso de frameworks modernos como React para el frontend y Node.js para el backend facilitó el desarrollo rápido y efectivo. Sin embargo, la integración inicial con la base de datos SQL presentó desafíos debido a problemas de compatibilidad.
- ✓ Recomendaciones: Continuar utilizando tecnologías modernas y actualizar regularmente las habilidades del equipo en estas herramientas. Para futuros proyectos, realizar pruebas de integración de tecnologías antes de decidir su implementación completa.

3.- Colaboración y Comunicación:

- ✓ Observaciones: La comunicación entre el equipo fue efectiva a través de herramientas como Slack y Zoom. No obstante, se notaron brechas en la comunicación con las partes interesadas externas, lo que ocasionalmente causó desalineaciones con las expectativas.
- ✓ Acciones a Tomar: Mejorar los canales de comunicación con las partes interesadas a través de actualizaciones más regulares y reuniones de revisión de alcance. Implementar un sistema de gestión de expectativas más formal para evitar malentendidos.

4.- Solución de Problemas:

- ✓ Desafíos: Se enfrentaron problemas significativos en la fase de pruebas de integración, donde errores en la interfaz de usuario no detectados previamente causaron retrasos.
- ✓ Insights para el Futuro: Establecer un protocolo más robusto de pruebas unitarias y de integración desde las primeras etapas del desarrollo. Fomentar una cultura de pruebas continuas y feedback inmediato para resolver problemas tan pronto como aparezcan.

Conclusión del Documento:

Este documento, al ser compartido y revisado en la reunión final de cierre, ofrece una oportunidad valiosa para que todos los miembros del equipo contribuyan con sus experiencias y aprendizajes, asegurando que los conocimientos adquiridos se aprovechen para mejorar la gestión de proyectos futuros en la organización.