Profundiza más

Recurso de Profundización

Clase 2: Fases del Desarrollo y Herramientas de Planificación
Autor: Damián Nicolalde Rodríguez

Objetivo del Documento:
Este documento ofrece recursos textuales y gráficos diseñados para ampliar los conceptos abordados en la Clase 2. Se incluyen un cuadro comparativo, una tabla de pasos y un estudio de caso, que facilitan la comprensión de temas fundamentales como las fases del desarrollo de software (análisis, diseño, codificación y prueba) y la utilización de herramientas de planificación (pseudocódigo y diagramas de flujo). Estos recursos permiten apreciar las ventajas y desafíos de cada fase, así como la importancia de estructurar y planificar soluciones antes de implementar código en proyectos de ingeniería en ciberseguridad y gestión de TI.

Recurso de Profundización 1: Cuadro Comparativo – Fases del Desarrollo de Software
Este cuadro comparativo presenta las características principales de cada una de las fases del desarrollo de software: análisis, diseño, codificación y prueba. El recurso permite identificar de manera clara las actividades, objetivos y ventajas de cada etapa, facilitando la comprensión de cómo se integran para generar soluciones robustas y eficientes. Se destacan las ventajas de una planificación detallada en cada fase, así como los posibles desafíos que se pueden enfrentar si alguna etapa no se ejecuta adecuadamente.

Profundiza más

Tabla 1. Cuadro Comparativo: Fases del Desarrollo de Software

Fase	Descripción y Actividades Principales	Ventajas	Ejemplo de Aplicación
Análisis	Identificación del problema, recolección de requerimientos mediante entrevistas, encuestas y revisión documental; definición de objetivos.	Permite comprender a fondo el problema, delimitar el alcance y establecer una base sólida para el proyecto.	Para un sistema de inventario, definir las necesidades de control de stock, registros de ventas y actores involucrados.
Diseño	Planificación de la solución: definición de la arquitectura, diseño de modelos, elaboración de diagramas (casos de uso, clases, datos).	Facilita la visualización de la solución, mejora la organización y prepara el terreno para una codificación ordenada.	Creación de un diagrama de casos de uso que muestre cómo interactúan usuarios y módulos en el sistema de inventario.
Codificación	Traducción del diseño a código fuente mediante un lenguaje de programación, aplicación de buenas prácticas y pruebas unitarias en módulos.	Asegura que el código sea claro, modular y mantenible, lo que reduce la probabilidad de errores y facilita futuras modificaciones.	Desarrollo de funciones específicas para registrar productos y actualizar existencias en el sistema de inventario.
Prueba	Ejecución de pruebas unitarias, de integración, de sistema y de aceptación para	Identifica y corrige errores antes de la implementación en producción, mejorando la	Realización de pruebas de integración para verificar la correcta

	validar que el software cumple con los requerimientos y funciona correctamente.	confiabilidad y seguridad del software.	interacción entre el módulo de inventario y el de reportes.

Damián Nicolalde Rodríguez. (2024).

Recurso de Profundización 2: Tabla de Pasos – Del Requerimiento al Software Funcional

Esta tabla presenta los pasos esenciales que se deben seguir para transformar un requerimiento en un producto de software funcional. Cada paso se explica de manera breve y se acompaña de un ejemplo práctico. El recurso tiene como objetivo guiar a los estudiantes a través de la transición desde la identificación de necesidades hasta la implementación final del sistema, enfatizando la importancia de revisar y optimizar cada etapa del proceso.

Tabla 2. Tabla de Pasos: Del Requerimiento al Software Funcional

Paso	Descripción	Ejemplo
1. Definir el Requerimiento	Identificar y entender el problema a resolver, recopilando toda la información necesaria.	Para un sistema de inventario, identificar la necesidad de registrar y gestionar el stock.
2. Diseñar el Algoritmo	Elaborar un pseudocódigo o diagrama de flujo que detalle la secuencia de operaciones para resolver el problema.	Pseudocódigo: Leer datos, procesar información, calcular resultados y mostrar la salida.
3. Revisar y Optimizar	Analizar el algoritmo en busca de redundancias o ineficiencias y realizar ajustes para mejorar la lógica.	Revisar si se pueden combinar pasos o eliminar operaciones innecesarias en el cálculo.
4. Implementar el Código Fuente	Traducir el algoritmo a un lenguaje de programación, asegurándose de seguir buenas prácticas y estándares.	Código en Python: Definir funciones, escribir bucles y estructuras condicionales para procesar datos.

5. Probar y Depurar	Ejecutar pruebas unitarias y de integración para identificar y corregir errores en cada módulo.	Ejecutar el programa, verificar que el resultado sea correcto y corregir fallos detectados.
6. Documentar el Proceso	Incluir comentarios y documentación que expliquen la función y lógica de cada bloque de código.	Añadir comentarios en el código para explicar cada función y el flujo de datos utilizado.

Damián Nicolalde Rodríguez. (2024).
Recurso de Profundización 3: Estudio de Caso – Diseño y Planificación de un Sistema de Gestión de Inventario
Este estudio de caso presenta de forma detallada el proceso de diseño y planificación de un sistema de gestión de inventario, ejemplificando cómo se integran las fases del desarrollo de software y las herramientas de planificación (pseudocódigo y diagramas de flujo). El estudio se divide en etapas, desde la redacción del pseudocódigo hasta la implementación de un diagrama de flujo que ilustra la lógica del sistema. Se incluyen ejemplos de código, diagramas y tablas resumen que permiten analizar el proceso de forma integral.

Etapas del Estudio de Caso:

1. **Redacción del Pseudocódigo:**
   - **Objetivo:** Definir la lógica del sistema sin preocuparse por la sintaxis de un lenguaje de programación.
   - **Ejemplo:**

```
INICIO
   LEER producto, cantidad, precio_unitario
   CALCULAR valor_total = cantidad * precio_unitario
   REGISTRAR producto y valor_total en inventario
   MOSTRAR confirmación
FIN
```

Este pseudocódigo ilustra los pasos básicos para registrar un producto y calcular su valor total.

2. **Diseño del Diagrama de Flujo:**
   - **Objetivo:** Visualizar la secuencia de operaciones y decisiones que conforman el proceso.

- **Ejemplo (en formato de texto):**
  - Inicio → Entrada de datos (producto, cantidad, precio) → Proceso de cálculo (valor_total) → Registro en inventario → Salida (confirmación) → Fin.
- **Diagrama de Flujo:**

Imagen 1. Diagrama de Flujo Integrativo del Proceso de Desarrollo
Damián Nicolalde Rodríguez. (2024). Diagrama de Flujo Integrativo del Proceso de Desarrollo.

3. **Implementación en Código Fuente:**
   - **Objetivo:** Traducir el diseño a un lenguaje de programación (por ejemplo, Python).
   - **Ejemplo en Python:**

```
def registrar_producto(producto, cantidad, precio_unitario):
    try:
        valor_total = cantidad * precio_unitario
        # Simulación de registro en inventario
        inventario[producto] = valor_total
        return f"Producto {producto} registrado con valor {valor_total}"
    except Exception as e:
        return f"Error en el registro: {str(e)}"

inventario = {}

def main():
    producto = input("Ingrese el nombre del producto: ")
    cantidad = float(input("Ingrese la cantidad: "))
    precio_unitario = float(input("Ingrese el precio unitario: "))
    resultado = registrar_producto(producto, cantidad, precio_unitario)
    print(resultado)

if __name__ == "__main__":
    main()
```

Este código ejemplifica cómo se implementa el pseudocódigo en un programa funcional.

4. **Modularización y Documentación:**

- **Objetivo:** Dividir el proceso en funciones independientes para facilitar la prueba y el mantenimiento.
- Se comenta el código para explicar la función de cada módulo y se organiza de manera que sea fácilmente modificable.

5. **Tabla Resumen del Estudio de Caso:**

Tabla 3. Estudio de Caso – Diseño y Planificación de un Sistema de Gestión de Inventario

Etapa	Objetivo	Resultado Esperado
Redacción del Pseudocódigo	Definir la lógica de registro y cálculo de inventario.	Pseudocódigo claro que detalle cada operación.
Diseño del Diagrama de Flujo	Visualizar el proceso completo y detectar posibles errores.	Diagrama que ilustra la secuencia y bifurcaciones del proceso.
Implementación en Código	Traducir el pseudocódigo a un programa funcional.	Programa en Python que registre productos y calcule valor.
Modularización y Documentación	Facilitar la comprensión, mantenimiento y pruebas.	Código organizado en funciones, con comentarios explicativos.

Damián Nicolalde Rodríguez. (2024).