UT1_6.- DESARROLLO DE SOFTWARE – DESARROLLO DE UNA APLICACIÓN

RESULTADOS DE APRENDIZAJE ASOCIADOS
1.- Reconoce los elementos y herramientas que intervienen en el desarrollo de un programa informático, analizando sus características y las fases en las que actúan hasta llegar a su puesta en funcionamiento.
CRITERIOS DE EVALUACIÓN
a) Se ha reconocido la relación de los programas con los componentes del sistema informático: memoria, procesador, periféricos, entre otros.
b) Se han identificado las fases de desarrollo de una aplicación informática.
c) Se han diferenciado los conceptos de código fuente, objeto y ejecutable.
d) Se han reconocido las características de la generación de código intermedio para su ejecución en máquinas virtuales.
e) Se han clasificado los lenguajes de programación.
f) Se ha evaluado la funcionalidad ofrecida por las herramientas utilizadas en programación.

UT1_6.- DESARROLLO DE SOFTWARE – DESARROLLO DE UNA APLICACIÓN

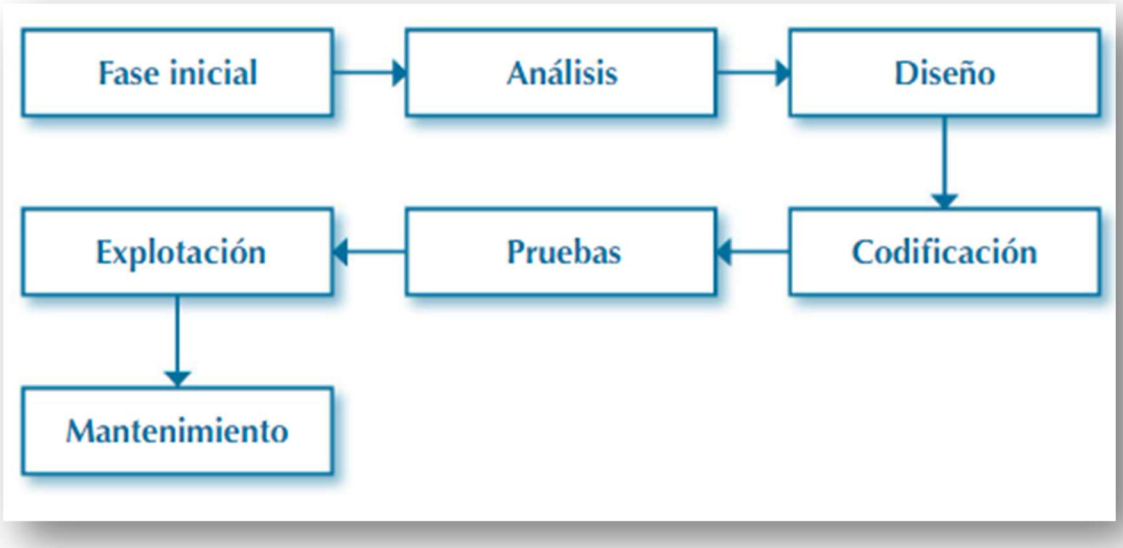
Índice de contenido

1.- Desarrollo de una aplicación 3
1.1.- Fases del desarrollo de una aplicación 3
1.2.- La documentación 12
1.3.- Roles o figuras que forman parte del proceso de desarrollo de software 14
2.- Resumen del tema 15
3.- Referencias bibliográficas 18

1.- Desarrollo de una aplicación

1.1.- Fases del desarrollo de una aplicación

En el ámbito del desarrollo de software, existen numerosos paradigmas y enfoques, aunque en general, comparten una serie de etapas fundamentales. A continuación, se describirán las etapas comunes que se encuentran en cualquier proceso de desarrollo de software, independientemente del ciclo de vida adoptado.



A) FASE INICIAL

Objetivo: Establecer las bases para el éxito del proyecto.

Actividades Clave:

1. **Planificación detallada:** Definir objetivos, recursos y cronograma.

   Ejemplo: En un proyecto de construcción, organizar el trabajo y los materiales.

2. **Estimaciones críticas:** Evaluar costos, plazos y riesgos.

   Ejemplo: En un proyecto de desarrollo de software, calcular costos de desarrollo y mantenimiento.

Requisitos Clave:

Equipo con experiencia en proyectos y tecnologías relevantes.

Ejemplo: Un equipo de ingenieros para proyectos de ingeniería.

Documentación Fundamental:

- Planes generales y detalles específicos.
- Datos económicos detallados.

Ejemplo: Documentos que describen la estructura del proyecto y desglosan los costos en cada etapa.

Importancia: Decisiones en esta fase pueden afectar todo el proyecto.

Ejemplo: Elección temprana de tecnología en un proyecto de software influye en decisiones posteriores.

Conclusión: La fase de planificación y estimación es crítica y requiere precisión, detalle y experiencia para asegurar el éxito del proyecto.

B) ANÁLISIS

Objetivo: Comprender y definir claramente los requisitos del proyecto.

Actividades Clave:

1. **Recopilación de requisitos:** Reunir información y necesidades del cliente.

   Ejemplo: Realizar entrevistas documentadas con el cliente para registrar los requisitos del proyecto.

2. **Examen de restricciones:** Identificar posibles limitaciones y restricciones aplicables.

   Ejemplo: Evaluar las restricciones de tiempo, presupuesto y recursos.

3. **Documentación formal:** Consensuar los documentos de requisitos con el cliente.

   Ejemplo: En algunos proyectos, se firma un acuerdo de requisitos donde el equipo de desarrollo se compromete a seguir las especificaciones del cliente.

Importancia: Esta fase es crucial para establecer una base sólida y comprender las expectativas del cliente.

Documentación Fundamental:

- Documentos de requisitos recopilados en entrevistas.
- Acuerdos contractuales en proyectos donde se requieren compromisos formales.

Conclusión: La fase de análisis es esencial para definir claramente los requisitos del proyecto, asegurando un entendimiento mutuo entre el equipo de desarrollo y el cliente, y estableciendo una base sólida para las etapas posteriores del proyecto.

Pregunta para reflexionar…

¿Por qué se debe documentar de manera oficial todos los requisitos funcionales y restricciones, y posteriormente formalizar su aprobación mediante la firma de ambas partes?

C) DISEÑO

Objetivo: Establecer los requisitos generales de la arquitectura de la aplicación y definir con precisión cada subconjunto de esta.

Actividades Clave:

1. Determinación de requisitos de arquitectura: Identificar los requisitos generales de la estructura de la aplicación.

   Ejemplo: Especificar los componentes esenciales y la interacción entre ellos.

2. Creación de documentos de diseño: Elaborar dos tipos de documentos de diseño.

   - **Documento General:** Proporciona una visión global de la aplicación.
   - **Documento Detallado:** Profundiza en los aspectos técnicos de cada módulo del sistema.

3. Participación de analistas y supervisión del jefe del proyecto en la creación de documentos.

Importancia: Esta fase es esencial para definir la estructura técnica de la aplicación, proporcionando una hoja de ruta clara para el desarrollo y garantizando una comprensión común entre el equipo.

Documentación Fundamental:

Documentos de diseño de arquitectura, tanto generales como detallados.

Conclusión: La fase de Diseño de Arquitectura es crucial para establecer los cimientos técnicos del proyecto, brindando una comprensión completa de la aplicación y asegurando que el desarrollo se realice de manera eficiente y precisa.

D) CODIFICACIÓN O IMPLEMENTACIÓN

Objetivo: Llevar a cabo la implementación del software en un lenguaje de programación para dar vida a las funciones definidas en la etapa de diseño.

Actividades Clave:

1. **Desarrollo de software:** Traducir el diseño en código ejecutable.

   Ejemplo: Escribir el código de acuerdo con las especificaciones de diseño.

2. **Creación de documentación detallada:** Generar documentos que contengan información clave sobre el código.

   - Incluir comentarios en el código para explicar la lógica.
   - Documentar detalles de funciones, como entradas, salidas, parámetros y propósito.
   - Registrar información sobre los responsables, fechas y revisiones realizadas.

Importancia: Esta fase es fundamental para llevar a cabo la implementación precisa del software y proporcionar documentación detallada que facilitará futuras tareas de mantenimiento y comprensión del código.

Documentación Fundamental:

- Código fuente con comentarios.

```
/**
 * Método para activar usuarios
 * @param usuarioEnSession usuario que realiza la acción
 * @param pass contraseña
 * @param passRepetido contraseña repetida
 * @param mail email
 * @param mailRepetido email repetido
 * @param lenguajePreferido  identificador del lenguaje elegido por el usuario
 * @param idAvatar identificador del avatar elegido por el usuario
 * @param borrable S o N indica respectivamente si el usuario se puede borrar o no
 * @return devuelve true en caso de actualizar el usuario y falso en caso contrario
 * @throws TrivinetException
 */
public boolean actualizarUsuario(Usuario usuarioEnSession, String pass,
        String passRepetido, String mail, String mailRepetido,String lenguajePreferido,
        Integer idAvatar, String borrable) throws TrivinetException {
    /* LOG */
    Util.log("actualizarUsuario("+usuarioEnSession+", pass: ***, passRepetido: ***, " +
            ""+mail+", "+mailRepetido+", "+
            lenguajePreferido+", " +
            "idAvatar: "+idAvatar+", borrable: "+borrable+")");
    // Comprobaciones
```

- Documentos detallados de funciones y módulos.

Method Detail

onCreate

```
public void onCreate(Bundle savedInstanceState)
```

---

addIntegers

```
public int addIntegers(int a,
                       int b)
```

Method that adds two integers together

Parameters:
a - The first integer to add
b - The second integer to add
Returns:
The resulting sum of a and b

---

throwException

```
public void throwException(boolean shouldThrow)
                   throws java.lang.Exception
```

This method simply throws an Exception if the incoming parameter a is not a positive number, just for fun.

Parameters:
a - Whether or not to throw an exception
Throws:
java.lang.Exception

Conclusión: La fase de Implementación del Software transforma el diseño en código funcional, asegurando que las especificaciones se materialicen. La documentación minuciosa es esencial para facilitar la comprensión y el mantenimiento a lo largo del ciclo de vida del software.

E) PRUEBAS

Objetivo: Verificar que la aplicación cumple con las especificaciones originales y está lista para su despliegue, evaluando tanto aspectos funcionales como técnicos.

Actividades Clave:

1. **Pruebas Funcionales:**
   - Pruebas con el cliente presente: Asegurar que la aplicación cumple con las especificaciones acordadas con el cliente.
   - Registro detallado de resultados: Anotar fallos y modificaciones, si es necesario, para su revisión con el cliente.
   - Evitar la introducción de nuevas funcionalidades o variaciones en esta etapa.

2. **Pruebas Estructurales:**
   - Pruebas técnicas: Evaluar aspectos técnicos y de rendimiento.
   - Cargas reales y estrés: Someter la aplicación y el sistema a situaciones límite.

Importancia: La fase de pruebas es esencial para garantizar la calidad y el cumplimiento de las especificaciones antes de la puesta en marcha. Las pruebas funcionales involucran al cliente para confirmar la satisfacción de sus requerimientos, mientras que las pruebas estructurales se centran en aspectos técnicos y de rendimiento.

Documentación Fundamental:

- Documento de pruebas funcionales: Detalla resultados y fallos en relación con las especificaciones.
- Documento de pruebas estructurales: Registra los resultados de pruebas técnicas y de rendimiento.

Conclusión: La fase de Pruebas de Software es crucial para validar que la aplicación cumple con los requisitos y se encuentra en condiciones óptimas para su implementación. La documentación detallada de los resultados es esencial para garantizar la conformidad y estabilidad del sistema.

F) EXPLOTACIÓN

Objetivo: Instalar el software en el entorno de uso real y trabajar con él de manera habitual, abordando posibles incidencias y nuevas necesidades que puedan surgir.

Actividades Clave:

1. **Implementación y uso del software**: Instalar y utilizar el software en el entorno real.
   - Lidiar con incidencias y nuevas necesidades: Abordar cualquier problema o requisito adicional que surja durante esta fase.

2. **Documentación detallada de incidencias**: Registrar errores o fallos de manera explícita en un documento.
   - Revisión y resolución de fallos: Los programadores y analistas analizan y solucionan los fallos detectados.

Importancia: Esta fase es fundamental para poner en práctica el software y abordar problemas o requerimientos adicionales que puedan surgir en el entorno de uso real. La documentación detallada de incidencias es esencial para asegurar una resolución eficiente.

Documentación Fundamental:

- Documento de incidencias: Registra errores y fallos, junto con detalles de su resolución.

Conclusión: La fase de Implementación y Uso en Entorno Real representa la etapa donde el software se utiliza en situaciones cotidianas. Es crítica para identificar y resolver problemas, así como para atender nuevas necesidades que puedan surgir durante la operación real del software. La documentación precisa es clave para la eficiencia en la resolución de incidencias.

G) MANTENIMIENTO

Objetivo: Realizar procedimientos correctivos para solucionar fallos y llevar a cabo actualizaciones secundarias del software, adaptándolo y mejorándolo con el tiempo.

Actividades Clave:

1. **Procedimientos Correctivos:**
   - Corrección de fallos: Identificar y solucionar errores en el software.

2. **Mantenimiento Continuo:**
   - Actualizaciones secundarias: Realizar mejoras y adaptaciones para evolucionar la aplicación con base en las necesidades cambiantes.

3. **Uso de Documentación Técnica:**
   - Consulta de documentación técnica: La documentación es esencial para llevar a cabo tareas de mantenimiento efectivas.

4. **Documentación de Operaciones de Mantenimiento:**
   - Registro de operaciones: Documentar quién realizó la operación, qué se hizo y cómo se realizó.
   - Pruebas por terceros: Otra persona distinta al programador debería verificar y validar las operaciones de mantenimiento.

Importancia: El mantenimiento del software es crítico para garantizar la corrección de errores y la evolución del software a lo largo del tiempo. La documentación técnica y el registro de operaciones son esenciales para la eficiencia y la calidad del mantenimiento.

Documentación Fundamental:

- Documentación técnica de la aplicación.
- Registro de operaciones de mantenimiento.

Conclusión: La fase de Mantenimiento del Software asegura la funcionalidad continua y la mejora del software con el tiempo. La documentación técnica y el registro detallado de operaciones son cruciales para mantener la calidad y la trazabilidad del mantenimiento.

1.2.- La documentación

El proceso de documentación es fundamental en cada una de las etapas previas del proyecto, y su importancia no debe subestimarse en ningún sentido.

La documentación adecuada no solo es esencial para el éxito del proyecto, sino que también es una herramienta valiosa para los usuarios y técnicos involucrados.

En este contexto, es imperativo generar una serie de documentos clave que sirvan como guía y referencia. A continuación, se describen estos documentos esenciales, junto con ejemplos que ilustran su importancia:

1. Manual de Usuario: Facilitando la Experiencia del Usuario

Objetivo: Proporcionar a los usuarios una guía clara y auto explicativa para utilizar la aplicación de manera efectiva.

***Ejemplo*:** En el desarrollo de un software de edición de imágenes, el Manual de Usuario incluiría instrucciones detalladas sobre cómo cargar imágenes, aplicar filtros y guardar el trabajo final. Ilustraría cada paso con capturas de pantalla y ejemplos prácticos.

2. Manual Técnico: Facilitando la Comprensión Técnica

Objetivo: Dirigido a técnicos y desarrolladores, este manual brinda una visión profunda de la estructura y funcionamiento interno de la aplicación.

Ejemplo: En la creación de una plataforma web, el Manual Técnico describiría la arquitectura de la base de datos, los lenguajes de programación utilizados y cómo se gestionan las solicitudes de los usuarios. Esto permitiría a otros técnicos comprender y mantener el sistema de manera efectiva.

3. Manual de Instalación: Garantizando una Implementación sin Problemas

Objetivo: Proporcionar instrucciones detalladas sobre los requisitos y pasos necesarios para instalar y poner en funcionamiento la aplicación de manera adecuada.

Ejemplo: En el despliegue de un software empresarial, el Manual de Instalación explicaría los requisitos del sistema, cómo configurar la base de datos y cómo realizar la instalación en servidores específicos. Esto aseguraría una implementación exitosa y sin problemas.

En resumen, la documentación en cada proyecto es esencial para garantizar la comprensión y el funcionamiento efectivo de la aplicación.

Ignorar la documentación adecuada puede resultar en confusión y dificultades en la implementación y el uso de la aplicación.

1.3.- Roles o figuras que forman parte del proceso de desarrollo de software

El equipo de desarrollo de software desempeña un papel crucial en la ejecución exitosa de un proyecto, y está conformado por diversos roles, cada uno con atribuciones y responsabilidades específicas. A continuación, se detallan los roles principales en un proyecto de desarrollo de software, junto con ejemplos que ilustran sus funciones y responsabilidades:

A) Arquitecto de Software:

Responsabilidad: Decidir la estructura y los recursos para la realización del proyecto, incluyendo tecnologías, *frameworks* y librerías.

Ejemplo: En el desarrollo de un sistema de gestión de inventario, el Arquitecto de Software elige el lenguaje de programación, la base de datos y define cómo se integrarán los componentes del sistema.

B) Jefe de Proyecto:

Responsabilidad: Dirigir el proyecto, gestionar el equipo, administrar los plazos y mantener una comunicación efectiva con el cliente.

Ejemplo: Un Jefe de Proyecto en un proyecto de desarrollo de una aplicación móvil se asegura de que el equipo esté cumpliendo los hitos y de que las expectativas del cliente se cumplan.

C) Analista de Sistemas:

Responsabilidad: Realizar un análisis exhaustivo del problema a resolver y diseñar el sistema en su totalidad.

Ejemplo: En el desarrollo de un sistema de reservas de vuelos, el Analista de Sistemas identifica los requisitos, los flujos de trabajo y las funcionalidades clave del sistema.

D) Analista Programador:

Responsabilidad: Este rol está a medio camino entre el analista y el programador. Realiza análisis de requerimientos y también codifica.

Ejemplo: En un proyecto de desarrollo de una plataforma de comercio electrónico, el Analista Programador trabaja en la definición de funcionalidades y luego implementa estas funcionalidades en código.

E) Programador:

Responsabilidad: Codificar las tareas asignadas por el analista o el analista-programador, teniendo un conocimiento profundo del lenguaje de programación.

Ejemplo: En el desarrollo de un videojuego, el Programador se encarga de escribir el código que controla el comportamiento de los personajes y las interacciones del juego.

Cada uno de estos roles desempeña una función esencial en el proceso de desarrollo de software, contribuyendo a la creación de soluciones eficientes y efectivas que cumplan con los requisitos del cliente y los estándares de calidad. La colaboración y la comunicación efectiva entre estos roles son fundamentales para el éxito del proyecto.

2.- Resumen del tema

A continuación, se muestra un resumen del tema para que su estudio sea más sencillo:

- El software es la parte intangible de un sistema informático, el equivalente al equipamiento o soporte lógico.
- Características del software:
  - El software es lógico, no físico. Es intangible.
  - El software se desarrolla, no se fabrica.
  - El software no se estropea y una copia suya da lugar a un clon.
  - En ocasiones, puede construirse a medida.
- Un programa es una serie de órdenes o instrucciones secuenciadas u ordenadas con una finalidad concreta que realizan una función determinada.
- Un software a medida es una o varias aplicaciones realizadas según los requerimientos e instrucciones de una empresa u organismo.
- El software estándar o enlatado es aquel de tipo genérico que resuelve múltiples necesidades.
- Los lenguajes de programación son, por lo tanto, un lenguaje artificial creado para que, al traducirse a código máquina, cada una de las instrucciones de dicho lenguaje den lugar a una o varias instrucciones máquina.
- El lenguaje máquina es ininteligible y se compone de combinaciones de unos y ceros. No necesita ser traducido, por lo tanto, es el único lenguaje que entiende directamente el ordenador. Fue el primer lenguaje utilizado y difiere para cada procesador.

- El lenguaje de medio nivel o ensamblador sustituyó al lenguaje máquina para facilitar la labor de programación. Sigue estando cercano al hardware, pero, en lugar de unos y ceros, se programa usando mnemotécnicos. Trabaja con los registros del procesador, direcciones físicas y es difícil de comprender y programar.
- Los lenguajes de alto nivel tienen una forma de programar más intuitiva y sencilla. Son más cercanos al lenguaje humano que al lenguaje máquina.
- Los lenguajes pueden clasificarse en:
  - Lenguajes compilados.
  - Lenguajes interpretados.
  - Lenguajes virtuales.
- Los traductores son programas cuya finalidad es traducir lenguajes de alto nivel a lenguajes de bajo nivel como ensamblador o código máquina.
- Existen dos grandes grupos de tipos de traductores:
  - *Compiladores:* traducen el código fuente a código máquina.
  - *Intérpretes:* traducen el código fuente línea a línea.
- Los estados de un programa son los siguientes:
  - *Código fuente*. Código escrito en un lenguaje de programación.
  - *Código objeto*. Resultado de compilar el código fuente.
  - *Código ejecutable*. Resultado de compilar y enlazar el código con las librerías.

- Las fases clásicas del desarrollo de una aplicación son las siguientes:
  - Fase inicial.
  - Análisis.
  - Diseño.
  - Codificación.
  - Pruebas.
  - Explotación.
  - Mantenimiento.
- En cualquier aplicación, como mínimo, deberán generarse los siguientes documentos:
  - Manual de usuario.
  - Manual técnico.
  - Manual de instalación.
- Los roles en un proyecto software son:
  - Arquitecto de software.
  - Jefe de proyecto.
  - Analista de sistemas.
  - Analista programador.
  - Programador.

ACTIVIDAD 6.1

Realiza un listado de tareas y asígnaselas a los distintos roles de un desarrollo de software.

Algunas tareas podrían ser:

- Cargar la base de datos con datos de prueba
- Entrevista con el cliente para establecer requisitos
- Elección de las herramientas de desarrollo.
- Elección del lenguaje de desarrollo
- Pruebas unitarias
- Pruebas finales
- Crear la estructura de la base de datos
- Establecer los requisitos del proyecto
- Entrega de la primera versión del proyecto
- Documentación técnica
- Documentación para el usuario

Establece en qué paso del desarrollo software va a realizarse dicha tarea, con lo que cual lo más eficiente es que elabores un cuadro con las siguientes columnas: tarea, paso de desarrollo y rol que ejecuta.

3.- Referencias bibliográficas

- Moreno Pérez, J.C. *Entornos de desarrollo.* Editorial Síntesis.
- Ramos Martín, A. & Ramos Martín, M.J. *Entornos de desarrollo*. Grupo editorial Garceta.