ENTORNOS DE DESARROLO

UNIDAD 1

Introducción al Software y los Programas Informáticos

¿Qué es un programa informático?

Un programa informático, según el estándar IEEE/ISO/IEC 24765:2017, es:

1. Un conjunto de instrucciones y datos para realizar tareas específicas.
2. Una solución a problemas mediante un lenguaje de programación.

¿Qué es el software?

El software incluye:

- Programas: Instrucciones para que la computadora realice acciones.
- Procedimientos: Métodos necesarios para ejecutar los programas.
- Documentación: Guías y manuales de uso.

No olvidemos a las personas que usan y desarrollan estos sistemas, conocidas como **peopleware**.

Tipos de Software

Software de Sistema

- Gestiona los recursos básicos de la computadora.
- Sistemas operativos: Windows, macOS, Linux.
- Utilidades: Programas de limpieza de disco, antivirus.
- Drivers: Controladores para impresoras o tarjetas gráficas.

Software de Aplicación

- Ayuda a realizar tareas específicas.
- Microsoft Word para escribir documentos.
- Photoshop para edición de imágenes.
- GIMP y LibreOffice como alternativas gratuitas de código abierto.

Software de Soporte

- Herramientas utilizadas por desarrolladores para crear otros programas
- Compiladores: Transforman código fuente en código máquina (GCC).

- Intérpretes: Ejecutan código línea por línea (Python).
- Editores de texto: Visual Studio Code, Sublime Text.

TEST

1. ¿Qué es un programa informático?

b) Instrucciones y datos que permiten realizar tareas específicas.

2. ¿Cuál de estos NO es un tipo de software?

c) Software de hardware.

3. ¿Qué herramienta usarías para desarrollar en Python?

a) Visual Studio Code.

Lenguajes de Programación

Las máquinas solo entienden ceros y unos, también conocido como lenguaje binario. Para salvar esta brecha, los programadores crearon lenguajes más cercanos a la lógica humana, que permiten traducir nuestras ideas al formato que las computadoras entienden.

La programación, entonces, es el arte de traducir soluciones de problemas en instrucciones que una máquina puede ejecutar.

¿Qué es un Lenguaje de Programación?

Según el estándar IEEE/ISO/IEC 24765:2017, un lenguaje de programación es un conjunto de reglas y palabras que los programadores usan para crear instrucciones comprensibles por una computadora.

Diferencia entre “Lenguaje de Programación” y “Lenguaje Informático”

- Lenguaje de Programación: Diseñado para escribir programas que ejecuten tareas específicas. Ejemplos: Python, Java.
- Lenguaje Informático: Más amplio, incluye lenguajes para estructurar o transportar datos. Ejemplos: XML, YAML.

Clasificación de los Lenguajes de Programación

Por Nivel de Abstracción

Bajo Nivel

- Muy cercano al hardware. Ejemplo: Ensamblador.
- Ventaja: Control total del hardware.

- Desventaja: Difícil de aprender.
- Usos: Sistemas embebidos, controladores.

Nivel Intermedio

- Mezcla características de alto y bajo nivel. Ejemplo: C.
- Ventaja: Equilibrio entre control y facilidad.
- Usos: Sistemas operativos, software de alto rendimiento

Alto Nivel

- Similar al lenguaje humano. Ejemplo: Python
- Ventaja: Fáciles de aprender.
- Desventaja: Menor control sobre el hardware.
- Usos: Aplicaciones web, software científico.

Por Propósito

Propósito General:

- Ejemplo:

› Python: Análisis de datos, desarrollo web.

› JavaScript: Interactividad web.

› Java: Aplicaciones empresariales.

Propósito Específico:

- Ejemplo:

› R: Estadística y análisis de datos.

› SQL: Gestión de bases de datos.

Por Evolución Histórica

1. 1GL (Primera Generación): Código máquina.
- Ejemplo: Programación de microcontroladores.
2. 2GL (Segunda Generación): Ensamblador.
- Ejemplo: Sistemas embebidos.
3. 3GL (Tercera Generación): Lenguajes modernos.
- Ejemplo: C++, Java.

4. 4GL (Cuarta Generación): Lenguajes para grandes volúmenes de datos.

- Ejemplo: SQL.

5. 5GL (Quinta Generación): Resolución de problemas complejos.

- Ejemplo: Prolog.

Por Forma de Ejecución

1. Compilados: Traducen todo el código antes de ejecutarse. Ejemplo: C.
2. Interpretados: Ejecutan el código línea por línea. Ejemplo: Python.
3. Mixtos: Compilan a un bytecode que luego es interpretado. Ejemplo: Java.

Por Estilo o Paradigma de Programación

1. Imperativa:

- Procedimental: Divide problemas en subprogramas. Ejemplo: C.
- Orientada a objetos: Usa objetos y clases. Ejemplo: Java.

2. Declarativa:

- Funcional: Usa funciones puras. Ejemplo: Haskell.
- Lógica: Usa reglas y hechos. Ejemplo: Prolog

Por Lugar de Ejecución

1. Frontend: Ejecutado en el navegador. Ejemplo: JavaScript.
2. Backend: Ejecutado en el servidor. Ejemplo: PHP.
3. Full-Stack: Combina frontend y backend.

TEST

1. ¿Qué tipo de lenguaje es Ensamblador?

a) Bajo nivel

2. ¿Cuál es un lenguaje interpretado?

c) Python

3. ¿Qué paradigma utiliza objetos y clases?

d) Orientado a objetos

Traductores, Compiladores e Intérpretes

Traductores

Los traductores son programas que convierten lenguajes de programación de alto nivel en código máquina, haciéndolo comprensible para el hardware.

Compiladores

- Función: Traducen el programa completo antes de su ejecución.
- Ejemplos: C, C++.
- Ventajas:
- Detectan errores antes de la ejecución.
- El ejecutable resultante es más rápido al ejecutarse.
- Desventajas:
- El proceso de compilación puede ser lento.

Intérpretes

- Función: Traducción y ejecución línea por línea del programa.
- Ejemplos: Python, JavaScript.
- Ventajas:
- Ejecución más rápida durante el desarrollo.
- Ideal para prototipado y pruebas rápidas.
- Desventajas:
- Más lentos en ejecución que los compiladores.
- Los errores se detectan durante la ejecución.

Código Fuente, Código Objeto y Código Ejecutable

Código Fuente

- Definición: Instrucciones escritas por un programador en un lenguaje de alto nivel.
- Ejemplo: Un archivo con extensión .java para Java o .py para Python.

Código Objeto

- Definición: El resultado de traducir el código fuente a un formato más cercano al lenguaje máquina, pero aún no ejecutable.
- Ejemplo: Archivos .o o .obj.

Código Ejecutable

- Definición: El código final que puede ser ejecutado directamente por la máquina.
- Ejemplo: Archivos .exe en Windows o binarios en Linux.

Proceso de Compilación

1. Edición: Creación del código fuente utilizando un editor de texto o IDE.
2. Traducción: El compilador transforma el código fuente en código objeto.
3. Enlazado: Combina los archivos de código objeto con bibliotecas necesarias para generar el ejecutable.

Tecnologías de Virtualización: Java y .NET

Máquinas Virtuales

- Definición: Programas que permiten la ejecución de código en un entorno independiente del hardware y sistema operativo subyacentes.

Tipos de Máquinas Virtuales

1. Máquinas Virtuales de Sistema:
- Simulan un hardware completo.
- Ejemplo: VMware, VirtualBox.
2. Máquinas Virtuales de Proceso:
- Ejecutan aplicaciones en un entorno independiente del hardware.
- Ejemplo: Java Virtual Machine (JVM), Common Language Runtime (CLR).

1. Java:
- Traduce el código fuente a bytecode.
- La JVM ejecuta el bytecode en cualquier sistema operativo.

2. .NET:
- Compila el código fuente a Common Intermediate Language (CIL). • El CLR se encarga de ejecutarlo.

El Concepto de Multiplataforma

El software multiplataforma es aquel que puede ejecutarse en distintas combinaciones de hardware y sistemas operativos.

Ejemplos de Aplicaciones Multiplataforma

- LibreOffice: Alternativa gratuita a Microsoft Office.
- Mozilla Firefox: Navegador web.
- GIMP: Editor gráfico.

1. Software Libre:
    - Permite a los usuarios ejecutar, modificar y redistribuir el código.
    - Ejemplo: Linux.
    2. Software Propietario:
    - Solo se distribuye el ejecutable.
    - Ejemplo: Microsoft Windows.

TEST

1. ¿Qué hace un compilador?

b) Traduce todo el código antes de ejecutarlo.

2. ¿Qué extensión corresponde a un código objeto?

c) .o

3. ¿Qué tecnología utiliza bytecode?

b) JVM

Desarrollo de Aplicaciones

Fases en el Desarrollo de una Aplicación

Análisis

En esta fase se identifican y documentan los requisitos funcionales y no funcionales. Es como hacer una lista de compras detallada antes de empezar a cocinar.

Herramientas comunes:

- Diagramas de flujo de datos: Representan cómo se mueve y procesa la información
- Diagramas de casos de uso y de secuencia: Muestran las interacciones entre los usuarios y el sistema
- Diagramas entidad-relación: Representan la estructura de la base de datos.
- Wireframes y prototipos: Representaciones visuales de la interfaz de usuario

Diseño

Se define cómo será la aplicación. Es como planear una receta y los pasos a seguir.

- Arquitectura del sistema: Describe la estructura del sistema y las interacciones entre componentes.
- Diseño de la base de datos: Define cómo se organizarán y conectarán los datos.
- Diseño de la interfaz de usuario: Cómo interactuarán los usuarios con el sistema.

Codificación

Es la fase en la que los desarrolladores traducen el diseño en código fuente usando lenguajes de programación adecuados.

Pruebas

Se verifican y validan las funcionalidades del sistema.

- Unitarias: Prueban componentes individuales.
- De integración: Evalúan cómo interactúan diferentes partes del sistema.
- De aceptación: Aseguran que el sistema cumple con los requisitos del usuario.

Implantación

Se pone el sistema en producción para que los usuarios finales puedan utilizarlo.

Mantenimiento

Es una fase continua para garantizar que el sistema siga funcionando correctamente y cumpla con las necesidades cambiantes.

- Correctivo: Corregir errores.
- Adaptativo: Ajustar a nuevas tecnologías.
- Perfectivo: Añadir funcionalidades.

Modelos de Desarrollo de Software

Ciclo de Vida en Cascada

Un enfoque secuencial donde cada fase debe completarse antes de pasar a la siguiente.

- Ventajas:
- Proceso estructurado y controlado.

- Puntos de control claros.
- Desventajas:
- Poco flexible ante cambios.
- No permite retroalimentación temprana.

Modelos de Desarrollo Evolutivo

Más flexibles y centrados en iteraciones continuas.

- Prototipado:
- Se crea un prototipo para validar requisitos antes del desarrollo completo.
- Ventajas: Minimiza errores en etapas tempranas.
- Iterativo e Incremental:
- Desarrolla y mejora el sistema en ciclos continuos.
- Ventajas: Permite entregas parciales con funcionalidades.

Modelo en Espiral

Iterativo, con énfasis en la gestión de riesgos.

- Ventajas: Adecuado para proyectos complejos con requisitos cambiantes.

TEST

1¿Cuál de las siguientes no es una fase del ciclo de vida del software?

c) Finanzas

2. ¿Qué modelo de desarrollo es más adecuado para proyectos con requisitos cambiantes?

d) Modelo evolutivo

3. ¿Qué tipo de mantenimiento implica añadir nuevas funcionalidades?

b) Perfectivo

Metodologías Ágiles en el Desarrollo de Software

Surgieron como respuesta a esta necesidad, ofreciendo ciclos cortos de desarrollo, entregas frecuentes y una comunicación continua con el cliente.

Scrum

Scrum es una de las metodologías ágiles más utilizadas. Organiza los proyectos en sprints, intervalos de tiempo definidos (2-4 semanas), en los que el equipo debe entregar un incremento funcional del producto.

Organización del Equipo

Los equipos en Scrum suelen ser pequeños, con roles bien definidos:

1. Scrum Master: Facilita el proceso de Scrum, elimina impedimentos y asegura que el equipo siga las prácticas ágiles.

2. Product Owner: Representa al cliente, prioriza los requisitos en el Product Backlog y asegura que el equipo esté alineado con los objetivos.

3. Equipo de Desarrollo: Implementa las funcionalidades del producto.

- Desarrolladores: Escriben el código.
- QA: Aseguran la calidad mediante pruebas.

Reuniones en Scrum

1. Sprint Planning:

- Se seleccionan tareas del Product Backlog y se crean los objetivos del sprint.
- Duración: 4 horas.

2. Daily Scrum:

- Reunión breve para discutir el progreso diario y los obstáculos.
- Duración: 15 minutos.

3. Sprint Review:

- Se presenta el producto al cliente y se recopila feedback.
- Duración: 2 horas.

4. Sprint Retrospective:

- Análisis del sprint, identificando áreas de mejora.
- Duración: 1.5 horas.

Otros Marcos y Metodologías Ágiles

Kanban

- Descripción: Método visual para gestionar el trabajo en progreso mediante un tablero con columnas (por hacer, en progreso, hecho).

- Ventajas:

- Visibilidad del flujo de trabajo.
- Alta flexibilidad.
- Desventajas:
- Menos estructurado para proyectos grandes.

Extreme Programming (XP)

- Descripción: Enfocado en mejorar la calidad del software mediante prácticas como programación en parejas y revisión continua.
- Ventajas:
- Alta calidad del código.
- Rápida respuesta a cambios.
- Desventajas:
- Requiere alta colaboración y disciplina.

Lean Development

- Descripción: Basado en principios de manufactura lean, busca maximizar el valor al cliente reduciendo desperdicios.
- Ventajas:
- Entrega rápida de valor.
- Eficiencia en el uso de recursos.
- Desventajas:
- Difícil implementación en equipos grandes

TEST

1. ¿Cuál de las siguientes es una característica de Scrum?

b) Sprints con objetivos claros.

2. ¿Qué metodología utiliza un tablero para gestionar tareas?

c) Kanban

3. ¿Cuál es una práctica común en Extreme Programming (XP)?

b) Programación en parejas.