1. Software y programa. Tipos de software.

- Partes del ordenador: hardware y software.
- Software: conjunto de programas informáticos que actúan sobre el hardware para ejecutar lo que el usuario desea. Tres tipos:
  - Sistema Operativo: software base del ordenador para que las aplicaciones funcionen.
  - De programación: conjunto de herramientas que nos permiten desarrollar programas informáticos.
  - Aplicaciones informáticas: conjunto de programas con una finalidad.

2. Relación hardware-software

- Dos puntos de vista desde la arquitectura de John Von-Neumann:
  - Desde el punto de vista del sistema operativo: coordina al hardware mientras el ordenador funciona, actuando como intermediario entre este y las aplicaciones. Todas las aplicaciones necesitan recursos; el SO se encarga de esto.
  - Desde el punto de vista de las aplicaciones: hay muchos lenguajes de programación, que tienen algo en común: todos se pueden interpretar por el ser humano, aunque el hardware solo entiende señales eléctricas.

3. Desarrollo de software

- Desarrollo de software: proceso que ocurre desde que se concibe una idea hasta que un programa está implementado en el ordenador y funcionando.
  - Es una serie de pasos obligatorios para hacer que los programas sean eficientes, fiables, seguros y responden a las necesidades.

4. Lenguajes de programación

- Lenguaje de programación: idioma creado de forma artificial, formado por símbolos y normas sobre un alfabeto, para que el hardware lo entienda y ejecute. Son los instrumentos para que el ordenador realice las tareas que queremos.
- Hay muchos lenguajes de programación, cada uno con sus propios símbolo y estructuras. Cada uno está centrado en una programación de tareas o áreas determinadas.

Características de los Lenguajes de Programación

- Lenguaje máquina:
  - Instrucciones: combinaciones de unos y ceros.
  - Único lenguaje que entiende directamente el ordenador.
  - Primer lenguaje utilizado.
  - Único para cada procesador.
  - Nadie programa en ese lenguaje.
- Lenguaje ensamblador:
  - Sustituyó al lenguaje máquina.
  - Utiliza mnemotécnicos.
  - Necesita traducción al lenguaje máquina.
  - Instrucciones: sentencias que hacen referencia a la ubicación física de los archivos.
  - Difícil de utilizar.
- Lenguaje de alto nivel basados en código:
  - Sustituyó al lenguaje ensamblador.
  - Utiliza sentencias y órdenes derivadas del inglés.
  - Necesita traducción al lenguaje máquina.
  - Los más utilizados hoy en día
- Lenguajes visuales:
  - Se programa gráficamente usando el ratón y diseñando en la apariencia del software.
  - El código se genera automáticamente.
  - Necesitan traducción al lenguaje máquina.
  - Son portables de un equipo a otro.

4.1. Concepto y características

- CONCEPTO. Un lenguaje de programación es el conjunto de:
  - Alfabeto: conjunto de símbolos permitidos.
  - Sintaxis: normas de construcción permitidas de los símbolos.
  - Semántica: significado de las construcciones.
- CARACTERÍSTICAS. Podemos clasificarlos en:
  - Según lo cerca que estén del lenguaje humano: de alto nivel y de bajo nivel.
  - Según la técnica de programación utilizada: estructurados, orientada a objetos o visuales.

4.2. Lenguajes de programación estructurados

- Técnica para escribir lenguajes de programación que solo permite tres tipos de sentencias: secuenciales, condicionales, o bucles. Reciben el nombre de lenguajes estructurados. Ventajas:
  - Programas fáciles de leer.
  - Mantenimiento sencillo.
  - Estructura sencilla y clara
- Inconvenientes:
  - Todo el programa se encuentra en un único bloque.
  - No permite la reutilización eficaz del código.

4.3. Lenguajes de programación orientados a objetos

- Tratan los programas como un conjunto de objetos que colaboran para realizar acciones.
- Principal desventaja: no es una programación tan intuitiva como la estructurada. Ventajas:
  - Código reutilizable.
  - Errores más fáciles de localizar y depurar.
- Características:
  - Los objetos tienen una serie de atributos que los diferencian.
  - Clase: colección de objetos con características similares.
  - Llamamos a los objetos mediante métodos.
  - Objetos: unidades individuales e indivisibles. Base de este tipo de programación.

5. Fases en el desarrollo y ejecución del software

- Siempre hay que seguir unas etapas para construir software fiable y de calidad:
  1. Análisis: se especifican los requisitos funcionales y no funcionales.
  2. Diseño: se divide el sistema en partes y se determina la función de cada una.
  3. Codificación: se elige lenguaje de programación y se codifica el programa.
  4. Pruebas: se prueban los programas en busca de errores y se depuran.
  5. Documentación: se documenta y guarda información de todas las etapas:
  6. Explotación: se prueba el programa en equipos cliente.
  7. Mantenimiento: se mantiene el contacto para actualizar y modificar la aplicación.

5.1. Análisis

- Fase de mayor importancia. También la más complicada, ya que depende en gran parte del analista.
- En esta fase se especifican los requisitos funcionales y no funcionales:
  - Funcionales: funciones a realizar por la aplicación.
  - No funcionales: tiempo de respuesta, legislación …

5.2. Diseño

- Sistema dividido en partes y establecemos relaciones entre ellas. Que hará cada parte. Decisiones a tomar:
  - Entidades y relaciones de las BB.DD.
  - Selección de SGBD.
  - Selección de los lenguajes de programación.
  - Definición de diagrama de clases.

5.3. Codificación. Tipos de código

- Realización el proceso de programación. Elegimos el lenguaje, codificar la información anterior y llevarlo a código fuente. El código pasa por diferentes estados:
  - Código fuente: escrito por programadores. Escrito en lenguaje de alto nivel y contiene las instrucciones necesarias.
  - Código objeto: código binario resultado de compilar el código fuente.
  - Código ejecutable: código binario resultante de enlazar los archivos de código objeto con ciertas bibliotecas.

5.4. Fases en la obtención de código

5.4.1. Fuente

- Conjunto de instrucciones que la máquina debe realizar. No es directamente ejecutable por la máquina.
- Debemos realizar un algoritmo antes de la elaboración, definido como conjunto de pasos para la solución del problema. Para obtenerlo:
  1. Partir etapas anteriores de análisis y diseño.
  2. Se diseñará algoritmo con los pasos a seguir para la solución.
  3. Se elegirá lenguaje de programación.
  4. Codificación del algoritmo ya diseñado.

5.4.2. Objeto

- Es un código intermedio. Bytecode distribuido en varios archivos, correspondiente a cada programa fuente compilado.
- Se puede realizar de dos formas:
  1. Compilación: proceso de traducción sobre el código fuente en un solo paso. Realizado por un compilador.
  2. Interpretación: proceso de traducción línea a línea. Realizado por un intérprete.

5.4.3. Ejecutable

- Resultado de enlazar los archivos de código objeto. Un único archivo ejecutado por la computadora. No necesita aplicación externa. Controlado por el sistema operativo.
- Para enlazar los archivos, se utiliza un software llamado linker para obtener un único archivo.
- Esquema de generación de código ejecutable:
  1. Escribimos el lenguaje fuente con un Lenguaje de Programación.
  2. Compilamos el código fuente para obtener el código objeto o bytecode.
  3. El bytecode, a partir de una máquina virtual, para a código máquina, ejecutable por la computadora.

5.5. Máquinas virtuales

- Tipo especial de software para separar el funcionamiento del ordenador de los componentes hardware.
- Con ella podremos ejecutar aplicación en cualquier equipo sin tener en cuenta el hardware. Garantiza la portabilidad.
- Funciones principales:
  - Hacer aplicaciones portables.
  - Reservar memoria para objetos creados y liberar la no utilizada.
  - Comunicarse con el host para controlar el hardware.
  - Cumplir las normas de seguridad.
- Características:
  - Aislar la aplicación del hardware de la máquina, así conseguimos ejecutar el bytecode en cualquier máquina.
  - Verificar el bytecode siempre antes de ejecutarlo.
  - Proteger las direcciones de memoria.

5.5.1. Frameworks

- Estructura de ayuda al programador. Nos permite empezar proyectos sin empezar desde 0.
- Plataforma software con programas soporte, bibliotecas… que ayudan a desarrollar y unir los diferentes módulos. Ventajas:
  - Desarrollo rápido de software.
  - Reutilización de partes del código.
  - Diseño uniforme del software.
  - Portabilidad.
- Inconvenientes:
  - Gran dependencia del código respecto al framework.
  - La instalación e implementación requiere de muchos recursos.

5.5.2. Entornos de ejecución

- Servicio de máquina virtual que sirve como base software para la ejecución de un programa. Se encargan de:
  - Configurar la memoria principal disponible en el sistema.
  - Enlazar archivos con bibliotecas ya existentes y subprogramas creados.
  - Depurar programas: comprobar errores semánticos del lenguaje.
- Funcionamiento:
  - Formado por máquina virtual y API's.
  - Funciona como intermediado entre lenguaje fuente y sistema operativo.
  - No es suficiente para crear nuevas aplicaciones.
  - Es necesario para desarrollar aplicaciones.

5.5.3. Java runtime environment

- Concepto: el JRE es un conjunto de utilidades que permite la ejecución de programas java en cualquier plataforma.
- Componentes:
  - Una máquina virtual JAVA, que interpreta el código de la aplicación.
  - Bibliotecas de clase estándar que implementan el API de Java.
  - Las dos conjuntas, siendo consistentes entre sí.

5.6. Pruebas

- Realizadas una vez obtengamos el software. Realizadas sobre unos datos de prueba. Tipos de pruebas:
  - Unitarias: probar una a una las diferentes partes del software y comprobar su funcionamiento.
  - Integración: realizadas una vez las unitarias hayan tenido éxito, y comprueban el funcionamiento del sistema completo
- Ultima prueba: Beta Test, realizada en el entorno de producción donde será utilizado por el cliente.

5.7. Documentación

- Necesaria para dar información a los usuarios y hacer futuras revisiones.
- Debemos documentarlo en todas las fases, para pasar de una a otra de forma clara y definida.
- Un buena nos permite reutilizar parte de los programas para otras aplicaciones.
- Debemos de elaborar los siguientes documentos: guía técnica, guía de uso y guía de instalación.

5.8. Explotación

- Una vez tengamos certeza de que el software es fiable, carece de errores y todo este documentado, llega este paso.
- Es la instalación, puesta a punto y funcionamiento de la aplicación en el equipo final del cliente.
- Fase de instalación: programas son instalados en la máquina del usuario.
- Fase de configuración: asignamos parámetros de configuración normales y probamos que la aplicación es operativa.
- Fase de producción normal: la aplicación se pasa a los usuarios finales.

5.9. Mantenimiento

- Debemos actualizar el software en base a las mejores del hardware y afrontar nuevas situaciones.
- Siempre surgirán más errores que habrá que ir corrigiendo y nuevas versiones. Tipos de cambios que se pueden realizar:
  - Perfectivos: mejoran funcionalidad del software.
  - Evolutivos: para las nuevas necesidades que vaya teniendo el cliente.
  - Adaptativos: para adaptarse a las nuevas tendencias, nuevo hardware…
  - Correctivos: para corregir errores.