Actividades UF1_1:

1a Part:

1. Elabora los apuntes de esta Unidad y guárdalos en GitHub en un repositorio llamado 'Apuntes_UF1_1'.

2. Haz una clasificación del software.

Software de aplicación: Elementos que permiten el mantenimiento del sistema: sistemas operativos, controladores de dispositivos, servidores, utilidades, herramientas de diagnóstico, de corrección y optimización.

Software de programación: Diferentes alternativas y lenguajes para desarrollar programas de informática: editores de texto, compiladores, intérpretes, enlazadores, depuradores, entornos de desarrollo integrados...

Software de sistema: Permite a los usuarios llevar a cabo una o varias tareas específicas en cualquier campo de actividad: aplicaciones ofimáticas, para control de sistemas y automatización industrial, software educativo, software empresarial, bases de datos, telecomunicaciones, videojuegos...

3. Describe la relación que existe entre los componentes hardware principales de un computador y el almacenamiento y ejecución del software.

Hay muchas relaciones entre el hardware, el almacenamiento y la ejecución del sotfware, por ejemplo el hardware tiene la memoria rom, que es donde se almacenan los datos, o la memoria ram, que muchos softwares utilizan como una memoria a corto plazo.

4. Define los siguientes conceptos:

· **Código fuente:** Conjunto de sentencias entendibles por el programador que componen el programa o una parte de ello. Suele estar almacenado en un fichero del tipo texto como los que se pueden abrir por ejemplo, con el bloc de notas o Wordpad en los entornos Windows. El código fuente estará escrito en un lenguaje de programación determinado, elegido por el programador.

· **Código objeto:** Conjunto de instrucciones y datos escritos en un lenguaje que entiende el ordenador directamente: binario o código máquina. Provienen de la traducción de cierto código fuente, es un fragmento del programa final y es específico de la plataforma de ejecución.

· **Código ejecutable:** Reúne diferentes códigos u objetos generados por los programadores junto con las "librerías de uso general" (propias del entorno o del lenguaje de programación) componiendo el programa final.

5. Nombra las fases principales del desarrollo de software y explica brevemente que se hace en cada una de ellas.

Etapa de análisis: Es el proceso de investigar un problema que se quiere resolver. Definir claramente el Problema que se desea resolver o el sistema que se desea crear. Identificar los componentes principales que integrarán el producto.

Etapa de Diseño: Es el proceso de utilizar la información recolectada en la etapa de análisis al diseño del producto. La principal tarea de la etapa de diseño es desarrollar un modelo o las especificaciones para el producto o Componentes del Sistema.

Etapa de Desarrollo: Consiste en utilizar los modelos creados durante la etapa de diseño para crear los componentes del sistema.

Etapa de Pruebas o Verificación Prueba : Consiste en asegurar que los componentes individuales que integran al sistema o producto, cumplen con los requerimientos de la especificación creada durante la etapa de diseño.

Etapa de Implementación o Entrega Implantación: Consiste en poner a disposición del cliente el producto.

Etapa de Mantenimiento: Consiste en corregir problemas del producto y re- liberar el producto como una nueva versión o revisión (producto mejorado).

Etapa final EOL (End-of-Life): El fin del ciclo del producto consiste en realizar todas las tareas necesarias para asegurar que los clientes y los empleados están conscientes de que el producto ya no será vendido ni soportado.

2a Part:

6. ¿Qué diferencia existe entre los lenguajes declarativos y los imperativos?. Nombra al menos 2 de cada tipo.

En la programación **imperativa**, de la cual hacen parte muchos de los principales lenguajes de programación tales como **C**, **Java** y **PHP**, **un programa se describe en términos de instrucciones, condiciones y pasos que modifican el estado de un programa al permitir la mutación de variables**, todo esto con el objetivo de llegar a un resultado. En contraparte, en la programación **declarativa un programa se describe en términos de proposiciones y afirmaciones que son declaradas para describir el problema, sin especificar los pasos para resolverlo**; en este tipo de programas, el estado no puede ser modificado ya que todos los tipos de datos son inmutables. De esta familia hacen parte lenguajes como **Scala**, **Haskell**, **Erlang** y **Elixir**.

7. ¿Explica qué es compilar? ¿Explica qué es interpretar?

Compilar se refiere al proceso de traducción del código fuente, entendiéndose por código fuente las líneas de código que se han escrito en un lenguaje de programación. Esta se realiza debido a que el código del lenguaje de programación no es ejecutable directamente por los dispositivos, es por ello la necesidad de traducir las instrucciones contenidas en el texto al llamado “lenguaje de máquina” o código binario.

Interpretar es como compilar, excepto por que los intérpretes solo realizan la traducción a medida que sea necesaria, instrucción por instrucción, y normalmente no guardan el resultado de dicha traducción. Usando un intérprete, un solo archivo fuente puede producir resultados iguales en sistemas diferentes (por ejemplo un PC y una consola).

8. Ventajas de los lenguajes compilados.

Lenguajes Compilados	
Ventajas	Desventajas
Preparados para ejecutarse	No son multiplataforma
A menudo más rápidos	Poco flexibles
El código fuente es inaccesible	Se requiere un paso extra (compilación)

9. Ventajas de los lenguajes interpretados.

Lenguajes Interpretados	
Ventajas	Desventajas
Son multiplataforma	Se requiere un intérprete
Son más sencillos de probar	A menudo más lentos
Los errores se detectan mas fácilmente	El código fuente es público

10. Pon un ejemplo de lenguaje de los siguientes tipos:

· **Bajo nivel:** Binario, Ensamblador

· **Alto nivel:** Java, Python, C#

11. Explica qué criterios pueden seguirse a la hora de elegir un lenguaje de programación para el desarrollo software.

A la hora de seleccionar con qué lenguaje trabajar, es importante conocer las diferencias y singularidades de cada uno de ellos, y sus ventajas e inconvenientes, en función de la plataforma para la que estemos desarrollando nuestro proyecto.