U1. DESARROLLO DE SOFTWARE Y ENTORNOS

1.- ¿Cuáles son los 2 elementos fundamentales que componen cualquier dispositivo informático? ¿Qué es el firmware? Según la función que realiza ¿Qué tipo de software podemos encontrar?

Hardware → Componentes físicos y tangibles del dispositivo, procesador, memoria RAM, disco duro, placa base... Permite la ejecución y operación de programas.

Software → Conjunto de programas y aplicaciones que permiten el funcionamiento del hardware y la realización de tareas específicas. Incluye el sistema operativo y aplicaciones que interactúan con el hardware.

Firmware → Software embebido en la memoria no volátil de un dispositivo (ROM o flash), proporciona instrucciones básicas para el funcionamiento. Es la clave en la seguridad de los sistemas, ya que gestiona el comportamiento del hardware.

Según la función que realizan, encontramos:

a) **Software de sistema**: Gestiona y controla los recursos del hardware.
   - Sistemas operativos.
   - Controladores de dispositivos o Drivers.
   - Utilidades del sistema (antivirus).

b) **Software de aplicación**: Ayudan al usuario a realizar tareas específicas.
   - Aplicaciones de productividad.
   - Software de diseño gráfico.
   - Aplicaciones de comunicación.

c) **Software de programación**: Herramientas para crear otros programas.
   - Editores de código.
   - Compiladores e interpretes (traducen código fuente a código maquina).
   - Entornos de desarrollo integrados (IDE) (Eclipse).

Bibliografía:

- https://micrositios.inai.org.mx/marcocompetencias/?page_id=372#:~:text=Todo%20dispositivo%20inform%C3%A1tico%20est%C3%A1%20compuesto,el%20hardware%20y%20el%20software.&text=Se%20refiere%20a%20las%20partes,%2C%20electr%C3%B3nicos%2C%20electromec%C3%A1nicos%20y%20mec%C3%A1nicos.
- https://www.incibe.es/incibe-cert/blog/analisis-de-firmware-en-dispositivos-industriales#:~:text=Un%20firmware%20se%20define%20como,las%20funciones%20b%C3%A1sicas%20del%20mismo.
- https://www.wolterskluwer.com/es-es/expert-insights/que-tipos-de-software-hay#:~:text=y%20la%20empresa-,Tipos%20de%20software%20por%20funcionalidad,programaci%C3%B3n%20y%20software%20de%20sistema
- Chatgpt

2.- ¿Qué es el desarrollo de software? ¿Cuáles son las fases clásicas del desarrollo de software? Explícalas brevemente.

Desarrollo de software → Proceso de crear aplicaciones, programas o sistemas informáticos, incluye el diseño, programación, pruebas y mantenimiento.

Fases:

1. **Análisis de requisitos**: Recopilación de necesidades del cliente para definir que debe hacer el software.
2. **Diseño**: Planificación de la arquitectura y componentes del software.
3. **Implementación**: Escritura del código y ensamblaje de funcionalidades.
4. **Pruebas**: Verificación de que el software funciona correctamente y cumple los requisitos.
5. **Despliegue**: Instalación y configuración del software en el entorno de producción.
6. **Mantenimiento**: Corrección de errores y actualización del software.

Bibliografía:

- https://www.santanderopenacademy.com/es/blog/metodologias-desarrollo-software.html
- https://aws.amazon.com/es/what-is/sdlc/
- https://www.solbyte.com/blog/5-etapas-del-proceso-de-desarrollo-de-software/
- Chatgpt

3.- Un porcentaje elevado de los desarrollos de software que se llevan a cabo no llegan a utilizarse, o lo hacen durante un periodo breve de tiempo, ¿Por qué piensas que sucede esta situación?

Hay varios factores que pueden ser la causa de este afirmación, algunos de ellos:

1) **Requisitos mal definidos**: Capturar y documentar adecuadamente lo que realmente necesitas desde el principio es esencial. Una comprensión deficiente puede resultar un producto no satisfactorio.
2) **Falta de comunicación**: Entre desarrolladores, clientes y otras partes interesadas es muy importante. Ayuda a minimizar errores y problemas además de asegurar que todos van en la misma dirección.
3) **Problemas de calidad**: Calidad del código y realización de pruebas rigurosas son fundamentales para evitar problemas.
4) **Cambios en el mercado**: La velocidad a la que evolucionan los entornos tecnológicos y las necesidades del mercado, puede provocar que quede obsoleto si no se adapta y evoluciona.
5) **Estimación de tiempo defectuosas**: Según los datos, el 25 % de los proyectos fracasan por una estimación de tiempo defectuosa. Esto puede llevar a presión en el equipo. Recortes y expectativos no cumplidas, lo que genera costos adicionales.

Estos son algunos de los factores que podría llevar a que no se cumpla un proyecto, sin embargo, cabe destacar que puede haber otros, como una gestión ineficaz, falta de recursos...

Bibliografía:

- https://cynoteck.com/es/blog-post/software-development-time-estimation/
- Chatgpt
- Copilot

4.- ¿Cuál piensas que es la fase más importante dentro de las fases clásicas del desarrollo de software? Razona las respuesta.

La fase mas importantes dentro del desarrollo de software es el **Análisis de requisitos**.

Esta fase define claramente que se espera del software y cuales son las principales necesidades. Si se hace correctamente, se garantiza que el equipo tenga una dirección clara desde el principio, con lo que minimizan errores y problemas en adelante.

Un buen análisis ayuda a identificar problemas potenciales antes de que sean mas graves. Ahorra tiempo y reduce costos a la empresa.

Se aumenta la probabilidad de entregar un producto que cumpla con las expectativas y una mayor satisfacción del cliente.

Bibliografía:

- https://www.solbyte.com/blog/5-etapas-del-proceso-de-desarrollo-de-software/#:~:text=1.,debe%20tener%20el%20resultado%20final.

5.- ¿Qué opinas sobre la documentación de los proyectos, consideras que es un elemento importante o prescindible para la gestión de los mismos?

La documentación de los proyectos es un elemento fundamental, imprescindible y un pilar crucial en la gestión del proyecto.

Algunas razones claves serian:

1) **Cohesión del equipo**: Facilita la comunicación, promueve la colaboración y es más fácil que el equipo trabaje de forma coordinada.
2) **Historial de decisiones**: Ayudara a entender porque se hicieron ciertas cosas, además de poder evaluar decisiones futuras en base a experiencias pasadas.
3) **Capacitación y Onboarding**: Es una herramienta valiosa para la captación de nuevos empleados. Permite que los nuevos miembros del equipo se adapten rápidamente al proyecto.
4) **Facilita la revisión**: Permite entender el trabajo realizado y proporcionar un feedback constructivo.
5) **Adaptación a cambios tecnológicos**: Puede incluir tecnologías emergentes o cambios en el sector, con esto se mantiene el equipo actualizado y con la capacidad de adaptarse a los cambios.

Bibliografía:

- https://cs.uns.edu.ar/~ldm/mypage/data/oc/info/guia_para_la_documentacion_de_proyectos_de_software.pdf
- Chatgpt
- Gemini

6.- ¿Qué es la mejora e integración continua en el desarrollo del software? ¿Qué es la gestión ágil de proyectos? ¿Cuáles son las principales diferencias respecto a las gestiones más clásicas?

La mejora e integración continua son practicas esenciales en el desarrollo del software para maximizar la calidad y eficiencia del proceso.

Mejora continua → Realizar mejoras incrementales en procesos y productos. Revisión constante de código, automatización de pruebas y capacitación del equipo. El objetivo es optimizar la calidad del software y aumentar la eficiencia y satisfacción del cliente.

Integración continua → Implica que los desarrolladores integran cambios de código que han escrito con mayor frecuencia a lo largo del ciclo de desarrollo. Desde el repositorio central se ejecutan las compilaciones y pruebas necesarias.

Gestión ágil de proyectos → Es un enfoque iterativo y flexible para el desarrollo de software, se centra en la colaboración continua con el cliente y adaptación a cambios. Se organiza el trabajo en ciclos cortos, 'sprints', permiten entregas frecuentes y ajustes rápidos. Esto mejora la eficiencia, reduce riesgo de fallos y promueve una mejora continua a largo plazo.

Las diferencias entre una **gestión ágil** (GA) y una **gestión clásica** (GC) (modelo cascada):

- **Enfoque**:
  - GA: Trabajo en ciclos cortos, permiten revisión y ajustes constantes.
  - GC: Secuencia lineal de fases, sin volver atrás.
- **Flexibilidad ante cambios**:
  - GA: Se adapta fácilmente a lo largo de los proyectos.
  - GC: Son difíciles de implementar una vez avanzadas las fases.
- **Colaboración con el cliente**:
  - GA: Involucra al cliente de forma continua, obtiene feedback en cada iteración.
  - GC: Interacción limitada, generalmente en fase de inicio y entrega.
- **Enfoque en resultados rápidos**:
  - GA: Entrega funcionalidades incrementales con frecuencia.
  - GC: Entrega al final del proyecto.
- **Documentación**:
  - GA: Comunicación directa y documentación mínima necesaria.
  - GC: Documentación mas exhaustiva en cada fase.
- **Gestión de riesgos**:
  - GA: Se gestionan de manera continua a través de revisiones.
  - GC: Es menos dinámica y mas tardía.

Bibliografía:

- https://www.atlassian.com/es/continuous-delivery/continuous-integration
- https://www.ibm.com/es-es/topics/continuous-integration
- https://www.aden.org/business-magazine/gestion-agil-de-proyectos-que-es-y-por-que-es-importante/
- https://www.atlassian.com/es/agile/project-management#:~:text=%C2%BFQu%C3%A9%20es%20la%20gesti%C3%B3n%20%C3%A1gil,del%20cliente%20con%20cada%20iteraci%C3%B3n.

7.- ¿Qué es un entorno de desarrollo y cuáles son sus ventajas e inconvenientes? ¿Cuándo crees que será conveniente utilizarlo?

Entorno de desarrollo → Es un espacio donde los desarrolladores crean, modifican y prueban aplicación sin afectar a la versión en producción del software. Actividades de mantenimiento, depuración y parches.

Entorno de Desarrollo Integrado (IDE) → Combina diversas herramientas esenciales en una única aplicación, facilitando el proceso de programación. Incluye:

- **Editor de código:** Escribir y organizar el código.
- **Compilador/interprete:** Transformar el código en formato ejecutable.
- **Depurador:** Identifica y corrige errores.
- **Funcionalidades adicionales:** Autocompletado, resaltado de sintaxis y gestión de proyectos.

Ventajas	Inconvenientes
Productividad mejorada	Curva de aprendizaje
Depuración eficiente	Consumo de recursos
Gestión de proyectos	Dependencia
Integración de herramientas	Alto coste
Compatibilidad y extensibilidad	

Un **IDE** seria conveniente en situaciones de:

- ✓ Desarrollo de software complejo.
- ✓ Múltiples lenguajes de programación (si el proyecto involucra varios lenguajes).
- ✓ Necesidad de depuración (depurador integrado).
- ✓ Colaboración en equipo.
- ✓ Automatización de tareas (automatizar tareas repetitivas).
- ✓ Aprendizaje y capacitación (novatos).
- ✓ Desarrollo rápido.
- ✓ Proyectos a largo plazo.

Bibliografía:

- https://www.hostinger.es/tutoriales/que-es-un-entorno-de-desarrollo
- https://datascientest.com/es/ide-que-es

8.- Elige la respuesta correcta. Para desarrollar un programa informático:

a) Hay que escribir las instrucciones en código binario para que las entienda el hardware.
b) Solo es necesario escribir el programa en algún lenguaje de programación y se ejecuta directamente.
c) Hay que escribir el programa en algún lenguaje de programación y contar con herramientas software que traduzcan a código binario.
d) Los programas informáticos no se pueden escribir, forman parte de los sistemas operativos.

9.- Elige la respuesta incorrecta. Respecto a los tipos de código:

a) El código objeto está escrito en lenguaje máquina, pero no es ejecutable.
b) El código ejecutable está escrito en lenguaje maquina y es ejecutable.
c) El código fuente está escrito en lenguaje de programación de alto nivel y no es ejecutable.
d) Antes de empezar a desarrollar con un lenguaje de programación hay que seleccionar que tipo de código vamos a generar (fuente, objeto o ejecutable).

10.- Elige las respuestas correctas. Referente a la documentación de un proyecto:

a) Hay autores que interpretan la documentación como una fase más de las clásicas de un proyecto informático.
b) Solo es necesario documentar aquellas cuestiones importantes o complejas de un proyecto.
c) Es la única forma de universalizar un proyecto y que cualquiera pueda llegar a conocerlo en detalle.
d) Si no está bien remunerado económicamente, no se generará ningún documento.

11.- Elige las respuestas incorrectas. En el contexto de los lenguajes de programación:

a) El lenguaje ensamblador es un lenguaje de bajo nivel muy utilizado por los desarrolladores.
b) Lo más habitual es programar directamente en código máquina, para que sea directamente ejecutable.
c) Los lenguajes estructurados pasaron a ser modulares cuando se permitió dividir el código en fragmentos reutilizables.
d) Los lenguajes orientados a objetos son los mas utilizados actualmente.