**TALLER**

**APRENDIZ: ENYEER MANUEL GRANADOS MARDINIS**

**FICHA: 2558104**

**SERVICIO NACIONAL DE APRENDIZAJE – SENA**

**CENTRO DE ELECTRICIDAD, ELECTRONICA Y TELECOMUNICACIONES – CEET**

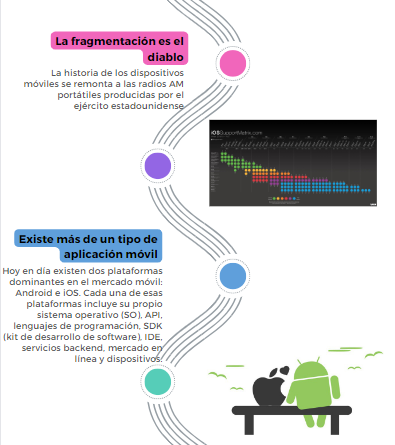
**INSTRUCTORA: ISAURA MARIA SUAREZ NOVOA**

**BOGOTÁ D.C**

***Parte #1***

1.De acuerdo a la historia de los dispositivos móviles elaborar una línea de tiempo











2.Cuadro comparativo entre Android y iOS

|  |  |  |
| --- | --- | --- |
| ***Caracteristicas*** | ***Android*** | ***ios*** |
| Base | Linux | Unix |
| Seguridad | Buena | Excelente |
| Interfaz usuario | Adaptable | Unificada |

3.Ventajas y desventajas de una aplicación nativa

Las aplicaciones nativas son aplicaciones que están diseñadas específicamente para un sistema operativo o plataforma. Están escritas en un lenguaje de programación nativo para ese sistema operativo, y aprovechan las funciones y capacidades específicas de ese sistema.

Ventajas de las aplicaciones nativas:

* Rendimiento: Las aplicaciones nativas suelen ofrecer un mejor rendimiento que las aplicaciones web o híbridas. Esto se debe a que están escritas en un lenguaje de programación nativo que puede acceder directamente a las funciones del sistema operativo.
* Experiencia de usuario: Las aplicaciones nativas suelen ofrecer una mejor experiencia de usuario que las aplicaciones web o híbridas. Esto se debe a que están diseñadas específicamente para el sistema operativo y el hardware en el que se ejecutan.
* Compatibilidad: Las aplicaciones nativas suelen ser más compatibles con los dispositivos y las versiones del sistema operativo. Esto se debe a que están diseñadas específicamente para un sistema operativo específico.

Desventajas de las aplicaciones nativas:

* Coste: El desarrollo de aplicaciones nativas puede ser más costoso que el desarrollo de aplicaciones web o híbridas. Esto se debe a que requiere conocimientos especializados en el lenguaje de programación nativo y en las API del sistema operativo.
* Tiempo de desarrollo: El desarrollo de aplicaciones nativas puede llevar más tiempo que el desarrollo de aplicaciones web o híbridas. Esto se debe a que las aplicaciones nativas deben ser diseñadas y desarrolladas específicamente para cada sistema operativo.
* Mantenimiento: El mantenimiento de aplicaciones nativas puede ser más complejo que el mantenimiento de aplicaciones web o híbridas. Esto se debe a que las aplicaciones nativas deben ser mantenidas para cada sistema operativo en el que se ejecutan.

4.Que es una aplicación nativa, hibrida y multiplataforma

Aplicación nativa

Una aplicación nativa es una aplicación que está diseñada específicamente para un sistema operativo o plataforma. Están escritas en un lenguaje de programación nativo para ese sistema operativo, y aprovechan las funciones y capacidades específicas de ese sistema.

Aplicación híbrida

Una aplicación híbrida es una aplicación que utiliza una combinación de tecnologías web y nativas para funcionar en múltiples plataformas. Las aplicaciones híbridas suelen estar escritas en HTML, CSS y JavaScript, y utilizan un marco híbrido para acceder a las funciones del sistema operativo.

Aplicación multiplataforma

Una aplicación multiplataforma es una aplicación que puede ejecutarse en múltiples plataformas sin necesidad de ser modificada. Las aplicaciones multiplataforma suelen estar escritas en un lenguaje de programación de alto nivel que puede ser compilado para múltiples plataformas.

5. ¿Estás de acuerdo o no, en que el diseño debe estar presente en todo el ciclo de desarrollo de software? Justifica tu respuesta.

Es importante el diseño del software ya que debido al avance del tiempo se pueden realizar actualizaciones y cambios que ayudan a la mejora del software.

6. Explica los 5 pasos del proceso del pensamiento de diseño. Luego aplícalo a un ejemplo.

El proceso de pensamiento de diseño es una metodología iterativa que se utiliza para resolver problemas de forma creativa. Consta de cinco pasos:

* Empatía: En esta etapa, el diseñador se centra en comprender las necesidades y los deseos de los usuarios. Para ello, puede realizar entrevistas, encuestas o grupos de enfoque.
* Definición: En esta etapa, el diseñador define el problema que se va a resolver. Para ello, puede utilizar una herramienta como el mapa de empatía.
* Ideación: En esta etapa, el diseñador genera ideas para resolver el problema. Para ello, puede utilizar técnicas como el brainstorming o el mindmapping.
* Prototipado: En esta etapa, el diseñador crea prototipos para probar las ideas. Los prototipos pueden ser simples o complejos, dependiendo del proyecto.
* Evaluación: En esta etapa, el diseñador evalúa los prototipos con los usuarios. Para ello, puede utilizar técnicas como las pruebas de usuario o las encuestas.

Ejemplo:

7. ¿Qué es Design Thinking?

Design Thinking es una metodología para la resolución de problemas de forma creativa, centrada en el usuario. Se trata de un proceso iterativo que consta de cinco pasos:

* Empatía: Comprender las necesidades, los deseos y las frustraciones de los usuarios.
* Definición: Identificar el problema a resolver.
* Ideación: Generar ideas para resolver el problema.
* Prototipado: Crear prototipos para probar las ideas con los usuarios.
* Evaluación: Obtener feedback de los usuarios sobre los prototipos y mejorarlos.

8. Explica y aplica con un ejemplo las etapas del Design Thinking

El Design Thinking es una metodología de resolución de problemas que se centra en la empatía con el usuario y la generación de soluciones creativas.

Empatía: Ponerse en los zapatos de alguien más: para la realización de un software de un gimnasio estaría bien preguntarles a las personas que asisten al gimnasio, como quisieran que hiciera el software.

Definición: El software para los usuarios sea fácil de usar, tenga una variedad de ejercicios.

Ideación: Generar ideas para la realización del software.

Prototipado: Realizar un prototipo del software con el fin de ser probado para comprobar la calidad y funcionamiento.

Evaluación: Recopilar información del prototipo para así poder realizar las correcciones

9. ¿Qué es ser diseñador inclusivo? ¿Por qué es importante? ¿Estás de acuerdo en que debemos ser diseñadores inclusivos? Justifica tu respuesta.

Un diseñador inclusivo es aquel que se centra en crear productos y servicios que sean accesibles y usables para todos, independientemente de sus capacidades, edad, género, raza, etnia, orientación sexual, etc.

El diseño inclusivo es importante porque permite que todas las personas tengan la oportunidad de participar plenamente en la sociedad. Cuando los productos y servicios son inclusivos, las personas con discapacidades pueden acceder a ellos y utilizarlos de forma independiente. Esto les permite vivir de forma más independiente y autónoma, y participar en las mismas actividades que las personas sin discapacidades.

RTA:

Es importante ya que con diseño del producto se hacen mejoras para estos, que acomoden a la necesidad del cliente independientemente de sus capacidades, etc.

10. De acuerdo a las herramientas que podemos utilizar para ser diseñadores inclusivos, Escoge 2 herramientas y realiza una investigación de cómo funciona y como se puede implementar en una aplicación web y una móvil.

UsabilityHub es una plataforma que proporciona una variedad de herramientas para evaluar la usabilidad y la accesibilidad de los diseños. Estas herramientas incluyen:

* Tests de usabilidad: Estos tests permiten a los diseñadores probar sus diseños con usuarios reales y obtener comentarios sobre la facilidad de uso y la satisfacción del usuario.
* Analizadores de usabilidad: Estos analizadores pueden ayudar a los diseñadores a identificar problemas de usabilidad en sus diseños.
* Estudios de accesibilidad: Estos estudios permiten a los diseñadores evaluar la accesibilidad de sus diseños según las pautas de accesibilidad.

Accessibility Scanner es una herramienta que puede ayudar a los diseñadores a identificar problemas de accesibilidad en sus diseños de sitios web y aplicaciones móviles. Esta herramienta escanea los diseños en busca de problemas comunes de accesibilidad, como texto no accesible, elementos de interfaz de usuario difíciles de usar y contenido no compatible con lectores de pantalla.

Implementación en una aplicación web:

* Crear una cuenta de UsabilityHub.
* Diseñar la aplicación web.
* Publicar la aplicación web en UsabilityHub.
* Invitar a los usuarios a probar la aplicación web.
* Analizar los resultados de los tests de usabilidad.

Implementación en una aplicación móvil:

* Crear una cuenta de UsabilityHub.
* Diseñar la aplicación móvil.
* Publicar la aplicación móvil en UsabilityHub.
* Invitar a los usuarios a probar la aplicación móvil.
* Analizar los resultados de los tests de usabilidad.

11. Explica la importancia de la empatía con el usuario en el desarrollo del software.

La empatía con el usuario es la capacidad de comprender y compartir los sentimientos y experiencias de otra persona. En el desarrollo de software, la empatía con el usuario es importante porque permite a los desarrolladores crear productos y servicios que satisfagan las necesidades y deseos de los usuarios.

* Comprender las necesidades y deseos de los usuarios: Los desarrolladores pueden comprender mejor lo que los usuarios necesitan y quieren al ponerse en su lugar. Esto les permite crear productos y servicios que sean útiles, eficientes y atractivos.
* Identificar los problemas de usabilidad: Los desarrolladores pueden identificar problemas de usabilidad al pensar en cómo los usuarios interactuarán con el producto o servicio. Esto les permite crear productos y servicios que sean fáciles de usar y comprender.
* Crear productos y servicios que sean accesibles: Los desarrolladores pueden crear productos y servicios que sean accesibles para todos los usuarios, independientemente de sus capacidades.

12. Que es y cómo se implementa un mapa de empatía.

Un mapa de empatía es una herramienta visual que ayuda a los diseñadores a comprender las necesidades, deseos y sentimientos de los usuarios. Se trata de una representación gráfica de un usuario, con seis cuadrantes que representan diferentes aspectos de su experiencia.

Los seis cuadrantes de un mapa de empatía son:

* ¿Qué ve el usuario?: Este cuadrante representa los estímulos visuales que el usuario recibe.
* ¿Qué escucha el usuario?: Este cuadrante representa los estímulos auditivos que el usuario recibe.
* ¿Qué dice el usuario?: Este cuadrante representa las palabras y frases que el usuario utiliza.
* ¿Qué piensa el usuario?: Este cuadrante representa los pensamientos y sentimientos del usuario.
* ¿Qué siente el usuario?: Este cuadrante representa las emociones del usuario.
* ¿Qué necesidades tiene el usuario?: Este cuadrante representa las necesidades del usuario.

Para implementar un mapa de empatía, los diseñadores deben seguir estos pasos:

* Identificar al usuario objetivo: El primer paso es identificar al usuario objetivo del producto o servicio que se está diseñando.
* Reunir información sobre el usuario: Una vez identificado el usuario objetivo, los diseñadores deben reunir información sobre él. Esta información puede recopilarse a través de entrevistas, encuestas, grupos de enfoque o observación.
* Completar el mapa de empatía: Con la información recopilada, los diseñadores pueden completar el mapa de empatía. Para ello, deben rellenar cada cuadrante con información relevante sobre el usuario.
* Revisar el mapa de empatía: Una vez completado el mapa de empatía, los diseñadores deben revisarlo para asegurarse de que es preciso y completo.

13. ¿Qué es un diagrama de contexto? ¿Y que es el context canvas?

Un diagrama de contexto es una representación visual de los actores y los sistemas que interactúan con un sistema dado. Se utiliza para comprender el entorno en el que se encuentra un sistema y cómo interactúa con el mundo exterior.

Un diagrama de contexto suele representarse como un círculo que representa el sistema, con líneas que conectan el sistema con los actores y sistemas externos. Los actores son personas, grupos o sistemas que interactúan con el sistema. Los sistemas externos son sistemas que no son controlados por el sistema en cuestión.

El context canvas es una herramienta visual que ayuda a los diseñadores a comprender el contexto en el que se encuentra un producto o servicio. Se basa en el diagrama de contexto, pero añade información adicional sobre los objetivos del sistema, los requisitos del sistema y los recursos del sistema.

El context canvas suele representarse como un lienzo dividido en cuatro cuadrantes:

* Actores: Representa los actores que interactúan con el sistema.
* Objetivos: Representa los objetivos del sistema.
* Requisitos: Representa los requisitos del sistema.
* Recursos: Representa los recursos del sistema.
* Diagrama de contexto vs. context canvas

La principal diferencia entre un diagrama de contexto y un context canvas es que el context canvas añade información adicional sobre los objetivos, requisitos y recursos del sistema. Esto lo convierte en una herramienta más completa para comprender el contexto en el que se encuentra un producto o servicio.

14. Elabora un cuadro comparativo entre el diagrama de contexto y el context canvas.

|  |  |  |
| --- | --- | --- |
| Caracteristicas | Diagrama de contexto | Context canvas |
| Información | Actores, sistemas externos. | Actores, objetivos, requisitos, recursos. |
| Proposito | Comprender el entorno en el que se encuentra un sistema y cómo interactúa con el mundo exterior. | Comprender el contexto en el que se encuentra un producto o servicio. |
| Ventajas | Fácil de entender y crear. | Proporciona una visión más completa del contexto del producto o servicio. |

15.Elabora un ejemplo de diagrama de contexto y context canvas

Diagrama de contexto:

* Proveedor
* Cliente
* Sistema de registros
* Sistema de reservas

Diagrama context canvas:

Actores:

* Proveedor: Registrar producto, donar producto
* Cliente: Observar productos, reservar productos

Objetivos:

* Evitar y disminuir el deterioro de productos
* Aumentar la comercialización de productos

Requisitos:

* El aplico debe ser fácil de usar
* Debe ofrecer una amplia selección de productos

16. Con base a la explicación. Elabora un ejemplo de personas y usuario.

Personas

Proveedor:

* Nombre: Juan Pérez
* Profesión: Agricultor
* Edad: 50 años
* Datos demográficos: Vive en un pueblo rural de Colombia
* Objetivos: Donar productos

Cliente:

* Nombre: María García
* Profesión: Estudiante
* Edad: 25 años
* Datos demográficos: Vive en una ciudad de Colombia
* Objetivos: Conseguir productos para una fundación

Proveedor:

* Nombre de usuario: JuanPerez
* Contraseña: 123456
* Perfil: Agricultor de productos orgánicos
* Productos: Frutas y verduras

Cliente:

* Nombre de usuario: MariaGarcia
* Contraseña: 789012
* Perfil: Estudiante de nutrición
* Productos: Alimentos

Proveedor

* Juan Pérez es un agricultor de 50 años que vive en un pueblo rural de Colombia. Su objetivo es donar sus productos antes que sean desperdiciados.
* Juan utiliza el sistema para registrar sus productos y donar los que están en mal estado a organizaciones benéficas.

Cliente

* María García es una estudiante de 25 años que vive en una ciudad de Colombia. Su objetivo es conseguir productos para una fundación.
* María utiliza el sistema para ver una amplia selección de productos. También utiliza el sistema para reservar productos.

***Parte #2***

1. ¿Qué es la arquitectura de datos y en que radica su importancia?

La arquitectura de datos es el conjunto de principios, prácticas y tecnologías que se utilizan para organizar y gestionar los datos de una organización. Su objetivo es garantizar que los datos sean accesibles, fiables y útiles para el cumplimiento de los objetivos de la organización.

La arquitectura de datos es importante por varias razones:

* Mejora la eficiencia y la productividad: Una arquitectura de datos bien diseñada puede ayudar a las organizaciones a ahorrar tiempo y dinero al facilitar la recopilación, el almacenamiento y el análisis de datos.
* Mejora la calidad de los datos: Una arquitectura de datos sólida puede ayudar a garantizar que los datos sean precisos, completos y coherentes.
* Facilita la toma de decisiones: Una arquitectura de datos bien diseñada puede ayudar a las organizaciones a tomar decisiones más informadas al proporcionarles acceso a los datos relevantes.

2. Menciona y explica cada uno de los métodos y procesos propuestos para diseño de proyectos de análisis y ciencia de datos.

Existen diversos métodos y procesos propuestos para el diseño de proyectos de análisis y ciencia de datos. Estos métodos y procesos se basan en diferentes enfoques y filosofías, y están diseñados para satisfacer las necesidades específicas de cada proyecto.

Algunos de los métodos y procesos más comunes para el diseño de proyectos de análisis y ciencia de datos son los siguientes:

* El método CRISP-DM: Este método es un enfoque sistemático y completo para el desarrollo de proyectos de análisis de datos. Se divide en seis fases: comprensión del negocio, comprensión de los datos, preparación de los datos, modelado, evaluación y despliegue.
* El método DMAIC: Este método es un enfoque centrado en la mejora de procesos. Se divide en cinco fases: definición, medición, análisis, mejora y control.
* El método OSEMN: Este método es un enfoque centrado en el aprendizaje automático. Se divide en seis fases: obtención de datos, exploración de datos, selección de características, modelado, evaluación y despliegue.

3. ¿Que son los datos, información y conocimiento? Realiza un ejemplo

***Datos*** son hechos brutos, sin procesar y sin significado. Pueden ser números, texto, imágenes o cualquier otro tipo de representación de la realidad. En programación, los datos se utilizan para representar el mundo real. Por ejemplo, una base de datos de clientes puede almacenar datos sobre los nombres, direcciones y números de teléfono de los clientes.

***Información*** es datos organizados y con significado. La información puede ser utilizada para tomar decisiones, resolver problemas o simplemente para aprender. En programación, la información se utiliza para representar el conocimiento del mundo real. Por ejemplo, un programa de reconocimiento facial puede utilizar información sobre las características faciales para identificar a las personas.

***Conocimiento*** es información que ha sido asimilada y entendida por un individuo o grupo. El conocimiento puede ser tácito, es decir, que está en la mente de las personas, o explícito, es decir, que está documentado. En programación, el conocimiento se utiliza para crear programas que sean eficientes y eficaces. Por ejemplo, un programador puede utilizar su conocimiento de algoritmos para crear un programa que resuelva un problema de manera eficiente.

Ejemplo:

Una empresa de comercio puede utilizar la recopilación de datos sobre las compras de sus clientes. Esta información hace que la empresa tenga conocimiento para poder mejorar la experiencia del cliente, aumentar las ventas y optimizar las campañas de marketing.

4. ¿Como se debe construir un motor de análisis?

La construcción de un motor de análisis es un proceso complejo que requiere una serie de pasos y consideraciones.

Pasos básicos que deben seguirse para construir un motor de análisis:

* Definición de los requisitos: El primer paso es definir los requisitos del motor de análisis. Esto incluye definir los datos que se van a analizar, los tipos de análisis que se van a realizar y los resultados que se esperan.
* Recopilación de datos: Una vez que se han definido los requisitos, se procede a recopilar los datos que se van a analizar. Los datos pueden ser recopilados de una variedad de fuentes, como bases de datos, archivos de texto, sensores y redes sociales.
* Preparación de datos: Los datos recopilados pueden requerir una preparación antes de poder ser analizados. Esto puede incluir la limpieza de los datos, la transformación de los datos y la reducción de la dimensionalidad de los datos.
* Diseño del motor de análisis: El siguiente paso es diseñar el motor de análisis. Esto incluye seleccionar los algoritmos de análisis que se van a utilizar y definir la arquitectura del motor.
* Implementación del motor de análisis: Una vez que se ha diseñado el motor de análisis, se procede a su implementación. La implementación puede realizarse utilizando una variedad de lenguajes de programación y frameworks.
* Pruebas del motor de análisis: El motor de análisis debe ser probado para garantizar que cumple con los requisitos. Las pruebas pueden realizarse utilizando una variedad de técnicas, como pruebas de unidad, pruebas de integración y pruebas de rendimiento.
* Despliegue del motor de análisis: Una vez que el motor de análisis ha sido probado, se procede a su despliegue. El despliegue puede realizarse en una variedad de entornos, como la nube, los servidores locales o los dispositivos móviles.

Además de estos pasos básicos, también es importante considerar los siguientes aspectos al construir un motor de análisis:

* La escalabilidad: El motor de análisis debe ser escalable para poder adaptarse a volúmenes crecientes de datos.
* La eficiencia: El motor de análisis debe ser eficiente para poder procesar datos de manera rápida y eficaz.
* La precisión: El motor de análisis debe ser preciso para poder generar resultados confiables.
* La interpretabilidad: El motor de análisis debe ser interpretable para que los usuarios puedan comprender los resultados del análisis.

5. ¿Qué es el mapa VD? Brinda un ejemplo y Explícalo.

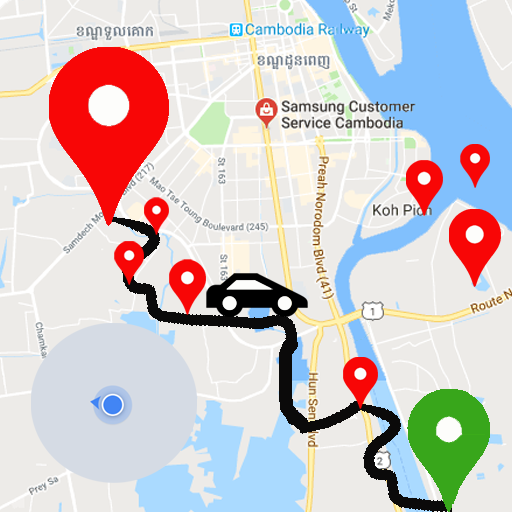
El mapa VD es un mapa digital de la República de Vanuatu, un país insular del océano Pacífico. El mapa fue desarrollado por el gobierno de Vanuatu en colaboración con la Agencia de los Estados Unidos para el Desarrollo Internacional (USAID).

El mapa VD se basa en datos de satélite y de campo. Los datos de satélite se utilizan para crear la base cartográfica del mapa, mientras que los datos de campo se utilizan para validar la base cartográfica y añadir información adicional, como los nombres de los lugares, las carreteras y los ríos.

El mapa VD se puede utilizar para una variedad de propósitos, como la planificación del desarrollo, la gestión de desastres y la educación.

características del mapa VD:

* Estándares internacionales: El mapa VD cumple con los estándares internacionales de cartografía, como la norma ISO 19115.
* Precisión: El mapa VD tiene un error de 10 metros para la mayoría de los lugares.
* Actualización: El mapa VD se actualiza regularmente con nueva información.
* Accesibilidad: El mapa VD está disponible en línea y se puede descargar de forma gratuita.



6. ¿Que son los escenarios funcionales y de calidad? Realiza un cuadro comparativo.

Los escenarios funcionales y de calidad son dos tipos de escenarios que se utilizan en el desarrollo de software. Los escenarios funcionales describen cómo el software debe funcionar, mientras que los escenarios de calidad describen los criterios de calidad que debe cumplir el software.

Escenarios funcionales

Los escenarios funcionales describen cómo el software debe funcionar en términos de tareas que los usuarios pueden realizar. Los escenarios funcionales se utilizan para definir los requisitos funcionales del software, que son las características que el software debe tener para satisfacer las necesidades de los usuarios.

Los escenarios funcionales suelen incluir los siguientes elementos:

* Un actor: La persona o entidad que realiza la tarea.
* Una acción: La tarea que realiza el actor.
* Un resultado: El resultado de la tarea.

Escenarios de calidad

Los escenarios de calidad describen los criterios de calidad que debe cumplir el software. Los escenarios de calidad se utilizan para definir los requisitos de calidad del software, que son los atributos que el software debe tener para ser aceptable para los usuarios.

Los escenarios de calidad suelen incluir los siguientes elementos:

* Un atributo de calidad: El atributo de calidad que se está evaluando.
* Un criterio de calidad: El criterio que se utiliza para evaluar el atributo de calidad.
* Una medida: La medida utilizada para cuantificar el atributo de calidad.

|  |  |  |
| --- | --- | --- |
| Características | Escenarios funcionales | Escenarios de calidad |
| Objetivo | Definir los requisitos funcionales del software | Definir los requisitos de calidad del software |
| Elementos | Actor, acción, resultado | Atributo de calidad, criterio de calidad, medida |
| Importancia | Ayudan a definir los requisitos funcionales y de calidad del software | Ayudan a evaluar si el software cumple con los requisitos de calidad |

7. Explica que es la GUI y el material design.

GUI es el acrónimo de Graphical User Interface, que en español significa Interfaz Gráfica de Usuario. Es un tipo de interfaz que utiliza elementos gráficos, como botones, iconos y menús, para permitir a los usuarios interactuar con el software.

Material Design es un lenguaje de diseño desarrollado por Google que proporciona directrices para crear interfaces de usuario que sean atractivas, fáciles de usar y accesibles. Material Design se basa en los principios de la física y la materialidad para crear una sensación de profundidad y movimiento en las interfaces.

GUI y Material Design son dos conceptos relacionados, pero no son lo mismo. GUI es un tipo de interfaz, mientras que Material Design es un lenguaje de diseño que puede utilizarse para crear interfaces de usuario GUI.

8. Explica que es el modelo UI y UX. Realiza un ejemplo.

El modelo UI y UX es un enfoque para el desarrollo de productos digitales que se centra en la experiencia del usuario. El modelo se divide en dos partes principales:

* UI (User Interface): El diseño de la interfaz de usuario, que se centra en la apariencia y el comportamiento de la interfaz.
* UX (User Experience): La experiencia del usuario, que se centra en cómo los usuarios interactúan con el producto.

El modelo UI y UX es importante porque ayuda a garantizar que los productos digitales sean fáciles de usar y agradables para los usuarios.



9. ¿Por qué son importante los Bocetos manuales y los prototipos digitales? Justifica.

Los bocetos manuales y los prototipos digitales son importantes en el diseño de productos digitales porque permiten a los diseñadores explorar diferentes ideas y conceptos de manera rápida y eficiente.

Bocetos manuales

Los bocetos manuales son una forma rápida y fácil de capturar ideas y conceptos. Son ideales para explorar diferentes opciones de diseño y para comunicar ideas a otros miembros del equipo.

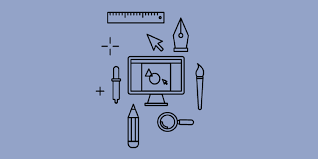
Los bocetos manuales también son útiles para probar diferentes layouts y flujos de trabajo. Esto puede ayudar a los diseñadores a identificar problemas potenciales en el diseño antes de que se implementen en el producto final.

Prototipos digitales

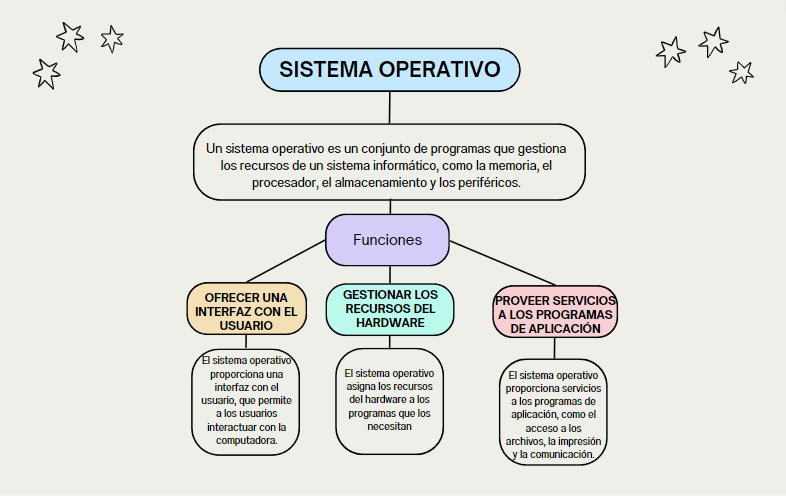
Los prototipos digitales son versiones simuladas de un producto digital. Se utilizan para probar la funcionalidad y la usabilidad de un producto antes de que se implemente en su totalidad.

Los prototipos digitales pueden ser estáticos o interactivos. Los prototipos estáticos son imágenes o videos que representan cómo se verá el producto. Los prototipos interactivos permiten a los usuarios interactuar con el producto de manera similar a como lo harían con el producto final.

Justificación: Los Bocetos manuales y los prototipos digitales sirven para observar y detallar como será la funcionalidad final esto sirve para corregir errores.



10. Explica que es un sistema operativo a través de un mapa mental.



11. ¿Qué es el kernel?

En informática, un kernel es un software que constituye una parte fundamental del sistema operativo, y se define como la parte que se ejecuta en modo privilegiado.

El kernel es responsable de las siguientes tareas:

* Gestión de recursos: El kernel gestiona los recursos del sistema, como la memoria, el procesador y el almacenamiento.
* Control de dispositivos: El kernel controla los dispositivos de hardware, como la pantalla, el teclado y el disco duro.
* Gestión de procesos: El kernel gestiona los procesos, que son las tareas que se ejecutan en un sistema operativo.
* Gestión de memoria: El kernel gestiona la memoria, que es el espacio de almacenamiento utilizado por los procesos.
* Gestión de interrupciones: El kernel gestiona las interrupciones, que son señales que envía el hardware al sistema operativo.

El kernel se ejecuta en modo privilegiado, lo que significa que tiene acceso completo a los recursos del sistema. Esto es necesario para que el kernel pueda gestionar los recursos de manera eficiente y segura.

Los kernels se pueden clasificar en dos tipos principales:

* Kernel monolítico: Un kernel monolítico es un kernel que contiene todo el código del sistema operativo en una sola pieza.
* Kernel modular: Un kernel modular es un kernel que divide el código del sistema operativo en módulos independientes.

12. Explica cada una de las capas en que se compone el sistema operativo Android e IOS

Android se compone de cuatro capas principales:

* Núcleo: El núcleo es la capa más baja y es responsable de la gestión de los recursos del sistema, como la memoria, el procesador y el almacenamiento. El núcleo de Android es un kernel Linux modificado.
* Librerías: La capa de bibliotecas proporciona funciones básicas para el sistema operativo, como la gestión de procesos, la gestión de memoria y la gestión de archivos.
* Framework: El framework proporciona una API para que las aplicaciones puedan interactuar con el sistema operativo. El framework de Android se basa en el framework de Java.
* Aplicaciones: La capa de aplicaciones proporciona las aplicaciones que los usuarios pueden utilizar, como la pantalla de inicio, el navegador web y la aplicación de correo electrónico. Las aplicaciones de Android se pueden desarrollar en Java o Kotlin.

Capas de iOS

iOS se compone de cinco capas principales:

* Kernel: El kernel es la capa más baja y es responsable de la gestión de los recursos del sistema, como la memoria, el procesador y el almacenamiento. El kernel de iOS es un kernel Mach modificado.
* Librerías: La capa de bibliotecas proporciona funciones básicas para el sistema operativo, como la gestión de procesos, la gestión de memoria y la gestión de archivos.
* Cocoa Touch: La capa Cocoa Touch proporciona una API para que las aplicaciones puedan interactuar con el sistema operativo. Cocoa Touch se basa en el framework de Objective-C.
* Frameworks: La capa de frameworks proporciona funciones adicionales para el sistema operativo, como la gestión de gráficos, la gestión de audio y la gestión de redes.
* Aplicaciones: La capa de aplicaciones proporciona las aplicaciones que los usuarios pueden utilizar, como la pantalla de inicio, el navegador web y la aplicación de correo electrónico. Las aplicaciones de iOS se pueden desarrollar en Objective-C o Swift.

Explicación de cada capa

Núcleo

El núcleo es la capa más baja del sistema operativo y es responsable de la gestión de los recursos del sistema, como la memoria, el procesador y el almacenamiento. El núcleo proporciona una interfaz entre el hardware y el software del sistema operativo.

Librerías

La capa de bibliotecas proporciona funciones básicas para el sistema operativo, como la gestión de procesos, la gestión de memoria y la gestión de archivos. Las bibliotecas proporcionan un conjunto de funciones que las aplicaciones pueden utilizar para interactuar con el sistema operativo.

Framework

El framework proporciona una API para que las aplicaciones puedan interactuar con el sistema operativo. El framework proporciona un conjunto de clases y métodos que las aplicaciones pueden utilizar para acceder a las funciones del sistema operativo.

Aplicaciones

La capa de aplicaciones proporciona las aplicaciones que los usuarios pueden utilizar. Las aplicaciones son programas que pueden realizar tareas específicas, como navegar por Internet, enviar mensajes de texto o jugar juegos.

13. Explique las dos principales arquitecturas de kernel.

Las dos principales arquitecturas de kernel son:

Kernel monolítico: Un kernel monolítico es un kernel que contiene todo el código del sistema operativo en una sola pieza.

Kernel modular: Un kernel modular es un kernel que divide el código del sistema operativo en módulos independientes.

Kernel monolítico

Un kernel monolítico es el tipo de kernel más común. Es un kernel que contiene todo el código del sistema operativo en una sola pieza. Esto significa que el kernel tiene acceso completo a todos los recursos del sistema, lo que puede mejorar el rendimiento.

Los kernels monolíticos son más eficientes que los kernels modulares, pero también son más difíciles de mantener y actualizar. Esto se debe a que cualquier cambio en el kernel puede afectar a todos los demás componentes del sistema operativo.

Kernel modular

Un kernel modular es un kernel que divide el código del sistema operativo en módulos independientes. Esto significa que cada módulo es responsable de una función específica del sistema operativo.

Los kernels modulares son más fáciles de mantener y actualizar que los kernels monolíticos. Esto se debe a que cualquier cambio en un módulo no afecta a los demás módulos.

14. ¿Qué es un framework? ¿Cuál es la importancia de estos? Menciona las ventajas y desventajas.

Un framework es un conjunto de herramientas y clases que proporcionan una estructura básica para el desarrollo de software. Los frameworks pueden ayudar a los desarrolladores a ahorrar tiempo y esfuerzo, y a crear aplicaciones más robustas y fáciles de mantener.

Los frameworks son importantes por las siguientes razones:

* Ahorro de tiempo y esfuerzo: Los frameworks proporcionan una base sólida para el desarrollo de software, lo que puede ayudar a los desarrolladores a ahorrar tiempo y esfuerzo.
* Mejor calidad: Los frameworks pueden ayudar a los desarrolladores a crear aplicaciones más robustas y fáciles de mantener.
* Mayor productividad: Los frameworks pueden ayudar a los desarrolladores a ser más productivos, ya que proporcionan una base sobre la que pueden construir sus aplicaciones.

Las principales ventajas de los frameworks son:

* Reutilización de código: Los frameworks proporcionan una biblioteca de código reutilizable que los desarrolladores pueden utilizar para crear sus aplicaciones.
* Seguridad: Los frameworks pueden ayudar a los desarrolladores a crear aplicaciones más seguras, ya que proporcionan funciones de seguridad integradas.
* Eficiencia: Los frameworks pueden ayudar a los desarrolladores a crear aplicaciones más eficientes, ya que proporcionan una base optimizada para el desarrollo de software.

Las principales desventajas de los frameworks son:

* Dificultad de aprendizaje: Los frameworks pueden ser difíciles de aprender, especialmente para los desarrolladores principiantes.
* Especificidad: Los frameworks pueden estar diseñados para un tipo específico de aplicación, lo que puede limitar su flexibilidad.
* Coste: Los frameworks pueden ser costosos, especialmente los frameworks de pago.