**Taller**

**Santiago Salamanca Díaz**

**ADSO**

**Ficha 2558104**

**Institución: Servicio Nacional de Aprendizaje (SENA), CEET**

**Bogotá D.C.**

**1.Cuadro comparativo entre Android y iOS**

|  |  |  |
| --- | --- | --- |
| Aspecto | Android | iOS |
| DESARROLLADOR | **Desarrollado por Google.** | **Desarrollado por Apple.** |
| DISPONIBILIDAD | **Disponible en una amplia variedad de fabricantes de dispositivos, lo que ofrece muchas opciones de hardware.** | **Exclusivo para dispositivos Apple como iPhone, iPad y iPod Touch.** |
| PERSONALIZACION | **Ofrece una mayor capacidad de personalización, permitiendo cambios en el sistema, widgets y la elección de tiendas de aplicaciones.** | **Limitada capacidad de personalización en comparación con Android.** |
| TIENDAS DE APLICACIONES | **Google Play Store, que ofrece una amplia selección de aplicaciones, incluyendo aplicaciones de terceros.** | **App Store, que es conocida por su estricto control de calidad y seguridad.** |
| ACTUALIZACIONES DEL SISTEMA | **Las actualizaciones del sistema pueden variar según el fabricante y el dispositivo.** | **Apple proporciona actualizaciones de sistema de manera consistente y durante varios años en la mayoría de los dispositivos.** |
| INTEROPERABILIDAD | **Mayor interoperabilidad con otros dispositivos y servicios, como Windows y servicios de Google.** | **Fuerte integración con otros dispositivos Apple y servicios, como macOS y iCloud.** |
| PRECIO | **Una amplia gama de dispositivos con precios variados, incluyendo opciones más asequibles.** | **Dispositivos Apple tienden a ser más caros en promedio.** |
| FRAFMENTACION | **La fragmentación es un problema, ya que los dispositivos Android ejecutan diferentes versiones del sistema y tienen variaciones en la interfaz.** | **Menos fragmentación debido a la estrecha integración de hardware y software en dispositivos Apple.** |
| SEGURIDAD | **Puede ser más vulnerable a malware y virus debido a la diversidad de dispositivos y tiendas de aplicaciones.** | **Mayor énfasis en la seguridad y control de calidad en la App Store, lo que reduce el riesgo de malware.** |
| ASISTENTES VIRTUALES | **Google Assistant es el asistente virtual predeterminado.** | **Siri es el asistente virtual predeterminado.** |
| MULTITAREA | **Mayor flexibilidad en la multitarea y la capacidad de ejecutar aplicaciones en segundo plano.** | **Control más estricto de la multitarea para garantizar un rendimiento óptimo del sistema.** |
| COMPATIBILIDAD CON HARDWARE | **Compatible con una amplia gama de dispositivos, incluyendo teléfonos, tabletas, relojes inteligentes y más.** | Limitado a dispositivos Apple, pero con una amplia gama de productos que funcionan juntos. |

**2.** **Ventajas y desventajas de una aplicación nativa**

**Ventajas de las aplicaciones nativas:**

1. Rendimiento superior: Las aplicaciones nativas suelen ofrecer un rendimiento más rápido y suave, ya que están diseñadas específicamente para la plataforma y el hardware en el que se ejecutan.
2. Acceso completo a las API del sistema: Las aplicaciones nativas tienen acceso completo a las API del sistema operativo, lo que les permite utilizar todas las características y capacidades del dispositivo, como la cámara, el GPS, los sensores y otros componentes.
3. Experiencia de usuario optimizada: Puedes crear una experiencia de usuario altamente personalizada y optimizada para la plataforma, lo que generalmente resulta en una interfaz de usuario más intuitiva y atractiva.
4. Disponibilidad offline: Las aplicaciones nativas pueden funcionar sin conexión a Internet, lo que permite a los usuarios acceder a ciertas funcionalidades incluso en ausencia de conexión.
5. Integración con tiendas de aplicaciones: Las tiendas de aplicaciones, como la App Store de Apple y Google Play Store, ofrecen una plataforma confiable para distribuir y promocionar aplicaciones nativas.
6. Seguridad mejorada: Las aplicaciones nativas suelen ser más seguras, ya que pueden aprovechar las medidas de seguridad del sistema operativo y recibir actualizaciones de seguridad regulares.
7. Mejor aprovechamiento del hardware: Las aplicaciones nativas pueden aprovechar al máximo el hardware del dispositivo, lo que resulta en un rendimiento óptimo y una mayor eficiencia en el uso de recursos.

**Desventajas de las aplicaciones nativas:**

1. Desarrollo costoso y complejo: El desarrollo de aplicaciones nativas puede ser costoso y llevar más tiempo debido a la necesidad de crear versiones separadas para diferentes plataformas (iOS y Android).
2. Mantenimiento de múltiples códigos: Mantener y actualizar dos códigos separados (uno para iOS y otro para Android) puede ser un desafío y requerir recursos adicionales.
3. Menor alcance inicial: Al desarrollar aplicaciones nativas para plataformas específicas, tu audiencia potencial inicial es más limitada que con una aplicación web o híbrida.
4. Proceso de aprobación de tiendas: Las aplicaciones nativas deben pasar por un proceso de revisión y aprobación en las tiendas de aplicaciones, lo que puede llevar tiempo y, en algunos casos, resultar en rechazos.
5. Dependencia de las actualizaciones de la tienda: Las actualizaciones de la aplicación pueden tardar en estar disponibles para los usuarios, ya que dependen de la aprobación de la tienda de aplicaciones correspondiente.

**3.Que es una aplicación nativa, hibrida y multiplataforma**

**Aplicación nativa:**

Una aplicación nativa es una aplicación móvil diseñada específicamente para una plataforma o sistema operativo particular, como iOS (para dispositivos Apple) o Android.

Se desarrolla utilizando lenguajes de programación y herramientas nativas de la plataforma, como Swift o Objective-C para iOS y Java o Kotlin para Android.

Ofrece un alto rendimiento y acceso completo a las características y API del dispositivo.

Suele proporcionar una experiencia de usuario optimizada para la plataforma en la que se ejecuta.

Puede distribuirse a través de las tiendas de aplicaciones, como la App Store de Apple o Google Play Store de Android.

**Aplicación híbrida:**

Una aplicación híbrida es una combinación de tecnologías web (HTML, CSS, JavaScript) y componentes nativos que se ejecutan en una "envoltura" nativa.

Utiliza frameworks de desarrollo híbrido, como Apache Cordova (anteriormente conocido como PhoneGap) o React Native, que permiten escribir código una vez y ejecutarlo en múltiples plataformas.

A menudo, la mayoría de la interfaz de usuario y la lógica de la aplicación se escriben en tecnologías web y se envuelven en una capa nativa que proporciona acceso a características específicas del dispositivo.

Puede ser más rápido y rentable para desarrollar una aplicación híbrida, ya que se puede utilizar un solo código base para múltiples plataformas.

**Aplicación multiplataforma:**

Una aplicación multiplataforma es un tipo de aplicación que se desarrolla con la intención de ejecutarse en múltiples plataformas, como iOS, Android y a veces otros sistemas operativos.

Por lo general, se utiliza un marco de desarrollo multiplataforma, como Flutter, Xamarin o React Native, para escribir una vez el código y luego compilarlo en aplicaciones nativas para diferentes plataformas.

Ofrece un equilibrio entre la eficiencia de desarrollo y el rendimiento, ya que permite reutilizar una parte significativa del código, al tiempo que proporciona acceso a las características nativas del dispositivo.

A menudo, se utiliza para reducir los costos de desarrollo y acelerar el tiempo de llegada al mercado al permitir que los desarrolladores utilicen un solo conjunto de habilidades para múltiples plataformas.

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

Sí, estoy de acuerdo en que el diseño debe estar presente en todo el ciclo de desarrollo de software. El diseño es una parte fundamental y esencial del desarrollo de software por varias razones:

Mejora de la usabilidad y experiencia del usuario: El diseño centrado en el usuario garantiza que las aplicaciones sean fáciles de usar y atractivas para los usuarios finales. El diseño de la interfaz de usuario (UI) y la experiencia del usuario (UX) son cruciales para el éxito de cualquier aplicación, ya que afectan directamente a la satisfacción del usuario.

Ahorro de tiempo y recursos: Si se incorpora el diseño desde el principio, se pueden identificar y abordar problemas de usabilidad y diseño de manera más eficiente. Esto evita retrabajo y costos adicionales en etapas posteriores del desarrollo.

Coherencia en la apariencia y funcionalidad: El diseño garantiza que la aplicación tenga una apariencia y un comportamiento coherentes en todas sus partes, lo que proporciona una experiencia de usuario más uniforme y profesional.

Facilita la comunicación entre equipos: El diseño proporciona una representación visual de la aplicación que facilita la comunicación entre diseñadores, desarrolladores y otros miembros del equipo. Todos pueden tener una visión clara de cómo debe verse y funcionar la aplicación.

Evaluación de conceptos y prototipos: El diseño permite la creación de prototipos y maquetas que pueden ser evaluados por usuarios y partes interesadas antes de invertir en el desarrollo completo. Esto ayuda a identificar problemas tempranos y realizar ajustes.

Adaptación a cambios y actualizaciones: Durante el ciclo de vida de una aplicación, pueden surgir cambios en los requisitos o actualizaciones. El diseño flexible facilita la adaptación a estos cambios de manera más sencilla y rápida.

Mejora la calidad del software: Un diseño bien pensado y planificado puede contribuir a una arquitectura de software más sólida y a un código más limpio, lo que a su vez reduce la probabilidad de errores y problemas técnicos.

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

Empatizar: En esta etapa, se busca comprender profundamente a los usuarios y sus necesidades. Esto implica la observación, la entrevista y la empatía con las personas que enfrentan el problema. El objetivo es obtener información valiosa sobre los desafíos y las perspectivas de los usuarios.

Definir: Con base en la empatía, se define el problema de manera clara y específica. Se crean definiciones de problemas basadas en la información recopilada en la etapa de empatía. Esta definición debe ser precisa y enfocada en el usuario.

Idear: En esta etapa, se generan ideas para solucionar el problema identificado. Se fomenta la creatividad y la generación de soluciones a través de sesiones de lluvia de ideas y técnicas de pensamiento divergente. El objetivo es explorar una amplia gama de posibles soluciones.

Prototipar: Se construyen prototipos o modelos de las soluciones más prometedoras. Estos prototipos pueden ser simples representaciones de la idea que permiten probarla y obtener comentarios. La prototipación ayuda a visualizar las soluciones y a identificar posibles problemas antes de la implementación completa.

Probar: Los prototipos se prueban con usuarios reales para evaluar su eficacia y recopilar comentarios. La retroalimentación de los usuarios se utiliza para refinar y mejorar las soluciones. Si es necesario, se repiten las etapas anteriores para iterar y perfeccionar la solución.

ejemplo:

Problema: Imagine que estamos trabajando en una empresa de entregas y queremos mejorar la eficiencia de la entrega de paquetes en zonas urbanas congestionadas.

Empatizar: Entrevistamos a repartidores, conductores y destinatarios de paquetes. Observamos su experiencia en la entrega, sus desafíos y frustraciones al navegar por calles congestionadas.

Definir: Definimos el problema como "la entrega ineficiente de paquetes en zonas urbanas congestionadas". Identificamos los factores clave, como el tráfico y la dificultad para encontrar direcciones.

Idear: Realizamos una sesión de lluvia de ideas y generamos ideas, como rutas más eficientes, el uso de tecnología de navegación avanzada o la entrega en horarios no pico.

Prototipar: Creamos un prototipo de una aplicación móvil que integra mapas en tiempo real, rutas óptimas y notificaciones en tiempo real para repartidores y destinatarios.

Probar: Probamos la aplicación con repartidores y destinatarios reales en una zona urbana congestionada. Recopilamos sus comentarios y observamos la mejora en la eficiencia de la entrega.

**6.¿Que es Design Thinking?**

Design Thinking es una metodología o enfoque de resolución de problemas centrado en el usuario que se utiliza para abordar problemas complejos y desarrollar soluciones innovadoras. Esta metodología se ha popularizado en el ámbito del diseño, la innovación y el desarrollo de productos, pero su aplicabilidad se extiende a una variedad de campos, incluyendo la gestión empresarial, la educación y la resolución de problemas sociales.

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

Etapa 1: Empatizar

En esta etapa, el objetivo es comprender a fondo a los usuarios y sus necesidades. Se buscan insights, se observa su comportamiento y se establece empatía con ellos. Esta etapa implica:

Realizar entrevistas con los usuarios para conocer sus experiencias y desafíos.

Observar el entorno en el que los usuarios interactúan con el producto o servicio.

Escuchar atentamente sus preocupaciones, deseos y necesidades.

Ejemplo: Supongamos que un equipo de diseño está abordando el problema de mejorar la experiencia de los usuarios en una aplicación de entrega de alimentos a domicilio. En la etapa de empatizar, el equipo realiza entrevistas con repartidores, restaurantes asociados y clientes para comprender sus experiencias y desafíos.

Etapa 2: Definir

En esta etapa, se sintetiza la información recopilada en la etapa de empatizar para definir de manera precisa y clara el problema a resolver. Se crea una "declaración de problema" que guiará el proceso de diseño.

Identificar patrones y tendencias en los datos recopilados durante la etapa de empatizar.

Definir el problema de manera específica y enfocada en el usuario.

Ejemplo: Tras analizar las entrevistas y observaciones, el equipo de diseño podría definir el problema como "Los clientes a menudo se sienten frustrados debido a la falta de visibilidad sobre el estado de su pedido y la demora en las entregas, lo que reduce la satisfacción del usuario".

Etapa 3: Idear

En esta etapa, se fomenta la creatividad y la generación de ideas. Se busca explorar una amplia gama de soluciones potenciales, sin juzgar ni descartar ninguna idea en esta etapa.

Realizar sesiones de lluvia de ideas en equipo.

Utilizar técnicas de pensamiento divergente para generar ideas innovadoras.

Ejemplo: El equipo de diseño podría llevar a cabo una sesión de lluvia de ideas para encontrar soluciones que mejoren la visibilidad del estado de los pedidos y reduzcan las demoras en las entregas. Las ideas pueden incluir desde notificaciones en tiempo real hasta opciones de seguimiento de repartidores en el mapa.

Etapa 4: Prototipar

En esta etapa, se construyen prototipos o modelos de las soluciones más prometedoras. Los prototipos son representaciones simples de las ideas que permiten probarlas y obtener comentarios.

Crear prototipos de baja fidelidad (por ejemplo, dibujos o maquetas) o prototipos de alta fidelidad (simulaciones interactivas).

Los prototipos deben ser lo suficientemente simples como para iterar y experimentar sin invertir demasiado tiempo o recursos.

Ejemplo: El equipo podría crear prototipos de pantalla para la aplicación que muestren cómo se verían las nuevas características, como notificaciones en tiempo real y seguimiento de repartidores.

Etapa 5: Probar

En esta etapa, se prueban los prototipos con usuarios reales para evaluar su eficacia y recopilar comentarios. La retroalimentación de los usuarios se utiliza para refinar y mejorar las soluciones.

Observar a los usuarios mientras interactúan con los prototipos.

Registrar sus comentarios y reacciones.

Evaluar si las soluciones abordan eficazmente el problema definido en la etapa 2.

Ejemplo: El equipo realiza pruebas de usabilidad con clientes reales que utilizan los prototipos de la aplicación. Recopilan comentarios sobre la usabilidad, la eficacia de las nuevas características y la satisfacción del usuario.

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

Ser un diseñador inclusivo implica crear productos, servicios, espacios y experiencias de manera que sean accesibles y acogedores para todas las personas, independientemente de su género, edad, origen étnico, habilidades, discapacidades, orientación sexual, identidad de género o cualquier otra característica personal. El diseño inclusivo busca eliminar barreras y discriminación, promover la equidad y garantizar que nadie quede excluido o marginado debido a diferencias individuales.

La importancia del diseño inclusivo radica en varios aspectos:

Acceso igualitario: El diseño inclusivo garantiza que todos tengan igualdad de oportunidades para acceder y utilizar productos y servicios esenciales en la sociedad, como sitios web, aplicaciones móviles, transporte público y más.

Diversidad: La sociedad es diversa, y el diseño inclusivo refleja esta diversidad. Al considerar una amplia gama de perspectivas y necesidades, se pueden crear soluciones más efectivas y útiles.

Ética y responsabilidad: Diseñar de manera inclusiva es un acto ético que demuestra respeto por la dignidad y los derechos de todas las personas. También es una responsabilidad social y legal en muchas jurisdicciones.

Ampliación de la audiencia: Al considerar una variedad de necesidades, un producto o servicio puede atraer a una audiencia más amplia, lo que puede aumentar la base de usuarios o clientes.

Innovación: La inclusión puede ser una fuente de innovación. Al abordar desafíos de diseño relacionados con la inclusión, los diseñadores pueden descubrir nuevas soluciones y oportunidades que benefician a todos.

Competitividad: Las organizaciones que adoptan el diseño inclusivo pueden destacarse en el mercado y ganar ventaja competitiva al mostrar su compromiso con la diversidad y la igualdad.

si estoy de acuerdo en que debemos ser diseñadores inclusivos. El diseño inclusivo es esencial para construir un mundo más equitativo y justo, donde todos tengan la oportunidad de participar plenamente en la sociedad. Es una cuestión de derechos humanos y justicia social. Además, el diseño inclusivo puede dar lugar a soluciones más efectivas y creativas. En resumen, la inclusión es un valor fundamental en el diseño y debe ser un objetivo en la planificación y desarrollo de productos y servicios.

**9. herramientas que podemos utilizar para ser diseñadores inclusivos, Escoge 2 herramientas y realiza una investigación de como funciona y como se puede implementar en una aplicación web y una móvil.**

Herramienta 1: Wave

Wave es una herramienta de evaluación de accesibilidad que puede utilizarse para sitios web y aplicaciones móviles. La herramienta proporciona una puntuación de accesibilidad para cada elemento de la interfaz de usuario, así como sugerencias para mejorar la accesibilidad.

Cómo funciona

Wave funciona analizando la estructura HTML de un sitio web o aplicación. La herramienta identifica los elementos que pueden causar problemas de accesibilidad, como texto que no es accesible con lectores de pantalla, imágenes sin texto alternativo y enlaces que no son accesibles para usuarios con discapacidades motoras.

Ejemplos de problemas de accesibilidad que Wave puede identificar

Texto que no es accesible con lectores de pantalla: Esto puede incluir texto que no tiene un tamaño de fuente lo suficientemente grande, texto que no tiene un contraste suficiente o texto que se encuentra en una imagen.

Imágenes sin texto alternativo: El texto alternativo es una descripción del contenido de una imagen que se proporciona para usuarios que no pueden ver imágenes.

Enlaces que no son accesibles para usuarios con discapacidades motoras: Esto puede incluir enlaces que son demasiado pequeños o que no tienen un texto de enlace descriptivo.

Cómo implementar Wave en una aplicación web

Para implementar Wave en una aplicación web, simplemente agregue el siguiente código a la cabecera de su documento HTML:

HTML

<script src="https://wave.webaim.org/js/wave.min.js"></script>

Utiliza el código con precaución. Más información

A continuación, puede llamar a la función wave() para iniciar una evaluación de accesibilidad:

JavaScript

wave();

Cómo implementar Wave en una aplicación móvil

Para implementar Wave en una aplicación móvil, debe utilizar la API de Wave. La API proporciona una serie de funciones que puede utilizar para evaluar la accesibilidad de su aplicación.

Para obtener más información sobre cómo implementar Wave en una aplicación móvil, consulte la documentación de la API de Wave.

Ejemplos de cómo Wave puede utilizarse para mejorar la accesibilidad de un sitio web o aplicación móvil

Para mejorar la accesibilidad de un sitio web con mucho texto, Wave puede sugerir aumentar el tamaño de fuente o el contraste del texto.

Para mejorar la accesibilidad de una aplicación móvil con muchas imágenes, Wave puede sugerir agregar texto alternativo a todas las imágenes.

Para mejorar la accesibilidad de una aplicación web con muchos enlaces, Wave puede sugerir agregar un texto de enlace descriptivo a todos los enlaces.

Conclusión

Wave es una herramienta poderosa que puede ayudar a los diseñadores a crear sitios web y aplicaciones más accesibles. La herramienta es gratuita y fácil de implementar, y puede utilizarse para identificar y corregir problemas de accesibilidad en una variedad de dispositivos y plataformas.

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

La empatía con el usuario es de suma importancia en el desarrollo de software por varias razones cruciales:

Comprensión profunda de las necesidades del usuario: La empatía permite a los desarrolladores entender las necesidades, deseos y desafíos reales de los usuarios. Al ponerse en el lugar del usuario, los desarrolladores pueden identificar problemas que quizás no serían evidentes de otra manera.

Diseño centrado en el usuario: La empatía es un pilar del diseño centrado en el usuario. Al comprender a quiénes se destina el software y cómo lo usarán, los desarrolladores pueden crear interfaces de usuario más intuitivas y funcionales.

Mejora de la experiencia del usuario: Cuando los desarrolladores se preocupan por la experiencia del usuario, pueden diseñar software que sea fácil de usar y que satisfaga las necesidades del usuario de la manera más efectiva. Esto se traduce en una experiencia más agradable y productiva para los usuarios.

Reducción de la fricción: La empatía ayuda a identificar puntos de fricción en la interacción del usuario con el software. Al abordar estos puntos de fricción, se mejora la eficiencia y la satisfacción del usuario.

Fomento de la lealtad del usuario: Los usuarios tienden a valorar y preferir el software que se siente diseñado pensando en ellos. La empatía en el diseño y desarrollo del software puede fomentar la lealtad de los usuarios y la recomendación del producto a otros.

Identificación de problemas de accesibilidad: La empatía con el usuario también implica la consideración de personas con discapacidades. Al ponerse en el lugar de los usuarios con discapacidades, los desarrolladores pueden identificar y abordar problemas de accesibilidad, garantizando que el software sea utilizable para todos.

Feedback valioso: La empatía permite a los desarrolladores escuchar y valorar el feedback de los usuarios. Esto es esencial para la mejora continua del software, ya que los usuarios son una fuente invaluable de información sobre cómo el software está funcionando en el mundo real.

Reducción de la brecha entre diseñadores y desarrolladores: La empatía puede mejorar la comunicación y la colaboración entre diseñadores y desarrolladores. Cuando ambos equipos entienden las necesidades y preocupaciones del usuario, pueden trabajar juntos de manera más efectiva para crear un producto exitoso.

**11. Que es y como se implementa un mapa de empatía.**

Un mapa de empatía es una herramienta visual que se utiliza para comprender de manera más profunda a los usuarios, sus necesidades, deseos, pensamientos, sentimientos, frustraciones y comportamientos. Es una herramienta ampliamente utilizada en el diseño centrado en el usuario y la investigación de usuarios para ayudar a los equipos a crear soluciones que estén más alineadas con las experiencias de los usuarios reales.

Paso 1: Definir al usuario o segmento de usuario

Identifica claramente a quiénes están dirigidos tus esfuerzos de empatía. Esto puede ser un usuario específico o un segmento de usuarios con características similares. Cuanto más específico puedas ser, mejor.

Paso 2: Reúne información

Recopila información sobre tus usuarios a través de investigaciones, entrevistas, observaciones y cualquier otro método que te ayude a comprender sus necesidades y experiencias. Esta información debe ser lo más detallada y precisa posible.

Paso 3: Crea el mapa de empatía

Dibuja un cuadro grande en una hoja de papel o utiliza software de diagramación para crear un mapa de empatía. Divide el cuadro en secciones que representen diferentes aspectos de la experiencia del usuario. Las secciones comunes suelen incluir:

Ver: Aquí puedes incluir información sobre lo que el usuario ve en su entorno, como su lugar de trabajo, su hogar o el lugar donde utiliza tu producto o servicio.

Oír: Registra lo que los usuarios escuchan, como conversaciones, anuncios, sonidos del entorno, etc.

Pensar y sentir: Describe lo que los usuarios pueden estar pensando y sintiendo en diferentes situaciones relacionadas con tu producto o servicio.

Decir y hacer: Anota lo que los usuarios dicen en voz alta y cómo se comportan en diversas situaciones.

Dolores: Identifica los problemas, frustraciones y preocupaciones que experimentan los usuarios.

Beneficios: Registra los deseos, necesidades y objetivos que tienen los usuarios y lo que les gustaría obtener de tu producto o servicio.

Paso 4: Llena el mapa

Llena cada sección del mapa de empatía con la información recopilada en el paso 2. Utiliza palabras clave, frases y notas para representar los pensamientos, sentimientos, necesidades y comportamientos de los usuarios.

Paso 5: Analiza y actúa

Una vez que hayas completado el mapa de empatía, analiza la información y busca patrones o insights que te ayuden a comprender mejor a tus usuarios. Utiliza estos insights para guiar el diseño de productos o servicios que estén más alineados con las necesidades y deseos de los usuarios.

**12. ¿Que es un diagrama de contexto? ¿Y que es el context canvas?**

Un diagrama de contexto y el Context Canvas son herramientas que se utilizan en el diseño y la gestión de proyectos para representar de manera visual el entorno y los elementos que rodean a un sistema, proceso o proyecto en un contexto más amplio. Aunque son conceptos similares, cada uno se utiliza en diferentes contextos y con un enfoque ligeramente diferente:

**Diagrama de Contexto:**

Un diagrama de contexto es una representación gráfica que muestra un sistema (o proceso) en el centro y cómo se relaciona con su entorno o contexto circundante. Este diagrama se utiliza comúnmente en el ámbito de la ingeniería de software y la modelización de sistemas. En un diagrama de contexto, el sistema central se representa como un círculo o un rectángulo, y se muestran las "entidades externas" o "actores" que interactúan con el sistema en forma de flechas que conectan el sistema con estas entidades externas. Estas entidades pueden ser usuarios, otros sistemas, bases de datos, o cualquier elemento que interactúe con el sistema central. El diagrama de contexto ayuda a comprender las entradas y salidas del sistema y cómo se relaciona con su entorno.

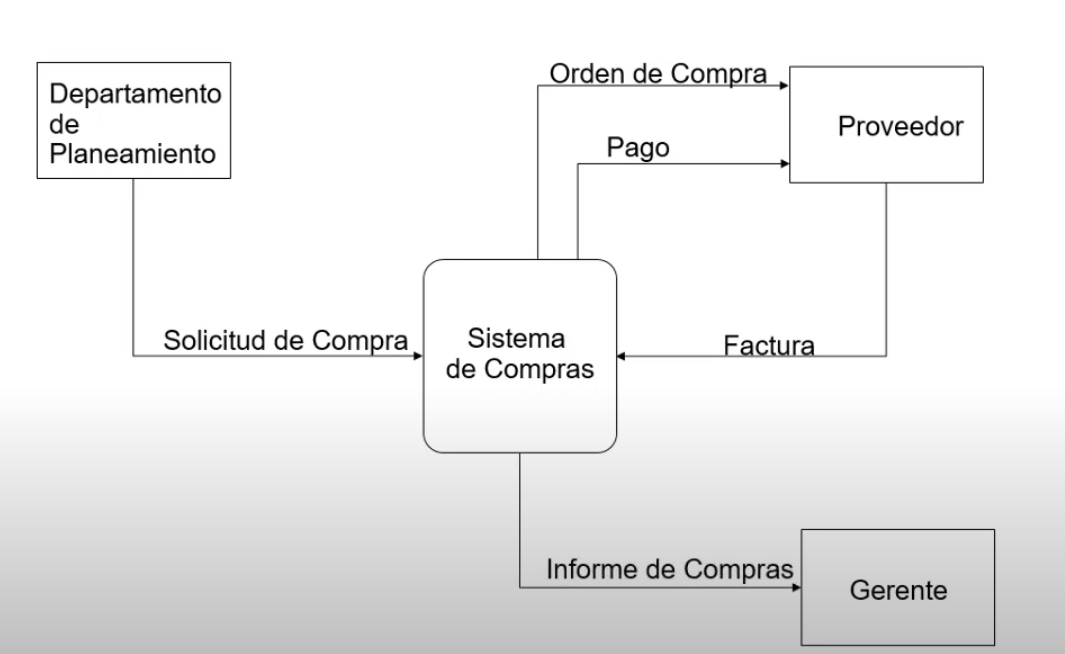
**Context Canvas:**

El Context Canvas es una herramienta utilizada en la metodología Lean Canvas, que se enfoca en la creación y validación de modelos de negocio. El Context Canvas se utiliza para comprender el contexto más amplio en el que se encuentra un producto o proyecto. A diferencia del diagrama de contexto, el Context Canvas se centra en la visión del negocio y busca comprender los aspectos clave que influyen en la estrategia y ejecución de un proyecto o producto. El Context Canvas incluye secciones para identificar socios clave, recursos clave, actividades clave, segmentos de clientes, relaciones con los clientes, canales, costos y fuentes de ingresos.

|  |  |  |
| --- | --- | --- |
| Aspecto | Diagrama de Contexto | Context Canvas |
| Propósito | Representar las interacciones de un sistema o proceso con su entorno. | Comprender el contexto más amplio en el que se encuentra un proyecto o producto desde una perspectiva de negocio. |
| Enfoque | Se centra en las relaciones del sistema con entidades externas o actores. | Se centra en aspectos de negocio, estrategia y ejecución, como socios, recursos, actividades, segmentos de clientes, etc. |
| Utilizado en | Ingeniería de software y modelización de sistemas. | Desarrollo y gestión de proyectos, en particular en la metodología Lean Canvas. |
| Representación gráfica | Sistema central (círculo o rectángulo) con flechas que conectan a las entidades externas. | Secciones y cuadros que describen elementos clave del negocio, organizados de manera visual. |
| Elementos clave | Sistema central y entidades externas. | Socios clave, recursos clave, actividades clave, segmentos de clientes, relaciones con los clientes, canales, costos y fuentes de ingresos. |
| Enfoque en el negocio | Principalmente se enfoca en la interacción técnica del sistema con su entorno. | Se enfoca en aspectos de negocio, estrategia, recursos y relaciones con clientes. |
| Ejemplos de uso | Representar cómo un sistema de reservas de hoteles se conecta con clientes y bases de datos. | Identificar los socios clave, recursos esenciales y canales para un nuevo servicio de entrega de alimentos. |

**13. .Elabora un ejemplo de diagrama de contexto y context canvas**

**Contexto:**

****

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

La arquitectura de datos es un marco estructurado que define cómo se recopilan, almacenan, gestionan, procesan y utilizan los datos en una organización. Consiste en un conjunto de reglas, estándares y directrices que determinan la estructura y organización de los datos para garantizar su disponibilidad, calidad, integridad y seguridad. La importancia de la arquitectura de datos radica en varios aspectos:

Gestión eficiente de datos: Ayuda a garantizar que los datos estén bien organizados y sean fáciles de gestionar. Esto incluye la definición de bases de datos, sistemas de almacenamiento y herramientas para administrar datos de manera efectiva.

Calidad de datos: La arquitectura de datos establece estándares para la calidad de los datos, lo que asegura que los datos sean precisos, consistentes y confiables.

Integración de datos: Facilita la integración de datos de múltiples fuentes y sistemas en toda la organización, lo que permite una visión más completa y coherente de la información.

Seguridad de datos: Define medidas de seguridad y control de acceso para proteger los datos contra amenazas y asegurar su confidencialidad e integridad.

Acceso a datos: Garantiza que los datos estén disponibles para aquellos que los necesitan, al tiempo que controla quién tiene acceso a qué información.

Apoyo a la toma de decisiones: Una arquitectura de datos bien diseñada proporciona una base sólida para la generación de informes y análisis que respaldan la toma de decisiones informadas en la organización.

Eficiencia operativa: Optimiza los procesos relacionados con la gestión de datos, lo que ahorra tiempo y recursos en la organización.

Cumplimiento normativo: Ayuda a cumplir con regulaciones y estándares de privacidad de datos, lo que evita sanciones legales y protege la reputación de la empresa.

Innovación tecnológica: La arquitectura de datos puede preparar a una organización para la adopción de tecnologías emergentes como la inteligencia artificial, el aprendizaje automático y el análisis de big data.

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

Marco de trabajo CRISP-DM (Cross-Industry Standard Process for Data Mining):

Explicación: CRISP-DM es un marco de trabajo ampliamente utilizado que consta de seis fases principales: comprensión del negocio, comprensión de los datos, preparación de datos, modelado, evaluación y despliegue. Cada fase tiene sus tareas y actividades específicas.

Importancia: Este enfoque estructurado permite a los equipos de análisis de datos abordar proyectos de manera sistemática, desde la comprensión de los objetivos comerciales hasta la implementación de modelos y soluciones.

Modelo de ciclo de vida de proyectos de datos:

Explicación: Este modelo considera tres fases: preprocesamiento, modelado y postprocesamiento. En la fase de preprocesamiento, se recopilan, limpian y preparan los datos. En la fase de modelado, se crean y evalúan los modelos de análisis de datos. En la fase de postprocesamiento, se interpretan los resultados y se comunican.

Importancia: Ayuda a los equipos a centrarse en aspectos específicos de las diferentes etapas del proyecto, asegurando que los datos se manejen adecuadamente y que los resultados se utilicen de manera efectiva.

Método CRISP-DM aplicado al Big Data:

Explicación: Una adaptación del marco CRISP-DM para abordar proyectos de datos a gran escala, que tienen desafíos únicos relacionados con el volumen, la variedad y la velocidad de los datos.

Importancia: Ofrece una guía específica para la gestión y análisis de datos masivos, lo que es crucial en el contexto de big data y análisis de datos a gran escala.

Método OSEMN (Obtención, Scrubbing, Exploración, Modelado y Evaluación):

Explicación: OSEMN es un enfoque para el diseño de proyectos de análisis de datos que se divide en cinco fases: obtención de datos, limpieza de datos, exploración de datos, modelado de datos y evaluación de modelos.

Importancia: Proporciona una estructura clara para el procesamiento y análisis de datos, desde la adquisición inicial hasta la evaluación de modelos.

Marco de trabajo TDSP (Team Data Science Process):

Explicación: Desarrollado por Microsoft, el TDSP se compone de seis fases: planificación del proyecto, obtención y limpieza de datos, exploración de datos, ingeniería de características, desarrollo de modelos y despliegue.

Importancia: Proporciona una metodología completa para proyectos de ciencia de datos que incluye la colaboración entre equipos, una comunicación efectiva y una orientación hacia la producción de resultados.

Enfoque Agile en proyectos de análisis de datos:

Explicación: Aplicar los principios ágiles en proyectos de análisis de datos implica la colaboración cercana con las partes interesadas, la adaptación a cambios y la entrega de resultados en incrementos iterativos.

Importancia: Ayuda a los equipos a mantenerse flexibles y receptivos a medida que se desarrollan proyectos de análisis de datos, lo que es especialmente útil cuando los requisitos pueden cambiar a lo largo del tiempo.

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

Los datos, la información y el conocimiento son conceptos relacionados pero distintos en el ámbito de la gestión de la información y el aprendizaje. Aquí te proporciono definiciones y un ejemplo para cada uno:

Datos: Los datos son hechos o cifras brutos sin procesar que representan eventos, mediciones o descripciones. Los datos por sí solos carecen de significado y contexto. Pueden ser números, palabras, imágenes o cualquier tipo de información.

Información: La información es el resultado del procesamiento y organización de los datos, lo que les da significado y contexto. La información es más estructurada y útil para la toma de decisiones o la comprensión de un tema específico.

Conocimiento: El conocimiento es un nivel más alto de comprensión y sabiduría que se deriva de la información y la experiencia. Implica la capacidad de interpretar, relacionar y aplicar la información en situaciones relevantes. El conocimiento implica la comprensión profunda y la habilidad de utilizar información de manera efectiva. Ejemplo: Un médico que utiliza su conocimiento para diagnosticar y tratar una enfermedad basándose en información médica.

Ejemplo:

Ejemplo completo:

Supongamos que estás gestionando un negocio de ventas de productos en línea. Aquí está cómo se aplicarían estos conceptos:

Datos: Los datos serían los puntos de datos brutos recopilados en tu negocio, como registros de ventas individuales, números de inventario, direcciones de envío y fechas de compra.

Información: La información se derivaría de esos datos cuando los organices y proceses. Por ejemplo, podrías analizar los datos de ventas mensuales y descubrir que el mes pasado se vendieron 1000 productos en total, con una facturación de $10,000.

Conocimiento: El conocimiento se basaría en la información y la experiencia acumulada. Si observas que las ventas aumentan cada diciembre debido a la temporada de compras navideñas, puedes usar ese conocimiento para planificar promociones especiales en diciembre y maximizar tus ganancias.

**17. ¿Como se debe construir un motor de análisis?**

La construcción de un motor de análisis (también conocido como motor de procesamiento analítico) implica la creación de un sistema o plataforma que puede recopilar, procesar y analizar datos para obtener información y conocimiento valiosos. A continuación, te proporciono una guía general para construir un motor de análisis:

**Definición de objetivos y requisitos:**

Comienza por comprender claramente los objetivos del motor de análisis. ¿Qué tipo de datos deseas analizar? ¿Cuáles son los resultados deseados? ¿Qué problemas o preguntas se intentan abordar?

Define los requisitos, incluyendo la fuente de datos, los tipos de análisis que se realizarán y los informes que se generarán.

**Recopilación y almacenamiento de datos:**

Determine las fuentes de datos y establece un proceso para recopilar y almacenar datos de manera efectiva. Esto puede incluir bases de datos, sistemas de archivos, APIs y otros canales de entrada.

Considera el uso de tecnologías de almacenamiento de datos como bases de datos relacionales, bases de datos NoSQL, sistemas de archivos distribuidos, almacenes de datos en la nube, etc.

**Limpieza y preparación de datos:**

Los datos crudos generalmente requieren limpieza y transformación. Esto implica la eliminación de valores atípicos, la normalización, la estandarización y la integración de datos de diferentes fuentes.

Utiliza herramientas de limpieza y transformación de datos, y define un proceso de ETL (Extract, Transform, Load) para automatizar estas tareas.

**Modelado y análisis de datos:**

Elige las técnicas y algoritmos de análisis de datos adecuados para tus necesidades. Esto puede incluir análisis estadísticos, minería de datos, aprendizaje automático, análisis de series temporales, etc.

Implementa modelos analíticos y algoritmos en el motor de análisis.

**Visualización y presentación de resultados:**

Crea visualizaciones efectivas para comunicar los resultados del análisis. Puedes utilizar herramientas de visualización de datos, como gráficos, tablas y paneles de control interactivos.

Diseña informes y paneles que muestren de manera clara y concisa los hallazgos y conclusiones.

**Optimización y escalabilidad:**

A medida que los volúmenes de datos aumentan, el motor de análisis debe ser escalable. Considera la optimización de consultas y procesos para manejar grandes conjuntos de datos de manera eficiente.

Aprovecha tecnologías como la computación en la nube y el procesamiento en clúster para escalar según sea necesario.

**Seguridad y privacidad:**

Implementa medidas de seguridad para proteger los datos y la información generada por el motor de análisis.

Asegúrate de cumplir con las regulaciones de privacidad de datos y los estándares de seguridad.

**Pruebas y validación:**

Realiza pruebas exhaustivas del motor de análisis para garantizar que los resultados sean precisos y confiables.

Valida los modelos y algoritmos para asegurarte de que estén cumpliendo con los objetivos establecidos.

**Despliegue y mantenimiento:**

Implementa el motor de análisis en un entorno de producción.

Establece un proceso de mantenimiento continuo para mantener los datos actualizados y garantizar que el motor siga siendo eficiente y efectivo.

**Monitoreo y retroalimentación:**

Implementa sistemas de monitoreo para rastrear el rendimiento y la eficacia del motor de análisis en tiempo real.

Utiliza la retroalimentación y las métricas para realizar ajustes y mejoras continuas.

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

Los escenarios funcionales y de calidad son dos tipos de escenariosutilizados en la ingeniería de software y el diseño de sistemas para definir y evaluar aspectos específicos de un producto o sistema. A continuación, te proporciono un cuadro comparativo entre ambos:

**Escenarios Funcionales:**

Definición: Los escenarios funcionales se centran en las funciones y características específicas que un sistema debe proporcionar. Describen cómo el sistema debe responder a ciertas acciones del usuario o eventos.

Enfoque: Se centran en lo que el sistema debe hacer y cómo debe comportarse en términos de funcionalidad y características específicas.

Ejemplos: Ejemplos de escenarios funcionales podrían incluir cómo un sistema de gestión de inventario debe permitir la creación de nuevos elementos, cómo debe calcular el total de una compra en un sistema de punto de venta, etc.

Evaluación: Se utilizan para verificar si el sistema cumple con las expectativas funcionales y si las funciones específicas se implementan correctamente.

**Escenarios de Calidad:**

Definición: Los escenarios de calidad se centran en los atributos de calidad del sistema, como el rendimiento, la seguridad, la usabilidad, la escalabilidad y la confiabilidad.

Enfoque: Se centran en cómo el sistema debe desempeñarse en términos de aspectos no funcionales, como la eficiencia, la seguridad, la facilidad de uso, la capacidad de respuesta, etc.

Ejemplos: Ejemplos de escenarios de calidad incluyen cómo el sistema debe manejar una carga de trabajo alta sin degradación del rendimiento, cómo debe proteger los datos confidenciales de los usuarios, cómo debe ser fácil de usar para diferentes tipos de usuarios, etc.

Evaluación: Se utilizan para evaluar la calidad del sistema en términos de atributos no funcionales y para garantizar que cumple con los estándares de calidad y rendimiento.

|  |  |  |
| --- | --- | --- |
| Aspecto | Escenarios Funcionales | Escenarios de Calidad |
| Enfoque | Funciones y características específicas. | Atributos de calidad y desempeño no funcional. |
| Definición | Describen cómo el sistema debe comportarse. | Describen cómo el sistema debe desempeñarse. |
| Ejemplos | Creación de elementos, cálculo de totales. | Rendimiento bajo carga, seguridad de datos. |
| Evaluación | Verifica si las funciones se implementan. | Evalúa la calidad del sistema en atributos no funcionales. |

**19. Explica que es la GUI y el material design.**

GUI (Interfaz Gráfica de Usuario):

La GUI, o Interfaz Gráfica de Usuario, es un componente esencial de cualquier software o aplicación que permite a los usuarios interactuar con un sistema informático de manera visual y gráfica. En una GUI, los elementos visuales como ventanas, botones, menús desplegables, iconos y campos de entrada proporcionan una forma intuitiva para que los usuarios interactúen con el software. En lugar de depender de comandos de texto o líneas de código, los usuarios pueden realizar acciones utilizando gestos de ratón, toques en pantallas táctiles y clics en elementos visuales.

Las GUI se utilizan en una amplia variedad de dispositivos y sistemas, desde aplicaciones de software en computadoras personales y dispositivos móviles hasta interfaces de usuario en electrodomésticos, cajeros automáticos, sistemas de entretenimiento y más. La GUI se ha convertido en un estándar en la informática moderna debido a su facilidad de uso y accesibilidad, lo que permite a personas con diversos niveles de experiencia interactuar con la tecnología de manera efectiva.

Material Design:

Material Design es un sistema de diseño visual desarrollado por Google que se utiliza principalmente en aplicaciones y servicios de software para crear interfaces de usuario coherentes y atractivas. Material Design se basa en principios de diseño que se inspiran en la apariencia y el tacto del papel y los materiales físicos. Algunas de las características clave de Material Design incluyen:

Diseño visual limpio: Material Design se centra en la simplicidad y la claridad en la presentación de información. Utiliza capas, sombras y animaciones para dar profundidad y realismo a los elementos visuales.

Usabilidad: Material Design se preocupa por la experiencia del usuario y proporciona pautas claras para la navegación, la jerarquía de contenido y la interacción intuitiva.

Consistencia: Material Design promueve la coherencia en la apariencia y el comportamiento de las aplicaciones, lo que ayuda a los usuarios a sentirse cómodos y familiarizados con ellas.

Adaptabilidad a múltiples dispositivos: Material Design se adapta a diferentes dispositivos y tamaños de pantalla, lo que garantiza que las aplicaciones tengan una apariencia y un funcionamiento óptimos en una variedad de plataformas.

Iconos y tipografía: Material Design incluye un conjunto de iconos y fuentes tipográficas específicas que se utilizan para mantener una apariencia coherente en las aplicaciones.

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

UI (User Interface - Interfaz de Usuario):

La UI, o Interfaz de Usuario, se refiere a la parte visual y tangible de una aplicación o sitio web con la que los usuarios interactúan directamente. Incluye elementos como botones, menús, iconos, ventanas, colores, fuentes y otros elementos visuales que los usuarios ven y utilizan para navegar y realizar acciones en la aplicación. La UI se centra en cómo se presenta la información y cómo los usuarios interactúan con ella.

Ejemplo de UI: Imagina que estás diseñando una aplicación de redes sociales. La UI incluiría la disposición de los botones "Me gusta", "Comentar" y "Compartir", así como la apariencia visual de los perfiles de usuario, las imágenes de portada y los colores del diseño. Una buena UI haría que estos elementos fueran atractivos y fáciles de usar.

UX (User Experience - Experiencia de Usuario):

La UX, o Experiencia de Usuario, se centra en la experiencia general del usuario al interactuar con una aplicación o sitio web. Incluye la facilidad de uso, la eficiencia, la satisfacción y la utilidad de la aplicación. La UX se preocupa por cómo se siente el usuario al utilizar la aplicación, si logra sus objetivos de manera eficiente y si tiene una experiencia agradable en el proceso.

Ejemplo de UX: Siguiendo con la aplicación de redes sociales, la UX se centraría en asegurarse de que los usuarios puedan navegar por la aplicación de manera intuitiva, encontrar fácilmente contenido relevante, disfrutar de una velocidad de carga rápida de imágenes y experimentar una interacción suave. Un buen diseño de UX se aseguraría de que los usuarios tengan una experiencia fluida y agradable al utilizar la aplicación.

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

Bocetos Manuales:

Exploración Rápida de Ideas: Los bocetos manuales permiten a los diseñadores plasmar ideas de manera rápida y sin restricciones. Pueden esbozar múltiples conceptos y soluciones en papel antes de comprometerse con un diseño específico.

Comunicación Visual: Los bocetos son una forma efectiva de comunicar ideas a otras personas, como miembros del equipo, partes interesadas y clientes. Pueden transmitir la visión de diseño de manera clara y visual.

Iteración y Refinamiento: Los bocetos facilitan la iteración y el refinamiento de ideas. Los diseñadores pueden realizar modificaciones y ajustes fácilmente en papel antes de pasar a etapas más costosas del diseño.

Enfoque en la Creatividad: Al eliminar la necesidad de preocuparse por detalles técnicos, los bocetos permiten a los diseñadores centrarse en la creatividad y la conceptualización de ideas.

Prototipos Digitales:

Interacción Simulada: Los prototipos digitales permiten simular la interacción real del usuario con la interfaz. Los usuarios pueden hacer clic en botones, desplazarse por las pantallas y experimentar la navegación, lo que proporciona una visión más precisa de cómo funcionará el producto final.

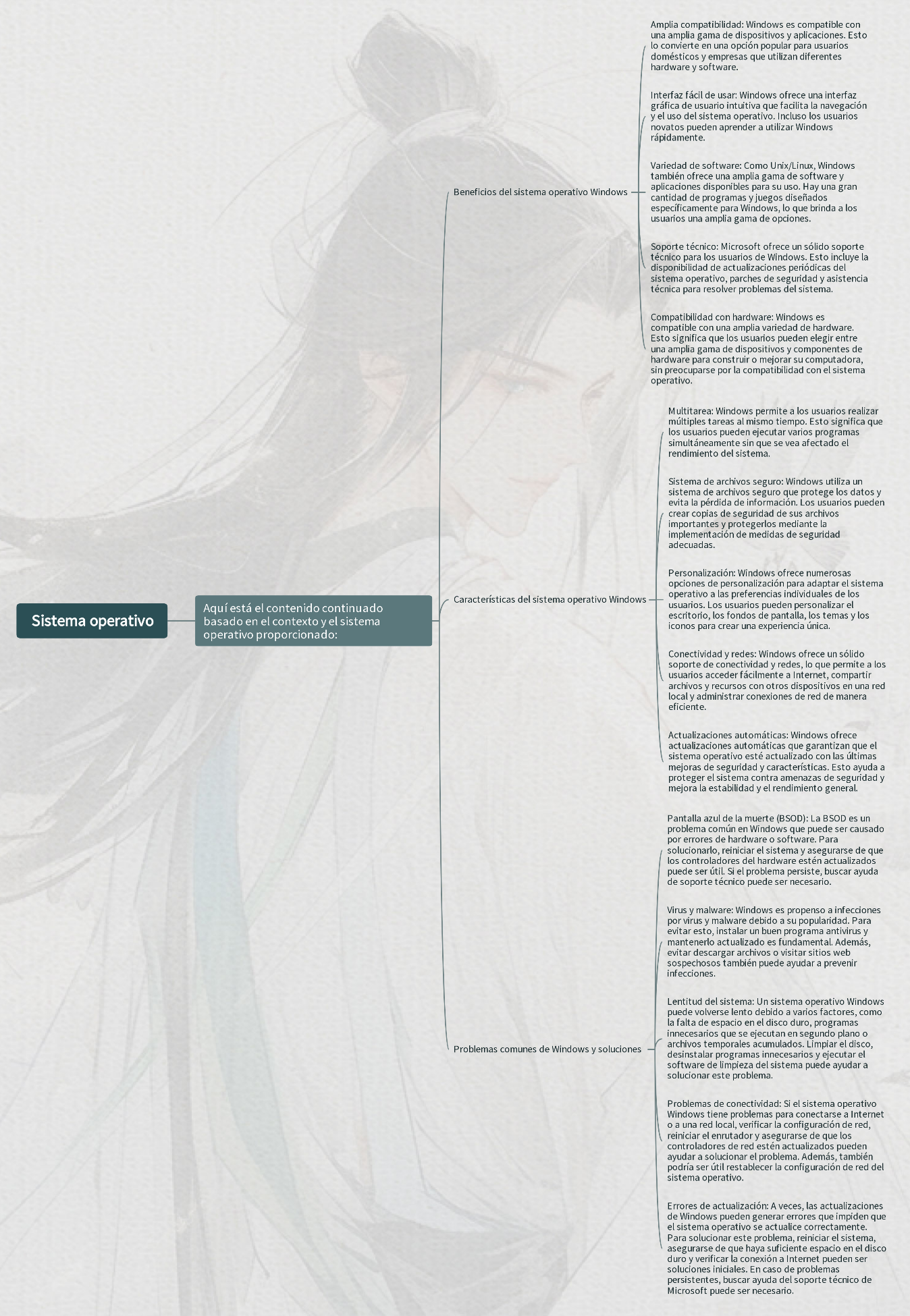
Prueba de Concepto: Los prototipos digitales son ideales para probar y validar conceptos de diseño. Los diseñadores y las partes interesadas pueden verificar la viabilidad de una idea antes de invertir en su desarrollo completo.

Retroalimentación Temprana: Al poner un prototipo en manos de usuarios reales, es posible obtener retroalimentación temprana y valiosa que puede guiar el diseño en una dirección más efectiva y satisfactoria para los usuarios.

Ahorro de Tiempo y Costos: Identificar problemas y realizar ajustes en las etapas de prototipado puede ahorrar tiempo y costos significativos en comparación con corregir problemas en una etapa más avanzada del desarrollo.

Colaboración Efectiva: Los prototipos digitales facilitan la colaboración entre miembros del equipo, diseñadores y desarrolladores, ya que todos pueden interactuar con un modelo interactivo del diseño.

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



**23. ¿Qué es el kernel?**

El kernel es una parte fundamental del sistema operativo de una computadora. Actúa como una interfaz entre el hardware de la computadora y el software de nivel de usuario, gestionando los recursos del sistema, permitiendo la comunicación entre las aplicaciones y el hardware, y controlando la ejecución de programas.

Las funciones clave del kernel incluyen:

Gestión de memoria: Controla cómo se asigna y libera la memoria física y virtual, asegurando que los programas no interfieran entre sí y que el sistema sea eficiente en la utilización de la memoria.

Gestión de procesos: Supervisa y coordina la ejecución de procesos y tareas en la computadora, asignando tiempo de CPU y recursos a cada uno.

Gestión de dispositivos: Administra la interacción con dispositivos de hardware, como discos duros, impresoras, teclados y pantallas, asegurando que los programas puedan comunicarse con ellos.

Gestión de archivos: Controla cómo se almacenan, se organizan y se acceden a los datos en el sistema de archivos, permitiendo la creación, modificación y eliminación de archivos y directorios.

Seguridad y control de acceso: Protege el sistema y los recursos al controlar quién puede acceder a qué recursos y qué operaciones pueden realizar.

Gestión de interrupciones: Maneja las interrupciones de hardware, como las señales de dispositivos, para garantizar que se atiendan adecuadamente sin causar problemas en el sistema.

Los kernels pueden variar en diseño y funcionalidad, y existen diferentes tipos de kernels, como los monolíticos (donde la mayoría de las funciones del sistema operativo se ejecutan en un solo espacio de memoria) y los microkernels (donde las funciones del sistema operativo se dividen en módulos independientes). Ejemplos de kernels populares incluyen el kernel de Linux, el kernel de Windows NT y el kernel de macOS.

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

**Android:**

Kernel: Android utiliza el kernel de Linux como su capa base. El kernel se encarga de las operaciones de bajo nivel, como la administración de memoria, el acceso al hardware, la gestión de procesos y la seguridad. Es la interfaz entre el hardware del dispositivo y las capas superiores del sistema.

Librerías del sistema: Esta capa incluye bibliotecas de software que proporcionan funcionalidades adicionales al sistema, como gráficos 2D y 3D, manipulación de datos, conectividad de red y más. Estas bibliotecas son utilizadas por las aplicaciones y servicios de nivel superior.

Entorno de ejecución de aplicaciones (Android Runtime): Android Runtime incluye el motor de ejecución Dalvik o, a partir de Android 5.0, el motor de ejecución ART (Android Runtime). Estos motores ejecutan el código de las aplicaciones y gestionan la compilación y ejecución de las aplicaciones escritas en lenguaje Java o en lenguajes compatibles con Java.

Marco de aplicaciones (Application Framework): El marco de aplicaciones proporciona un conjunto de componentes y servicios que permiten a los desarrolladores de aplicaciones crear aplicaciones de manera más fácil. Incluye elementos como la gestión de ventanas, notificaciones, servicios de ubicación y la gestión de contactos.

Aplicaciones: Esta es la capa superior del sistema operativo Android, donde residen las aplicaciones reales que los usuarios instalan desde Google Play u otras fuentes. Puede incluir aplicaciones preinstaladas (como el navegador web, el cliente de correo electrónico, etc.) y aplicaciones de terceros.

**iOS:**

Kernel: iOS utiliza un kernel personalizado basado en el kernel de Unix llamado "XNU" (X is Not Unix). Este kernel realiza tareas de bajo nivel, como la administración de la memoria, el acceso al hardware y la gestión de procesos.

Servicios del sistema: Esta capa incluye servicios esenciales que son fundamentales para el funcionamiento del sistema, como el sistema de archivos, la seguridad, la red y la gestión de energía. Además, también incluye el sistema de notificaciones y el servicio de localización.

Entorno de tiempo de ejecución (Runtime Environment): iOS utiliza Objective-C y Swift como lenguajes de programación para el desarrollo de aplicaciones. En esta capa, se proporcionan las bibliotecas y el tiempo de ejecución necesarios para ejecutar las aplicaciones escritas en estos lenguajes.

Cocoa Touch: Esta capa proporciona el conjunto de frameworks y bibliotecas que permiten el desarrollo de aplicaciones de usuario final. Incluye elementos para la creación de interfaces de usuario, gestión de eventos táctiles, gráficos y más.

Aplicaciones: En la capa superior se encuentran las aplicaciones que los usuarios utilizan en sus dispositivos iOS. Esto incluye aplicaciones preinstaladas, como Mensajes, Safari y Correo, así como aplicaciones descargadas desde la App Store.

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

Un framework es un conjunto de herramientas, bibliotecas y componentes predefinidos que proporcionan una estructura y guía para el desarrollo de software. Estas herramientas y componentes facilitan y agilizan el proceso de desarrollo al proporcionar funcionalidades comunes y estándares de diseño.

La importancia de los frameworks radica en que permiten a los desarrolladores ahorrar tiempo y esfuerzo al no tener que reinventar la rueda cada vez que inician un nuevo proyecto. Al utilizar un framework, los desarrolladores pueden aprovechar las mejores prácticas y soluciones probadas, lo que aumenta la eficiencia y la calidad del software desarrollado.

Ventajas de utilizar un framework:

1. Productividad: Los frameworks proporcionan una estructura y funcionalidades predefinidas que permiten a los desarrolladores enfocarse en la lógica específica de su aplicación, en lugar de tener que preocuparse por aspectos técnicos y repetitivos.

2. Reutilización de código: Los frameworks suelen ofrecer bibliotecas y componentes que pueden ser reutilizados en diferentes proyectos, lo que permite ahorrar tiempo y esfuerzo en el desarrollo.

3. Mantenibilidad: Al utilizar un framework, los desarrolladores siguen patrones de diseño y buenas prácticas, lo que facilita el mantenimiento del código a largo plazo.

4. Comunidad y soporte: Los frameworks suelen tener una comunidad activa de desarrolladores que comparten conocimientos, solucionan problemas y ofrecen soporte, lo que facilita el aprendizaje y la resolución de dudas.

Desventajas de utilizar un framework:

1. Curva de aprendizaje: Al utilizar un framework, los desarrolladores deben aprender su estructura y funcionalidades específicas, lo que puede llevar tiempo y esfuerzo.

2. Limitaciones: Los frameworks tienen ciertas limitaciones en términos de flexibilidad y personalización. En algunos casos, los desarrolladores pueden encontrar dificultades para implementar funcionalidades específicas que no estén cubiertas por el framework.

3. Dependencia: Al utilizar un framework, los desarrolladores se vuelven dependientes de las actualizaciones y soporte del mismo. Si el framework deja de ser mantenido o actualizado, puede generar problemas de compatibilidad y seguridad.