TECNICATURA SUPERIOR EN

Telecomunicaciones

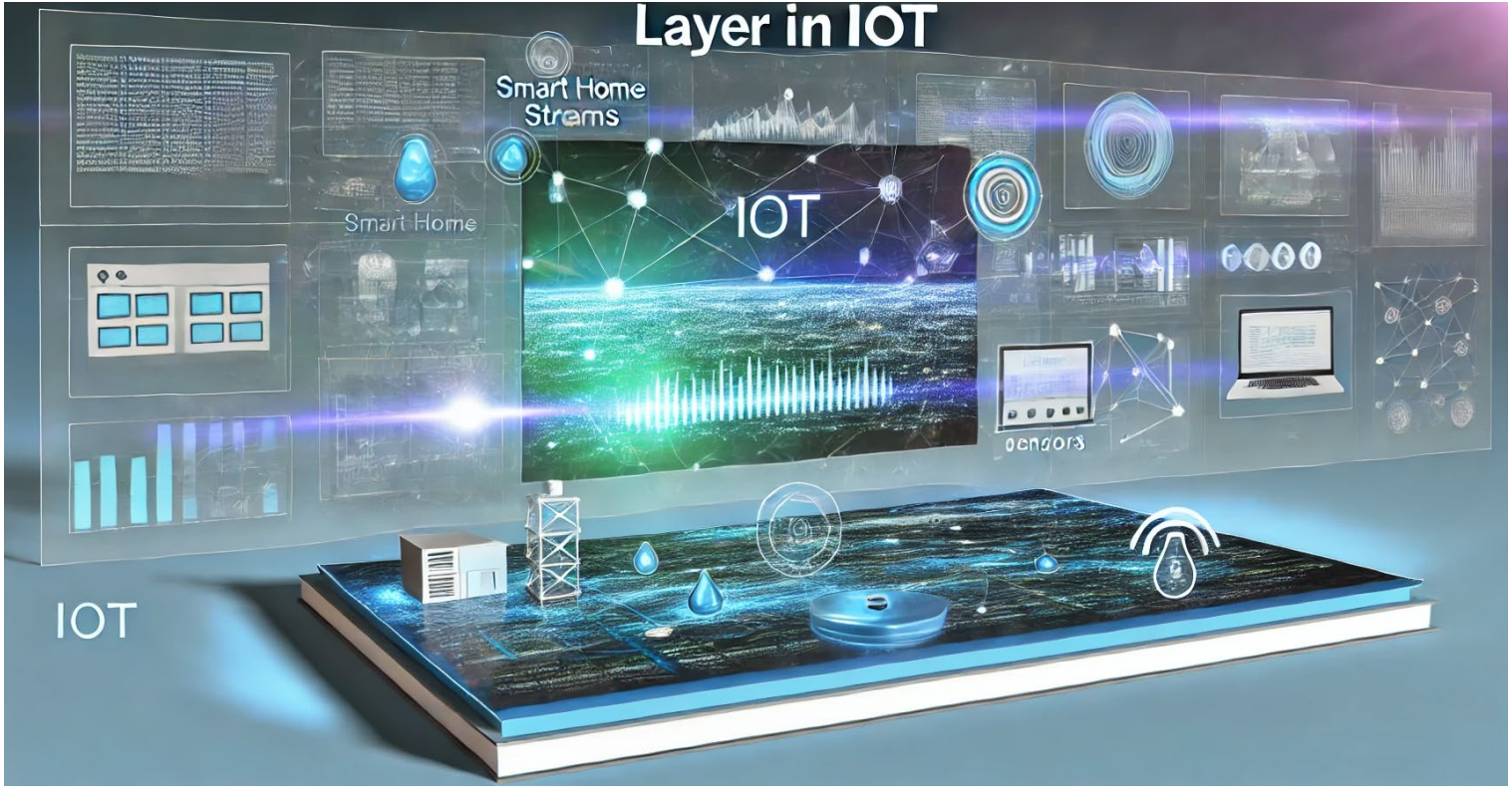
Proyecto Integrador I

Capa de presentación

ISPC
INSTITUTO SUPERIOR POLITÉCNICO CÓRDOBA

Dirección General de EDUCACIÓN TÉCNICA Y FORMACIÓN PROFESIONAL
Ministerio de EDUCACIÓN
CÓRDOBA entre todos

Capa de presentación



Capa de presentación

Definición de la Capa de Presentación en IoT

- La **capa de presentación** es la interfaz que conecta a los usuarios con el sistema IoT, permitiendo la visualización de datos y el control remoto de los dispositivos.
- Es la **interfaz gráfica** entre el usuario y el sistema IoT.
- Muestra los datos procesados por el sistema IoT en un formato comprensible.
- Facilita el **control de dispositivos** (actuadores, sensores) de manera remota.
- Es crucial para **interpretar** los datos generados por los sensores IoT.

Capa de presentación

Rol de la Capa de Presentación en IoT

- La capa de presentación cumple un rol clave en transformar datos técnicos en información comprensible y útil.
- **Visualización de datos** en tiempo real.
- **Interacción remota** con el sistema IoT para realizar acciones sobre los dispositivos.
- **Toma de decisiones** basada en datos, brindando acceso a la información necesaria para la operación eficiente del sistema.
- **Responsabilidad clave**: Conectar el procesamiento de datos con la interfaz que entiende el usuario.

Capa de presentación

Diferencias entre la Capa de Presentación y Otras Capas

- En un sistema IoT, cada capa tiene un rol específico. La capa de presentación se distingue porque su objetivo principal es permitir la interacción del usuario con el sistema, a diferencia de otras capas enfocadas en la captura o procesamiento de datos.
- **Percepción**: Recoge los datos con sensores.
- **Preprocesamiento**: Filtra y limpia los datos.
- **Conectividad**: Transporta los datos entre los dispositivos IoT.
- **Almacenamiento**: Guarda los datos en bases de datos.
- **Procesamiento**: Analiza los datos recogidos. (No confundir con análisis).
- **Presentación**: Hace accesible la información procesada al usuario, facilitando la visualización y control.

Capa de presentación

Importancia de la Experiencia de Usuario (UX) en IoT

- Una capa de presentación con un buen diseño de **Experiencia de Usuario (UX)** es clave para el éxito de un sistema IoT, ya que facilita la interacción y comprensión de los datos.
- **Accesibilidad**: La interfaz debe ser intuitiva y fácil de usar.
- **Eficiencia**: Permite la toma de decisiones rápida y precisa.
- **Escalabilidad**: Una buena UX permite que el sistema crezca y mantenga la usabilidad con más datos y dispositivos.
- Un mal diseño UX puede frustrar a los usuarios y reducir la efectividad del sistema IoT.

Capa de presentación

Impacto de la UX en la Interacción con IoT

- La UX tiene un impacto directo en la manera en que los usuarios interactúan con los sistemas IoT.
- Una interfaz clara y eficiente facilita la **toma de decisiones**.
- Un buen diseño asegura que los datos se presenten de manera comprensible, reduciendo errores operativos.
- Mejora la **satisfacción del usuario**, promoviendo la adopción de la tecnología.
- **UX deficiente**: puede causar dificultades para entender los datos, lo que lleva a decisiones incorrectas o retrasos.

Capa de presentación

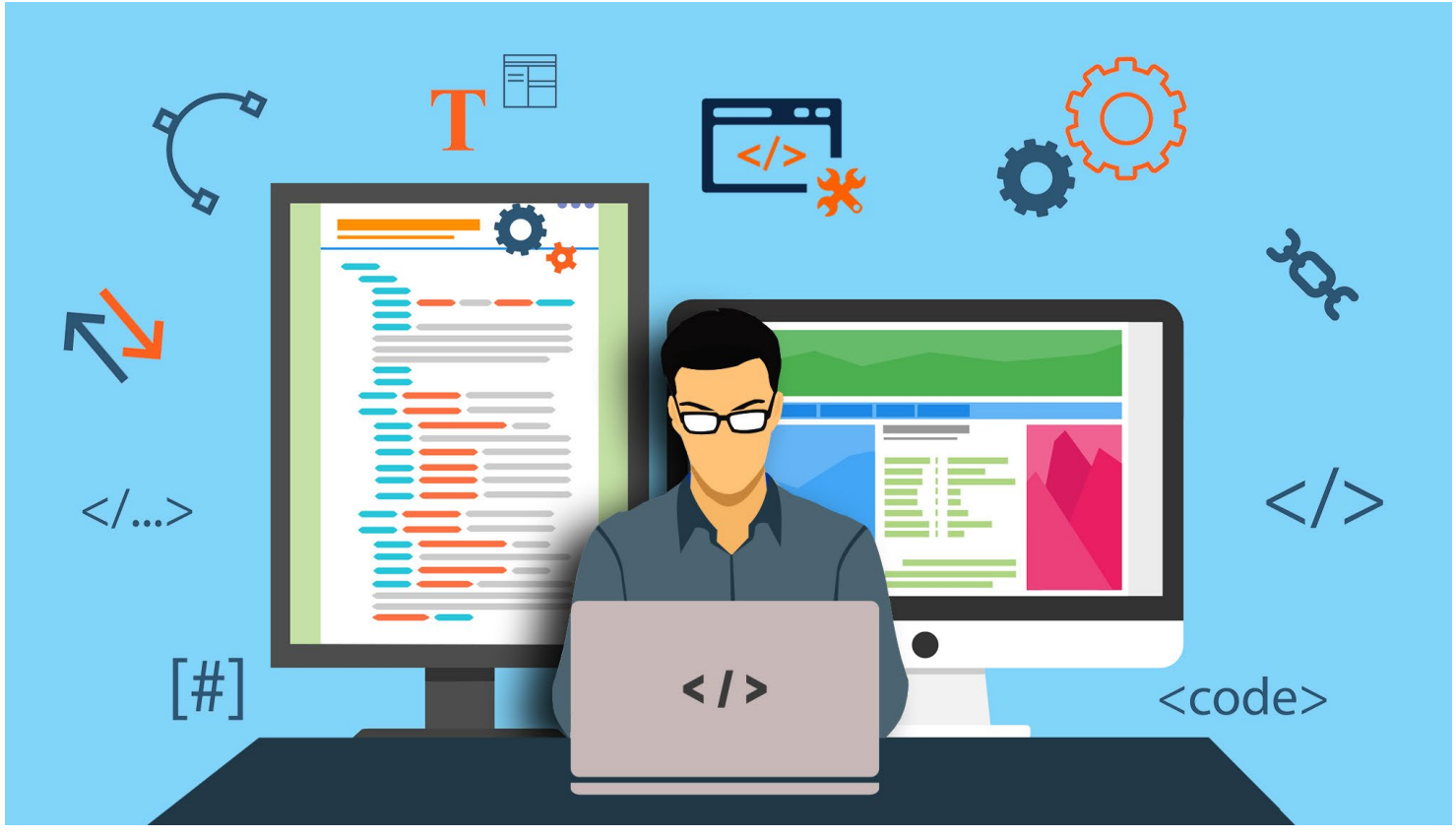
Interfaces Típicas en IoT

- Existen varias formas en que la capa de presentación puede implementarse en los sistemas IoT. Las más comunes son:

1. **App Web**:
   - Acceso desde navegadores web en cualquier dispositivo.
   - Actualizaciones fáciles y sin necesidad de instalación.
2. **App Móvil**:
   - Diseñadas para smartphones, con acceso directo a hardware como GPS y cámaras.
   - Descargables desde tiendas de apps (iOS, Android).
3. **Dashboards**:
   - Herramientas de visualización orientadas al monitoreo en tiempo real.
   - Ejemplos: **Grafana**, **Node-RED**.

Cada tipo de interfaz tiene ventajas y desventajas, según las necesidades del usuario y del sistema.

Capa de presentación



Capa de presentación

Enfoque Web-First

- El desarrollo web es una opción estratégica para iniciar proyectos IoT, ya que permite llegar a la mayor cantidad de usuarios sin comprometer la velocidad de implementación.
- **Accesibilidad**:
  - Disponible en cualquier dispositivo con navegador.
  - No requiere instalación ni descargas.
- **Costo y tiempo**:
  - Desarrollo más rápido y económico.
  - Actualizaciones automáticas y simultáneas.
- **Portabilidad**:
  - El código puede evolucionar y ser reutilizado en otros entornos (apps móviles, dashboards).

Ideal para proyectos que necesitan escalar rápidamente y abarcar múltiples dispositivos desde el inicio.

Capa de presentación

Comparación entre el Desarrollo Web y Móvil

- El desarrollo web y móvil tienen diferencias fundamentales en cuanto a alcance, implementación y acceso a hardware.

- **Desarrollo Web**:
  - Accesibilidad universal: compatible con cualquier navegador.
  - Menor costo y tiempo de desarrollo.
  - Actualización inmediata sin intervención del usuario.
  - Acceso limitado al hardware del dispositivo.

- **Desarrollo Móvil**:
  - Acceso directo a hardware: GPS, cámara, sensores.
  - Mejor rendimiento nativo.
  - Dependencia de tiendas de aplicaciones y necesidad de actualizaciones manuales.
  - Mayor esfuerzo y tiempo de desarrollo.

Capa de presentación

Ventajas de Empezar por la Web

- Comenzar con el desarrollo web en proyectos IoT ofrece una serie de ventajas prácticas y estratégicas.

- **Accesibilidad**:
  - Disponible para cualquier dispositivo con acceso a internet.
- **Desarrollo rápido**:
  - Menor tiempo de implementación en comparación con apps nativas.
- **Actualizaciones simples**:
  - Se aplican automáticamente para todos los usuarios.
- **Portabilidad del código**:
  - Fácil adaptación a aplicaciones móviles o dashboards sin duplicar esfuerzo.
- **Menos barreras de entrada**:
  - No es necesario distribuir la app en tiendas (App Store, Google Play).

Capa de presentación

Qué se Gana y Qué se Pierde

- El enfoque **web-first** tiene claros beneficios, pero también implica sacrificios en cuanto al uso de hardware nativo.
- **Qué se gana**:
- **Escalabilidad**: El sistema puede evolucionar y expandirse con facilidad.
- **Portabilidad**: Código reutilizable en otras plataformas.
- **Accesibilidad universal**: Todos los usuarios con un navegador pueden acceder.
- **Qué se pierde**:
- **Acceso a hardware nativo**: Funciones como cámara, GPS, sensores son limitadas en la web.
- **Rendimiento**: Las aplicaciones nativas suelen ser más rápidas y optimizadas para dispositivos móviles.

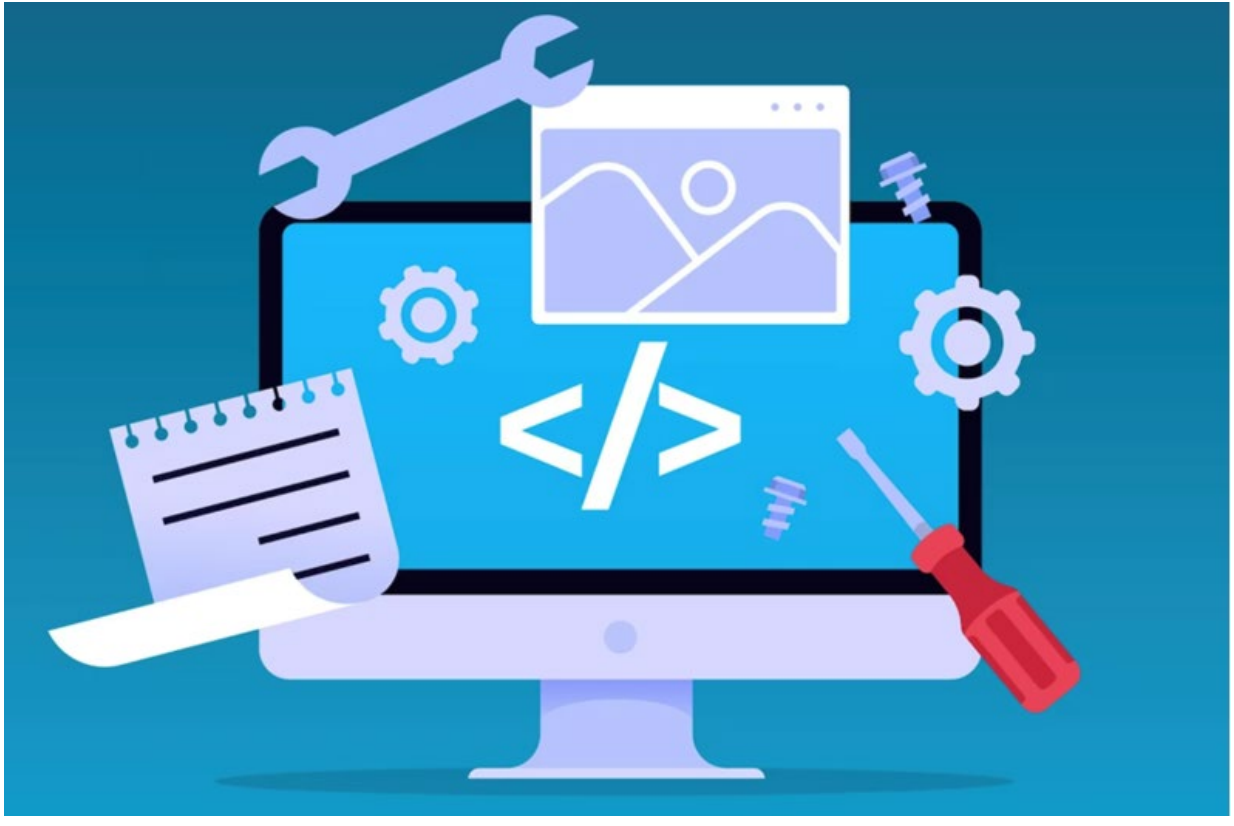
Capa de presentación

Escalabilidad y Flexibilidad

- Una de las principales razones para elegir el enfoque **web-first** es su capacidad para escalar y adaptarse a necesidades futuras.

- **Escalabilidad**:
  - Las aplicaciones web pueden crecer fácilmente añadiendo nuevas funcionalidades o más dispositivos.
- **Flexibilidad**:
  - El código puede ser adaptado y reutilizado para futuras aplicaciones móviles o dashboards.
- **Menor costo a largo plazo**:
  - Mantener una aplicación web es más económico en términos de desarrollo y actualizaciones.

Capa de presentación



Capa de presentación

Lenguajes y Herramientas Clave del Desarrollo Web

- El desarrollo de la capa de presentación web requiere una combinación de lenguajes y herramientas fundamentales para crear interfaces modernas, eficientes y responsivas.

- **HTML5**: Estructura básica de una página web.
- **CSS3**: Diseño y estilo, incluyendo diseño **responsivo**.
- **JavaScript**: Interactividad y manejo de eventos en la web.
- **Frameworks modernos**:
  - **React**: Librería de JavaScript para crear interfaces de usuario.
  - **Vue.js**: Framework progresivo para construir aplicaciones web interactivas.
  - **Angular**: Framework completo para crear aplicaciones empresariales robustas.

Capa de presentación

HTML5, CSS3 y JavaScript

- Estos lenguajes forman la base de cualquier desarrollo web y son indispensables para construir la capa de presentación.

- **HTML5**:
  - Proporciona la estructura del contenido.
  - Elementos semánticos para accesibilidad y SEO.
- **CSS3**:
  - Estiliza y organiza el diseño visual.
  - Asegura que las páginas sean **responsivas** y se adapten a diferentes tamaños de pantalla.
- **JavaScript**:
  - Permite añadir interactividad a las páginas web.
  - Gestiona eventos, manipulación del DOM y asincronía.

Capa de presentación

Frameworks Modernos: React, Vue.js, Angular

- Los **frameworks** y **librerías de JavaScript** modernos permiten construir interfaces dinámicas y manejables.

- **React**:
  - Basado en componentes reutilizables.
  - Gestión de estado a través de herramientas como **Redux** o el **Context API**.
  - Excelente para aplicaciones **Single Page Applications (SPA)**.
- **Vue.js**:
  - Framework progresivo, fácil de aprender e implementar.
  - Ideal para interfaces interactivas y escalables.
- **Angular**:
  - Framework completo basado en **TypeScript**.
  - Soporte para proyectos de gran escala con una arquitectura modular.
  - Utilizado comúnmente en aplicaciones empresariales.

Capa de presentación

Arquitectura de Aplicaciones Web: Single Page Applications (SPA)

- La arquitectura **Single Page Application (SPA)** permite a las aplicaciones web cargar solo una vez y actualizar dinámicamente su contenido sin recargar la página completa.

- **SPA**:
  - El contenido se actualiza sin recargar toda la página, mejorando la experiencia del usuario.
  - Utiliza **APIs** para obtener y enviar datos de forma dinámica.
  - Tecnologías como **React**, **Vue.js** y **Angular** son ideales para crear SPAs.
- **Ventajas**:
  - Carga rápida después de la primera solicitud.
  - Experiencia de usuario más fluida y similar a una aplicación nativa.

Capa de presentación

API REST y su Rol en la Comunicación

- Las **APIs REST** son esenciales para la comunicación entre el frontend (capa de presentación) y el backend (donde se procesan los datos).

- **Qué es una API REST**:
  - Conjunto de normas que permite la comunicación entre sistemas web a través de solicitudes HTTP.
  - Utiliza métodos como **GET**, **POST**, **PUT**, **DELETE**.
- **Rol en la Capa de Presentación**:
  - Proporciona datos del backend a la interfaz de usuario (frontend).
  - Permite acciones en tiempo real, como el control de dispositivos IoT.
- **Ventajas**:
  - Independencia entre frontend y backend.
  - Escalabilidad y flexibilidad en el desarrollo de aplicaciones.

Capa de presentación

Optimización y Diseño Responsivo

- El diseño **mobile-first** y **responsivo** asegura que la interfaz se vea y funcione bien en cualquier dispositivo, ya sea móvil, tablet o escritorio.

- **Diseño mobile-first**:
  - Prioriza la experiencia en dispositivos móviles antes de escalar a pantallas más grandes.
  - Mejora la **usabilidad** en pantallas pequeñas.
- **Técnicas responsivas**:
  - Uso de **media queries** en CSS para adaptar el diseño.
  - Frameworks como **Bootstrap** y **TailwindCSS** ayudan a crear diseños escalables y flexibles.
- **Optimización de performance**:
  - Minimizar recursos (imágenes, scripts).
  - Usar **CDNs** (Content Delivery Networks) para mejorar el tiempo de carga.

Capa de presentación

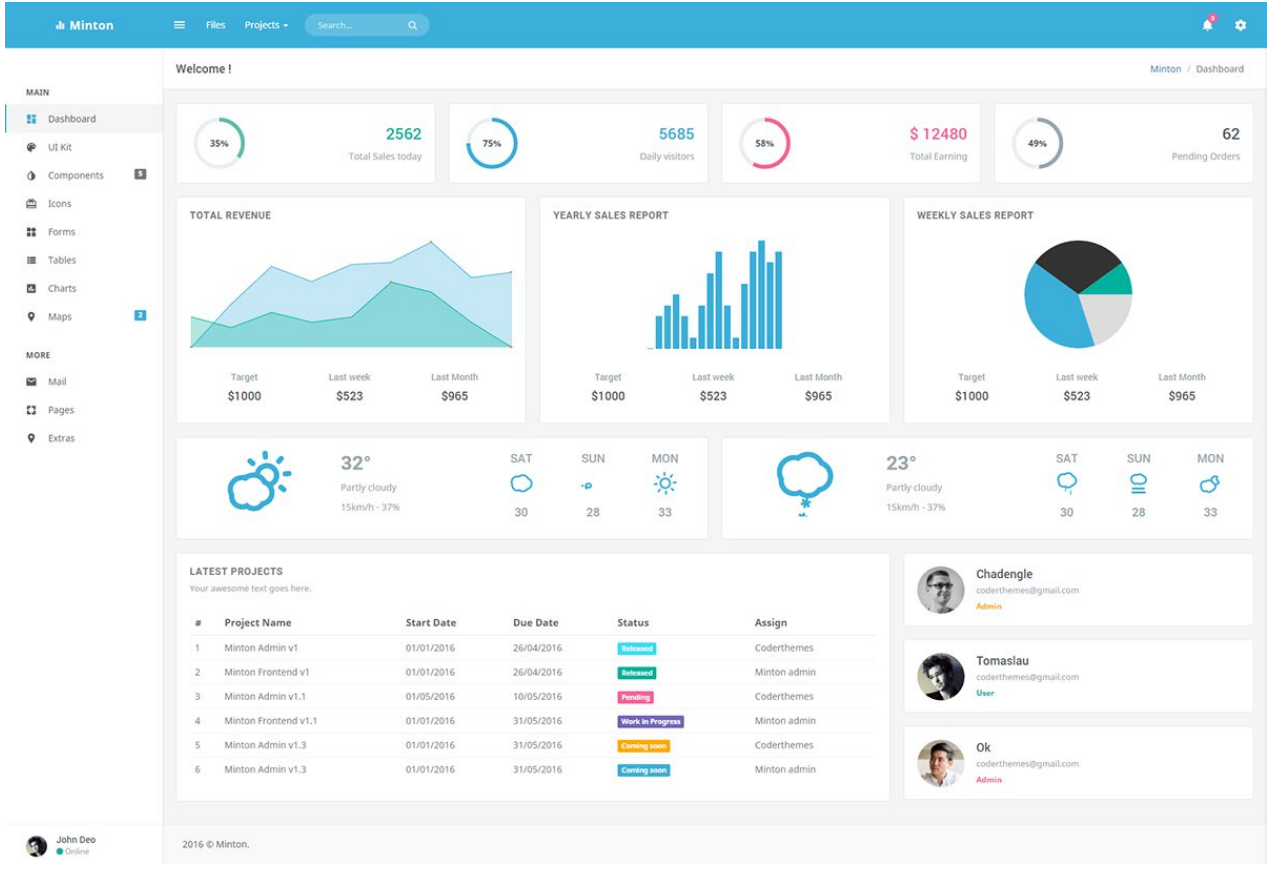
Herramientas para Garantizar Funcionalidad en Múltiples Dispositivos

- Existen herramientas y técnicas que garantizan que la aplicación web funcione en cualquier dispositivo.

- **Frameworks de diseño responsivo**:
  - **Bootstrap**: Proporciona una base de diseño predefinido para una fácil creación de interfaces.
  - **TailwindCSS**: Ofrece una mayor personalización de estilos sin depender de plantillas predefinidas.

- **Pruebas de compatibilidad**:
  - Uso de herramientas como **BrowserStack** para probar en múltiples navegadores y dispositivos.
  - Pruebas de rendimiento con **Lighthouse** para asegurar tiempos de carga rápidos.

Capa de presentación



Capa de presentación

Uso de Dashboards en la Capa de Presentación

- Los **dashboards** son herramientas visuales que permiten el monitoreo en tiempo real de los datos generados por los dispositivos IoT. Son una parte clave de la capa de presentación.
- **Qué son los dashboards**:
  - Interfaces gráficas que permiten visualizar y gestionar datos en tiempo real.
  - Orientados a mostrar métricas, gráficos y alertas de los sistemas IoT.
- **Funcionalidades clave**:
  - Mostrar el estado de sensores y dispositivos.
  - Facilitar la toma de decisiones basada en los datos.
  - Permitir el control remoto de dispositivos IoT desde la misma interfaz.

Capa de presentación

Cómo Funcionan los Dashboards en IoT

- Los dashboards funcionan como intermediarios entre los datos recolectados por los dispositivos IoT y el usuario final, permitiendo el acceso a la información de forma clara y organizada.

- **Función principal**:
  - Organizar y presentar los datos de sensores y dispositivos IoT de manera visual y comprensible.
- **Conexión con el backend**:
  - Los datos almacenados o procesados en el backend son presentados a través del dashboard.
- **Interactividad**:
  - Algunos dashboards permiten al usuario interactuar con los dispositivos en tiempo real (encender/apagar actuadores, ajustar configuraciones).

Capa de presentación

Herramientas de Dashboards Comunes en IoT

- Existen varias herramientas especializadas en la visualización de datos para IoT. Cada una ofrece diferentes funcionalidades y niveles de personalización.

- **Grafana**:
  - Ideal para la visualización de datos de series temporales.
  - Compatible con bases de datos como **InfluxDB** y **Prometheus**.
- **Node-RED**:
  - Herramienta de flujo visual que permite tanto la visualización como la gestión de datos IoT.
  - Orientado a la creación rápida de flujos y control de dispositivos IoT.
- **Kibana**:
  - Parte del stack ELK, orientado a la visualización de grandes volúmenes de datos (logs).
  - Utilizado principalmente en análisis de datos IoT.
- **Power BI**:
  - Plataforma de análisis de datos que permite crear reportes interactivos.
  - Puede integrarse con datos IoT para visualización avanzada.

Capa de presentación

Comparación de Herramientas:
Grafana, Node-RED, Kibana, Power BI

- Cada herramienta tiene ventajas y desventajas según el caso de uso y los requisitos del sistema IoT.

- **Grafana**:
  - Enfocado en visualización de métricas y series temporales.
  - Ideal para monitoreo en tiempo real de sensores IoT.
  - Muy configurable y soporta múltiples fuentes de datos.

Capa de presentación

- **Node-RED**:
  - Enfocado en flujos de trabajo visuales.
  - Permite tanto la gestión de datos como el control de dispositivos IoT.
  - Buena opción para prototipos rápidos y automatización.
- **Kibana**:
  - Orientado al análisis y visualización de grandes volúmenes de datos.
  - Más adecuado para datos históricos o logs que para series temporales en tiempo real.
- **Power BI**:
  - Plataforma completa de análisis de datos.
  - Útil para generar reportes avanzados, pero no está enfocado principalmente en el monitoreo en tiempo real.

Capa de presentación

Visualización de Datos en Tiempo Real

- Uno de los aspectos más importantes en IoT es la capacidad de visualizar datos en tiempo real para poder reaccionar de manera rápida a los cambios en el entorno.

- **Importancia de la visualización en tiempo real**:
  - Los datos de sensores IoT son valiosos solo si se pueden interpretar a tiempo.
  - Permite al usuario tomar decisiones informadas de manera inmediata.
- **Herramientas clave para la visualización en tiempo real**:
  - **Grafana**: Ideal para series temporales de sensores, con paneles que actualizan los datos continuamente.
  - **Node-RED**: Ofrece una interfaz visual para monitorizar datos y controlar dispositivos en tiempo real.
- **Monitoreo en sistemas críticos**:
  - IoT en fábricas, cultivos inteligentes, y sistemas de seguridad requieren la visualización constante para detectar fallas o problemas de inmediato.

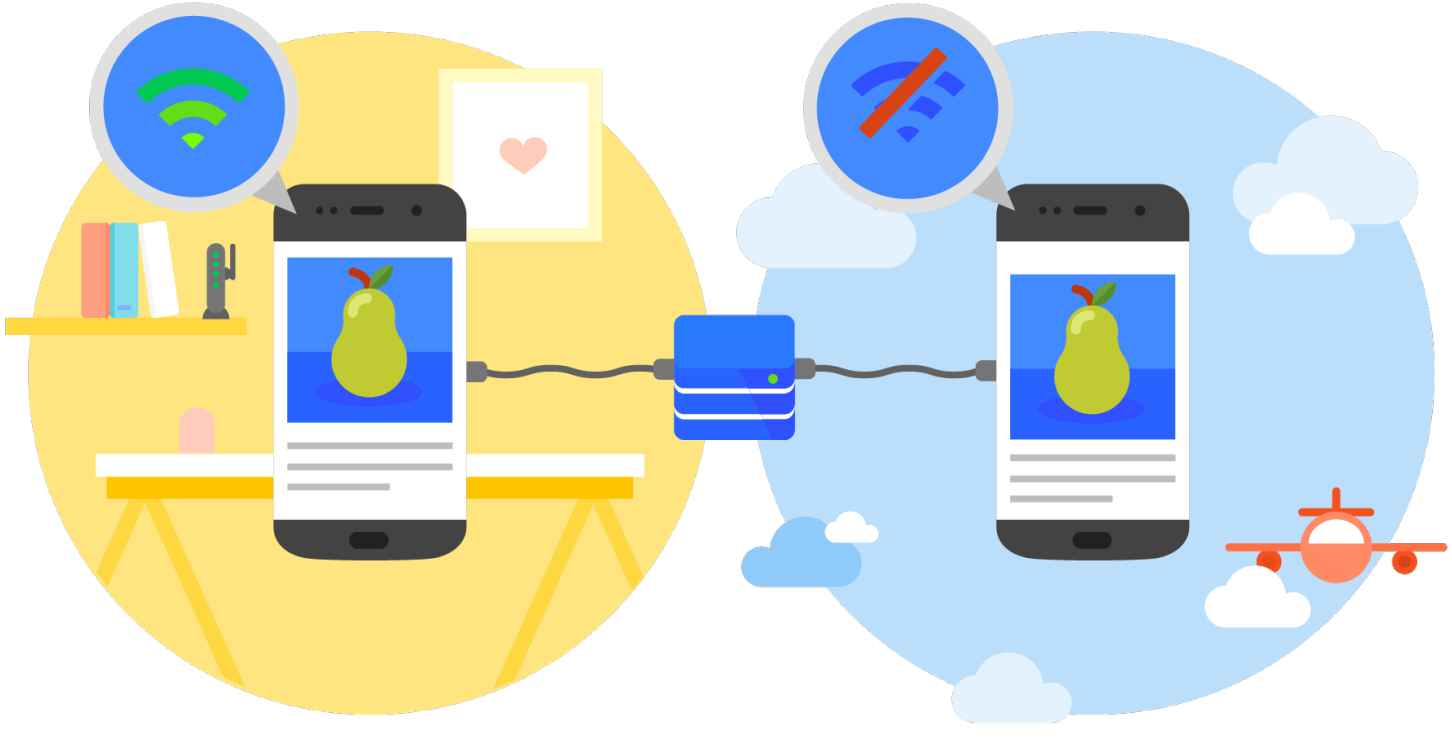
Capa de presentación

Conexión con Bases de Datos en Dashboards IoT

- Los dashboards se conectan a **bases de datos** que almacenan los datos de los dispositivos IoT. Estas conexiones permiten la actualización en tiempo real y el acceso a los datos históricos.

- **Bases de datos de series temporales**:
  - **InfluxDB**: Especializada en almacenar datos de sensores IoT con un enfoque en la eficiencia y la visualización en tiempo real.
  - **Prometheus**: Ideal para monitoreo de métricas y alertas.
- **Bases de datos tradicionales**:
  - **MySQL** y **PostgreSQL**: También pueden usarse para almacenar datos IoT, pero son más adecuadas para datos estructurados o históricos.
- **Integración de datos en dashboards**:
  - Grafana y Node-RED se integran fácilmente con bases de datos para mostrar los datos en tiempo real.
  - La configuración adecuada de la base de datos y su rendimiento es clave para asegurar que los dashboards funcionen sin problemas.

Capa de presentación



Capa de presentación

Progressive Web Apps (PWA)

- Las **Progressive Web Apps (PWA)** son aplicaciones web que combinan lo mejor de las aplicaciones móviles y web, permitiendo que una aplicación web se comporte como una aplicación nativa en dispositivos móviles.
- **Qué es una PWA**:
  - Aplicación web que puede ser instalada en dispositivos móviles y funcionar sin conexión.
  - Utiliza tecnologías web estándar como **HTML**, **CSS** y **JavaScript**.
- **Características principales**:
  - **Instalación desde el navegador**.
  - **Soporte offline**: Almacena datos en caché para funcionar sin conexión.
  - **Notificaciones push**.
  - **Experiencia similar a una app móvil**.

Capa de presentación

Ventajas de Utilizar PWA en la Capa de Presentación

- Las **PWAs** ofrecen una alternativa poderosa para crear una aplicación que puede funcionar tanto en web como en dispositivos móviles sin necesidad de desarrollo separado para cada plataforma.
- **Ventajas clave**:
  - **Instalación sin tienda de aplicaciones**: Los usuarios pueden instalar la app directamente desde el navegador.
  - **Menor costo y tiempo de desarrollo**: Un solo código que funciona tanto en dispositivos móviles como en escritorio.
  - **Soporte offline**: Continúa funcionando sin conexión, lo que es ideal para sistemas IoT.
  - **Notificaciones push**: Permite alertar al usuario de eventos importantes en el sistema IoT.
- **Ideal para**:
  - Proyectos que no requieren acceso profundo al hardware del dispositivo.
  - Aplicaciones que necesitan ser accesibles en varios dispositivos rápidamente.

Capa de presentación

Cómo Convertir una App Web en una PWA

- Convertir una aplicación web estándar en una **PWA** implica añadir ciertas funcionalidades para que pueda ser instalada y funcionar offline.
- **Service Workers**:
  - Script en segundo plano que permite manejar solicitudes y almacenar en caché para soportar el modo offline.
- **Manifest JSON**:
  - Archivo que define cómo la app aparecerá en la pantalla de inicio de un dispositivo, incluyendo iconos, colores y otras configuraciones visuales.
- **Caché**:
  - Controla los recursos y datos que se almacenarán en el dispositivo para permitir que la app funcione sin conexión.
- **Notificaciones Push**:
  - Permite enviar alertas a los usuarios incluso cuando no están utilizando la aplicación.

Capa de presentación

Frameworks Multiplataforma: Ionic y React Native

- Los **frameworks multiplataforma** permiten desarrollar aplicaciones que funcionan tanto en dispositivos móviles como en la web, utilizando un solo código base.

- **Ionic**:
  - Framework basado en tecnologías web (HTML, CSS, JS) que permite crear aplicaciones móviles y web.
  - Utiliza **Capacitor** o **Cordova** para acceder a las funcionalidades nativas de los dispositivos móviles (cámara, GPS, etc.).

Capa de presentación

- **React Native**:
  - Framework de JavaScript basado en **React** que permite desarrollar aplicaciones móviles nativas.
  - Utiliza componentes nativos en lugar de emular una interfaz web, lo que resulta en un mejor rendimiento en aplicaciones móviles.

- **Ventajas comunes**:
  - **Un solo código para múltiples plataformas**.
  - Reducción de costos de desarrollo y mantenimiento.
  - Acceso a funcionalidades nativas de los dispositivos móviles.

Capa de presentación

Ventajas de Frameworks Multiplataforma

- Los frameworks multiplataforma como **Ionic** y **React Native** permiten crear aplicaciones tanto para dispositivos móviles como para la web, reutilizando gran parte del código.

- **Ventajas**:
  - **Menor tiempo de desarrollo**: Un solo código base que funciona en múltiples plataformas.
  - **Acceso a hardware nativo**: Utilizando **Capacitor** (Ionic) o las APIs nativas de React Native, se puede acceder a funcionalidades como cámara, GPS, notificaciones, etc.
  - **Actualización más simple**: Las actualizaciones se pueden realizar en el mismo código base para todas las plataformas.
- **Cuándo utilizarlos**:
  - Ideal para proyectos IoT que necesitan acceso a dispositivos móviles y web, con funcionalidades nativas (notificaciones push, acceso a sensores).

Capa de presentación

Limitaciones de los Frameworks Multiplataforma

- Aunque ofrecen muchas ventajas, los frameworks multiplataforma también presentan algunas limitaciones en comparación con el desarrollo nativo.
- **Rendimiento**:
  - Las aplicaciones desarrolladas con frameworks multiplataforma suelen ser más lentas que las aplicaciones nativas, especialmente en tareas intensivas como gráficos y procesamiento de datos.
- **Acceso limitado a algunas APIs nativas**:
  - No todos los dispositivos y funcionalidades están completamente soportados. A veces se requiere desarrollar plugins adicionales o código nativo para acceder a ciertas características.
- **Dependencia de frameworks**:
  - El soporte y actualizaciones dependen de las comunidades y los mantenedores del framework, lo que puede generar problemas a largo plazo si el framework no se mantiene actualizado.

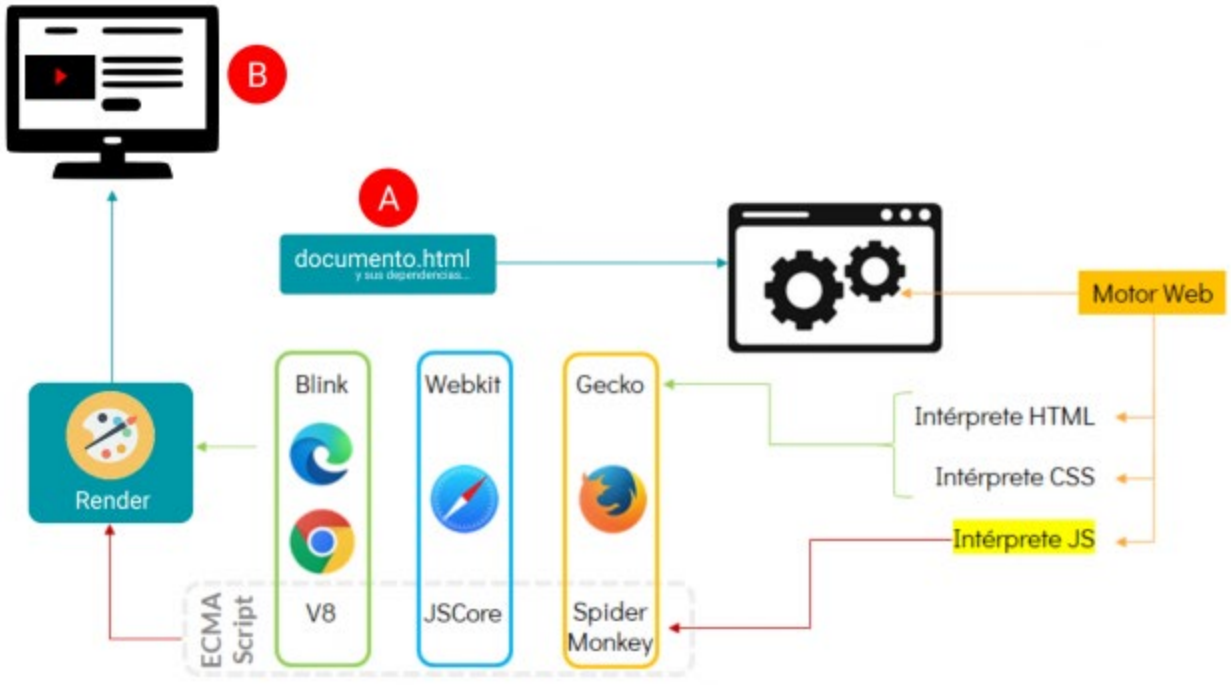
Capa de presentación

Aplicación de PWAs y Frameworks Multiplataforma en IoT

- La combinación de **PWAs** y **frameworks multiplataforma** puede expandir significativamente la capa de presentación en un sistema IoT, brindando flexibilidad y opciones de acceso desde múltiples dispositivos.

- **PWA para accesibilidad general**:
  - Ideal para aplicaciones ligeras que no requieren acceso profundo al hardware.
  - Ofrece soporte offline y puede ser instalada directamente desde el navegador.
- **Frameworks multiplataforma para funcionalidades avanzadas**:
  - Ionic o React Native permiten desarrollar apps móviles con acceso a hardware nativo.
  - Ideal para proyectos IoT que requieren control de dispositivos móviles o características específicas como cámaras o GPS.

Capa de presentación



Capa de presentación

Ejemplo Práctico de Implementación

- Para entender cómo funciona la capa de presentación en un sistema IoT, vamos a explorar un **ejemplo práctico**. Imaginemos que necesitamos desarrollar un panel de control web para monitorear y controlar sensores y actuadores en un sistema IoT.
- **Panel de control**:
  - Visualiza datos de sensores en tiempo real (temperatura, humedad, etc.).
  - Permite el control de actuadores (encender luces, activar riego).
- **Tecnologías clave**:
  - **Frontend**: **React.js** para la interfaz.
  - **Backend**: **Node.js** con **API REST** para manejar las solicitudes.
  - **Base de datos**: **InfluxDB** para almacenar datos en tiempo real.

Capa de presentación

Desarrollo del Panel de Control Web

- El desarrollo del **panel de control web** requiere una combinación de tecnologías y servicios que permitan la interacción entre la capa de presentación y los dispositivos IoT.
- **Frontend (React.js)**:
  - Interfaz web donde los usuarios pueden visualizar los datos de los sensores y controlar los actuadores.
  - Uso de componentes para mostrar los datos en tiempo real.
- **Backend (Node.js)**:
  - API REST que recibe los datos de los dispositivos IoT y los envía a la base de datos.
  - También permite que el usuario envíe comandos para controlar los actuadores.
- **Base de datos (InfluxDB)**:
  - Almacena datos de sensores en tiempo real, optimizado para series temporales.
  - Se conecta al dashboard para mostrar gráficos y métricas.

Capa de presentación

Uso de APIs REST para la Conexión entre Frontend y Backend

- La API REST actúa como un puente entre la capa de presentación y los datos de los dispositivos IoT, facilitando la comunicación entre el frontend y backend.
- **API REST**:
  - Define rutas para obtener y actualizar los datos de los sensores.
  - **GET**: Recupera los datos de los sensores y los envía al frontend.
  - **POST**: Envía comandos desde el frontend al backend para controlar los actuadores.
- **Autenticación y seguridad**:
  - Uso de **JWT (JSON Web Tokens)** para autenticar solicitudes y proteger los datos.

Capa de presentación

Visualización de Datos en Tiempo Real en la App Web

- La visualización de datos en tiempo real es crucial en un sistema IoT, ya que permite al usuario monitorear el estado de los dispositivos en todo momento.

- **Gráficos y métricas en tiempo real**:
  - Usar **Grafana** embebido en la app web para mostrar gráficos de datos en tiempo real.
  - Actualización automática de los gráficos sin recargar la página.
- **Conexión con la base de datos**:
  - Grafana se conecta a **InfluxDB** para mostrar datos de series temporales.
  - Los datos se actualizan constantemente a medida que los sensores envían nueva información.

Capa de presentación

Despliegue de la Aplicación Web

- Una vez desarrollada la aplicación web, el siguiente paso es desplegarla para que los usuarios puedan acceder a ella desde cualquier lugar.
- **Plataformas de despliegue**:
  - **Heroku**: Plataforma PaaS que permite desplegar aplicaciones web rápidamente.
  - **Netlify**: Excelente para alojar sitios estáticos o SPAs.
  - **AWS**: Ideal para proyectos que requieren escalabilidad y control total sobre la infraestructura.
- **Proceso de despliegue**:
  - Conectar el repositorio (GitHub) a la plataforma de despliegue.
  - Automatizar el despliegue continuo para facilitar las actualizaciones.

Capa de presentación

Actualización y Mantenimiento de la Aplicación Web

- Una vez desplegada, es importante mantener la aplicación actualizada y asegurarse de que funcione correctamente a medida que se añaden nuevas características o se corrigen errores.
- **Actualizaciones continuas**:
  - Utilización de **CI/CD (Integración Continua y Despliegue Continuo)** para automatizar las actualizaciones.
  - Cada vez que se realiza un cambio en el código, se prueba y despliega automáticamente.
- **Monitoreo y logs**:
  - Implementar herramientas de monitoreo como **Grafana** o **Datadog** para asegurarse de que la app esté funcionando correctamente.
  - Revisar logs de errores y rendimiento para detectar problemas rápidamente.

Capa de presentación

Capa de presentación

Cuándo y Cómo Migrar al Desarrollo Móvil

- Decidir cuándo migrar de una aplicación web a una aplicación móvil depende de varios factores que influyen en la necesidad de acceso a funcionalidades nativas y la experiencia del usuario.

- **Cuándo migrar**:
  - Cuando se necesita acceso directo a **hardware nativo** del dispositivo (cámara, GPS, sensores).
  - Si la experiencia del usuario necesita mejoras en términos de **rendimiento** y usabilidad en dispositivos móviles.
- **Cómo migrar**:
  - Utilizar **frameworks multiplataforma** como **React Native** o **Ionic** para aprovechar el código existente.
  - Desarrollar componentes nativos solo cuando sea necesario.

Capa de presentación

Proceso de Migración desde Web a Móvil

- El proceso de migración implica reutilizar el código web existente y adaptarlo para que funcione de manera óptima en plataformas móviles.
- **Reutilización del código web**:
  - Si ya se ha desarrollado una **PWA** o una aplicación web con **React** u otro framework, gran parte del código puede ser reutilizado.
  - Los componentes UI pueden ser adaptados con **React Native** o **Ionic** para generar aplicaciones nativas.
- **Adaptación a plataformas móviles**:
  - Asegurarse de que el diseño sea **responsive** y adaptado a pantallas móviles.
  - Incluir navegación fluida y rápida, optimizada para el uso táctil.

Capa de presentación

Reutilización del Código Web en Apps Móviles

- Uno de los grandes beneficios de los frameworks multiplataforma es la posibilidad de **reutilizar el código** que ya se ha escrito para la web, minimizando el esfuerzo de desarrollo.
- **Ventajas de la reutilización**:
  - Menor tiempo y costo de desarrollo.
  - Mantener un solo código base para ambas plataformas.
  - Facilita las actualizaciones simultáneas para web y móvil.
- **Herramientas clave**:
  - **React Native**: Basado en React, permite reutilizar componentes y lógica de la aplicación.
  - **Ionic**: Utiliza tecnologías web para construir aplicaciones móviles con acceso a funcionalidades nativas.

Capa de presentación

Capacitor y Cordova para Acceso a Funcionalidades Nativas

- Para aprovechar al máximo el hardware de los dispositivos móviles, **Capacitor** y **Cordova** permiten a las aplicaciones web acceder a funcionalidades nativas.

- **Capacitor (Ionic Framework)**:
  - Proporciona acceso a funcionalidades como la cámara, GPS, y almacenamiento local.
  - Compatible con aplicaciones web y móviles.
- **Cordova**:
  - Plataforma de desarrollo que permite ejecutar aplicaciones web en contenedores nativos y acceder a APIs nativas.
- **Cuándo utilizarlos**:
  - Cuando se necesita interactuar con hardware específico del dispositivo, como sensores o cámaras.

Capa de presentación

Acceso a Funcionalidades Nativas en Apps Móviles

- El acceso a **funcionalidades nativas** es una de las principales razones para migrar una aplicación web a móvil.
- **Funcionalidades nativas disponibles**:
  - **Cámara**: Captura de imágenes o escaneo de códigos QR.
  - **GPS**: Seguimiento y localización del usuario o dispositivo.
  - **Notificaciones push**: Envío de alertas en tiempo real, incluso si la app está cerrada.
  - **Acelerómetro y giroscopio**: Utilizados en aplicaciones que requieren datos del movimiento del dispositivo.
- **Ventajas**:
  - Permite una experiencia de usuario más rica e interactiva.
  - Mejora la integración con el dispositivo móvil y su sistema operativo.

Capa de presentación

Integración con Características como GPS, Cámaras y Notificaciones Push

- La integración con características nativas como el **GPS**, la **cámara**, y las **notificaciones push** es crucial para ofrecer una experiencia más inmersiva y funcional en dispositivos móviles.

- **GPS**:
  - Utilizado para rastreo y localización, ideal para aplicaciones IoT que monitorean ubicaciones o movimientos de dispositivos.
- **Cámara**:
  - Permite la captura de imágenes, escaneo de códigos QR o realidad aumentada.
- **Notificaciones push**:
  - Envío de alertas en tiempo real cuando se producen eventos importantes en el sistema IoT.
- **Mejora en la experiencia del usuario**:
  - Estas características aumentan la capacidad de respuesta e interacción con el sistema, haciendo que las aplicaciones sean más útiles y personalizadas.

Capa de presentación

Capa de presentación

Seguridad en Aplicaciones Web IoT

- La **seguridad** es un aspecto crítico en las aplicaciones IoT, ya que las conexiones entre los dispositivos y la capa de presentación pueden ser vulnerables si no se protegen adecuadamente.
- **Importancia de la seguridad**:
  - Protege los datos transmitidos entre el sistema IoT y el usuario.
  - Evita accesos no autorizados y posibles manipulaciones de dispositivos IoT.
- **Riesgos comunes**:
  - Ataques de **intercepción de datos**.
  - Acceso no autorizado a dispositivos.
  - Manipulación malintencionada de sensores o actuadores.

Capa de presentación

Autenticación y Autorización en Aplicaciones IoT

- La autenticación y autorización son procesos clave para asegurar que solo usuarios autorizados puedan acceder y controlar los dispositivos IoT.

- **Autenticación**:
  - Proceso para verificar la identidad del usuario.
  - Implementación de **OAuth 2.0** y **JWT (JSON Web Tokens)** para autenticación segura en aplicaciones web IoT.

- **Autorización**:
  - Define qué acciones puede realizar un usuario autenticado.
  - Gestión de permisos y roles para limitar el acceso a ciertas funcionalidades.

Capa de presentación

Uso de OAuth 2.0 y JWT para Autenticación

- Para asegurar las comunicaciones y el acceso, se utilizan estándares de seguridad como **OAuth 2.0** y **JWT**.
- **OAuth 2.0**:
  - Protocolo que permite a aplicaciones de terceros obtener acceso limitado a recursos protegidos.
  - Utilizado en aplicaciones IoT para gestionar el acceso seguro a dispositivos y datos.
- **JWT (JSON Web Tokens)**:
  - Método para crear tokens de autenticación que son enviados con cada solicitud para validar al usuario.
  - Cada token contiene la información de autenticación y los permisos del usuario.

Capa de presentación

Protegiendo la Capa de Presentación contra Ataques Comunes

- Existen varios ataques comunes a los que las aplicaciones web son vulnerables. La protección contra estos ataques es fundamental para asegurar la capa de presentación.
- **XSS (Cross-Site Scripting)**:
  - Inyección de código malicioso en la aplicación web.
  - Mitigación: Validación y escape de las entradas del usuario.
- **CSRF (Cross-Site Request Forgery)**:
  - Ataques en los que se realiza una acción en nombre de un usuario sin su consentimiento.
  - Mitigación: Uso de tokens CSRF y verificación de solicitudes.
- **SQL Injection**:
  - Inyección de consultas SQL maliciosas en las entradas de la aplicación.
  - Mitigación: Uso de consultas preparadas y validación de entradas.

Capa de presentación

Implementación de HTTPS para Proteger la Comunicación

- El uso de **HTTPS** es esencial para proteger las comunicaciones entre la capa de presentación y los dispositivos IoT.

- **Por qué HTTPS es importante**:
  - Protege la integridad y confidencialidad de los datos transmitidos.
  - Cifra todas las comunicaciones, evitando que los datos puedan ser interceptados por atacantes.
- **Implementación de HTTPS**:
  - Uso de certificados SSL/TLS para asegurar que las comunicaciones sean seguras.
  - Todos los intercambios de datos entre el servidor y el cliente deben estar encriptados.

Capa de presentación

Control de Acceso en Dashboards y Apps Web

- El **control de acceso** es fundamental en las aplicaciones IoT para asegurar que los usuarios tengan acceso solo a las funcionalidades y datos que les corresponden.

- **Gestión de roles y permisos**:
  - Definir roles (admin, usuario, operador) con distintos niveles de acceso.
  - Solo los administradores tienen acceso a configuraciones críticas, mientras que los operadores solo pueden monitorear y actuar sobre dispositivos.
- **Implementación**:
  - Uso de sistemas de autenticación y autorización para definir qué funciones y datos están disponibles para cada usuario.
  - Separación clara de privilegios para evitar que usuarios sin autorización puedan modificar dispositivos o datos sensibles.

Capa de presentación

Seguridad en APIs para IoT

- Las **APIs** son una parte fundamental de la comunicación entre la capa de presentación y el backend de IoT. Asegurar estas APIs es crucial para evitar brechas de seguridad.

- **Protección de APIs REST**:
  - Implementación de **tokens de acceso** (OAuth 2.0 y JWT) para validar cada solicitud.
  - Limitar las rutas y métodos disponibles para ciertos usuarios o roles.
- **Rate limiting y monitoreo**:
  - Implementar **rate limiting** para evitar abusos y sobrecargas.
  - Monitoreo continuo de las solicitudes a las APIs para detectar comportamientos inusuales o intentos de ataque.

Capa de presentación

Capa de presentación

Tendencias en la Capa de Presentación en IoT

- La capa de presentación en sistemas IoT está evolucionando rápidamente para mejorar la experiencia del usuario y aprovechar las nuevas tecnologías emergentes.

- **Tendencias clave**:
  - **Interfaces más inmersivas**: Uso de **Realidad Aumentada (AR)** y **Realidad Virtual (VR)** para visualizar datos.
  - **Integración de IA**: Utilización de inteligencia artificial para mejorar la visualización y predicción de datos.
  - **Automatización**: Los dashboards pueden incluir más capacidades de automatización y toma de decisiones autónomas.

Capa de presentación

Integración con Inteligencia Artificial y Machine Learning

- La integración de **Inteligencia Artificial (IA)** y **Machine Learning (ML)** en la capa de presentación está permitiendo una visualización y análisis de datos más avanzada.

- **IA en la Capa de Presentación**:
  - Permite la **predicción de eventos** y el análisis avanzado de los datos obtenidos de sensores IoT.
  - Mejora la toma de decisiones automatizada en tiempo real.
- **Machine Learning**:
  - Modelos de ML se integran en el backend para analizar grandes volúmenes de datos y luego se presentan en forma accesible para el usuario.
  - Ejemplo: Predicción de fallas en dispositivos o sistemas IoT.

Capa de presentación

Uso de Realidad Aumentada (AR) y Realidad Virtual (VR)

- El uso de **Realidad Aumentada (AR)** y **Realidad Virtual (VR)** está abriendo nuevas posibilidades para visualizar y controlar los sistemas IoT de manera más inmersiva.
- **Realidad Aumentada (AR)**:
  - Permite superponer información del sistema IoT en el entorno físico usando dispositivos móviles o gafas AR.
  - Ejemplo: Ver información sobre un dispositivo IoT físico directamente en el campo.
- **Realidad Virtual (VR)**:
  - Crea simulaciones virtuales para visualizar el estado del sistema en un entorno controlado.
  - Permite la interacción con dispositivos IoT en un entorno completamente virtual.

Capa de presentación

Aplicaciones Emergentes de la Capa de Presentación

- Las aplicaciones de la capa de presentación están expandiéndose más allá de los dashboards tradicionales, con nuevos usos y tecnologías en constante desarrollo.

- **Aplicaciones emergentes**:
  - **Interacción por voz**: Integración con asistentes virtuales (como Alexa, Google Assistant) para controlar dispositivos IoT con comandos de voz.
  - **Interfaces autónomas**: Capacidad de los dashboards para ejecutar acciones automáticamente en función de reglas predefinidas.
- **Hacia dónde se dirige la capa de presentación**:
  - Mayor personalización para cada usuario.
  - **Automatización avanzada**: Los sistemas actuarán basándose en datos sin intervención del usuario.
  - **Adaptación en tiempo real**: Las interfaces cambiarán dinámicamente según el contexto y los datos obtenidos.

Capa de presentación

Personalización y Automatización en el Futuro de la Capa de Presentación

- La personalización y la automatización serán elementos clave para los futuros desarrollos de la capa de presentación en IoT.
- **Personalización**:
  - Interfaces adaptativas que ajustan la información y el control según las necesidades específicas del usuario.
  - Ejemplo: Un agricultor verá métricas detalladas del estado de sus cultivos en función de los sensores en su terreno.
- **Automatización**:
  - Las capas de presentación incluirán más herramientas de **automatización** y **control autónomo**, donde las acciones son tomadas automáticamente según reglas predefinidas.
  - Ejemplo: Un sistema que ajusta la temperatura de una instalación basado en datos en tiempo real sin intervención humana.

Capa de presentación

Futuras Herramientas y Tecnologías en la Capa de Presentación

- Las nuevas herramientas y tecnologías continúan expandiendo el alcance de la capa de presentación en sistemas IoT.

- **Dashboards con análisis predictivo**:
  - Los dashboards incorporarán **algoritmos de predicción** para anticipar fallas o problemas en el sistema.
- **Nuevas tecnologías para mejorar la experiencia del usuario**:
  - **Interfaces por voz**, **control gestual**, y **realidad mixta** serán comunes en la capa de presentación del futuro.
  - Más interacción sin necesidad de interfaces gráficas tradicionales.
- **Nuevas integraciones**:
  - **Blockchain** para garantizar la seguridad y trazabilidad de los datos.
  - **Edge Computing** integrado para reducir la latencia y permitir decisiones más rápidas.

Capa de presentación

¡Muchas gracias!