Universidad Tecnológica de Acapulco
Organismo Público Descentralizado del Gobierno del Estado

T.S.U en TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN ÁREA SISTEMAS INFORMÁTICOS

DESARROLLO MÓVIL MULTIPLATAFORMA

Elaborado por:

JOSSELINE SOTELO HERNÁNDEZ

202204017 Sotelo Hernández Josseline

Acapulco, Guerrero, enero 2024

ÍNDICE

I. OBJETIVOS DE LA INVESTIGACIÓN

OBJETIVO GENERAL

El alumno desarrollará aplicaciones móviles multiplataforma mediante el uso de frameworks para el control de dispositivos de hardware abierto y gestión de información en bases de datos.

OBJETIVOS ESPECÍFICOS

1. El alumno programará interfaces de usuario para aplicaciones móviles.
2. El alumno desarrollará aplicaciones multiplataforma para dispositivos móviles.
3. El alumno desarrollará aplicaciones multiplataforma para dispositivos de hardware abierto y administración de información de información en Bases de Datos.

UNIDAD I Marco de referencia para desarrollo de aplicaciones multiplataforma

OBJETIVO

El alumno programará interfaces de usuario para aplicaciones móviles.

ACTIVIDADES

Actividad 1-1 “Características de las APPs nativas y multiplataforma.”

Actividad 1-2 “ Desarrollo con HTML5 y CSS3.”

Actividad 1-3 “Funcionalidad con JavaScript.”

RESULTADO DE APRENDIZAJE

Desarrolla interfaces de usuario para aplicaciones móviles utilizando:

- HTML5.
- CSS3.
- Java Script.

Actividad 1-1 "Características de las APPs nativas y multiplataforma."

APPs nativas:

El desarrollo nativo se refiere a utilizar el lenguaje oficial del dispositivo así como las herramientas proporcionadas por el fabricante. Desarrollar una aplicación nativa te costará más que una multiplataforma o una híbrida (la mayoría de las veces) y también tomará más tiempo. Tratar con dos aplicaciones nativas para tu proyecto en lugar de una multiplataforma es una tarea más compleja.

- Corre mejor y más rápido.
- Integrar nuevas funciones es más rápido y fácil.
- Todo lo relacionado con el hardware, como seguimiento de ubicación geográfica, cámara o micrófono, es mucho más sencillo de implementar.
- Android y iOS crean y mejoran constantemente herramientas para resolver problemas de desarrollo.
- Las reglas de diseño y experiencia de usuario son diferentes para cada plataforma.

En resumen, las aplicaciones nativas son tu mejor oportunidad para crear experiencias de usuario específicas de la plataforma. Para ello,debes saber que usuario está en el centro de cada decisión que tome con respecto a tu negocio. Tu usuario no sabrá la diferencia entre un código de aplicación nativo y uno multiplataforma, pero ciertamente verán y sentirán la diferencia.

APPs multiplataforma:

Como su nombre sugiere, multiplataforma implica el desarrollo de aplicaciones en un conjunto de varias plataformas. Esto significa que el código se crea una sola vez y se ejecuta en varios dispositivos o plataformas, en este caso iOS y Android. Este método de desarrollo es más adecuado para aquellas aplicaciones que generalmente necesitan una interacción simple y no requieren funcionalidades específicas y/o complejas.

- Solo se desarrolla una vez.
- Menor costo.
- Facilidad de agregar o modificar funcionalidades, ya que sólo se tiene que modificar en un solo lugar.

Las aplicaciones multiplataforma son perfectas para desarrollar prototipos y/o aplicaciones simples sin funcionalidad específica del hardware del dispositivo, lo que las hace magníficas para startups. En caso de que la aplicación crezca, es recomendable migrar a nativo para así poder agregar mejores funcionalidades y más específicas, así como una mejor experiencia del usuario.

Actividad 1-2 " Desarrollo con HTML5 y CSS3."

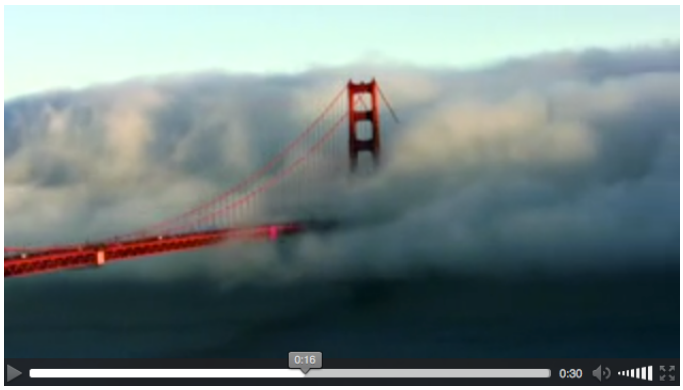
Sintaxis del atributo controls HTML5 para vídeo:

```
<video width="600" height="400" controls>
    <source src="video.mp4" type="video/mp4">
</video>
```

Tal y como se aprecia en el código superior solo hay que añadir el atributo controls para activar los controles:

- Con <video> indicamos que vamos que se va a embeber un vídeo, podemos asignar un alto y ancho al reproductor.
- Con <source> especificamos la ruta del vídeo a reproducir.
- Con **controls** activamos los controles del player.

Os muestro en la siguiente imagen un **player de vídeo HTML5** con los **controles activados** y visibles:



Sintaxis del atributo controls HTML5 para audio:

Os pongo la sintaxis en el caso del player de audio extraída de su web oficial:

```XHTML
<audio controls>
    <source src="014-audio-helicoptero.mp3" type="audio/mp3">
    Tu navegador no soporta HTML5 audio.
</audio>
```

Tal y como se aprecia en el código superior con tan solo 2 etiquetas podemos crear el player de audio:

- Con **<audio>** indicamos que vamos que se va a embeber un audio.
- Con **<source>** especificamos la ruta del audio a reproducir.
- Con **controls** activamos los controles del player.

Sintaxis del atributo controls HTML5 para entrada de datos:

<input>: Utilizado para crear campos de entrada, como texto, contraseña, número, etc.

Ejemplo de campo de texto:

```html
<input type="text" placeholder="Nombre de usuario">
```

Ejemplo de campo de contraseña:

```html
<input type="password" placeholder="Contraseña">
```

Sintaxis del atributo controls HTML5 para formularios:

<form>: Utilizado para crear formularios que permiten la entrada y envío de datos.

Ejemplo de **<form>** con campos de texto y botón de envío:

```html
<form action="/procesar" method="post">
  <label for="nombre">Nombre:</label>
  <input type="text" id="nombre" name="nombre">

  <label for="email">Correo electrónico:</label>
  <input type="email" id="email" name="email">

  <input type="submit" value="Enviar">
</form>
```

Atributos de CSS3 DIVs:

display: Este atributo controla el tipo de caja que se utiliza para un elemento. Los valores comunes son **block** (bloque), **inline** (en línea) y **flex** (flexible).

width y height: Definen el ancho y la altura de un elemento.

Atributos de CSS3 Colores:

color: Define el color del texto.

background-color: Define el color de fondo.

Atributos de CSS3 Posicionamiento:

position: Especifica el método de posicionamiento. Los valores comunes son **static** (por defecto), **relative**, **absolute**, y **fixed**.

top, right, bottom, left: Se utilizan junto con **position** para ajustar la posición del elemento.

Atributos de CSS3 Márgenes:

margin: Define el espacio alrededor de un elemento. Puede ser un valor único para todos los lados o especificar márgenes individuales.

Atributos de CSS3 Fuentes:

font-family: Especifica la familia de fuentes para un elemento.

font-size: Define el tamaño de la fuente.

font-weight: Establece el grosor de la fuente.

Actividad 1-3 “Funcionalidad con JavaScript.”

Sentencias de control:

Las aplicaciones JavaScript se componen de sentencias con una sintaxis propia. Una sentencia puede estar formada por múltiples líneas. Puede haber varias sentencias en una sola línea si separamos cada una de las sentencias por un punto y coma. No es una palabra clave, sino un grupo de palabras clave.

Block :Un bloque de sentencias se utiliza para agrupar cero o mas sentencias. El bloque se delimita por un par de llaves.

break :Finaliza la sentencia actual loop, switch, o label y transfiere el control del programa a la siguiente sentencia de la sentencia finalizada.

continue: Finaliza la ejecucion de las sentencias dentro de la iteracion actual del actual bucle, y continua la ejecucion del bucle con la siguiente iteracion.

Empty: Una sentencia vacía se utiliza para proveer una "no sentencia", aunque la sintaxis de JavaScript esperaba una.

```javascript
if (condición) {
  // código si la condición es verdadera
} else {
  // código si la condición es falsa
}
```

I

f...else: Ejecuta una sentencia si una condición especificada es true. Si la condición es false, otra sentencia puede ser ejecutada.

switch: Evalua una expresión, igualando el valor de la expresión a una clausula case y ejecuta las sentencias asociadas con dicho case.

throw: Lanza una excepción definida por el usuario.

```javascript
switch (expresion) {
  case valor1:
    // código si expresion es igual a valor1
    break;
  case valor2:
    // código si expresion es igual a valor2
    break;
  default:
    // código si expresion no coincide con ningún caso
}
```

try...catch: Marca un bloque de sentencias para ser probadas (try) y especifica una respuesta, en caso de que se lance una excepción.

Funciones:

function:Declara una función con los parámetros especificados.

```javascript
function nombreFuncion(parametro1, parametro2) {
  // código de la función
  return resultado; // opcional
}

// Ejemplo de llamada a la función
var resultado = nombreFuncion(valor1, valor2);
```

function*: Los generadores de funciones permiten escribir iteradores con mas facilidad.

```javascript
var nombreFuncion = function(parametro1, parametro2) {
  // código de la función
  return resultado; // opcional
};

// Ejemplo de llamada a la función
var resultado = nombreFuncion(valor1, valor2);
```

async function:Declara una función asíncrona con los parámetros especificados.

return:Especifica el valor a ser retornado por una función.

class:Declara una clase.

Clases y Objetos: Declaraciones de clase

```javascript
class NombreClase {
  constructor(parametro1, parametro2) {
    this.propiedad1 = parametro1;
    this.propiedad2 = parametro2;
  }

  metodo1() {
    // código del método
  }

  metodo2() {
    // código del método
  }
}

// Crear instancia de la clase
var objeto = new NombreClase(valor1, valor2);
```

Herencia de clase

```javascript
class ClasePadre {
  // código de la clase padre
}

class ClaseHija extends ClasePadre {
  // código de la clase hija
}
```

Objetos literales:

```javascript
var objeto = {
  propiedad1: valor1,
  propiedad2: valor2,
  metodo1: function() {
    // código del método
  },
  metodo2: function() {
    // código del método
  }
};

// Acceder a propiedades y métodos
var valor = objeto.propiedad1;
objeto.metodo1();
```

UNIDAD II Desarrollo para ambientes multiplataforma

OBJETIVO

El alumno desarrollará aplicaciones multiplataforma para dispositivos móviles.

ACTIVIDADES

Actividad 2-1 “Ambiente de desarrollo multiplataforma (framework).

Actividad 2-2 “ Programación de la funcionalidad de la aplicación móvil.”

Actividad 2-3 “Generación de estructura de archivos y carpetas.”

Actividad 2-4 “Generación de APP's multiplataforma.”

RESULTADO DE APRENDIZAJE

Desarrolla aplicaciones multiplataforma para dispositivos móviles utilizando:

- Plantillas desarrolladas con HTML5, CSS3 y Java Script.
- Framework de desarrollo.
- Pruebas en emuladores y dispositivos móviles.

Actividad 2-1 “Ambiente de desarrollo multiplataforma (framework).

Acceso al Dispositivo:

APIs Estándar: El ambiente de desarrollo multiplataforma proporciona APIs (Interfaz de Programación de Aplicaciones) estándar que permiten a los desarrolladores acceder a características del dispositivo de manera uniforme, independientemente de la plataforma.

Abstracción del software: Proporciona capas de abstracción que permiten a los desarrolladores interactuar con el hardware del dispositivo sin preocuparse por las diferencias entre plataformas.

Esmuladores y Simuladores:Ofrece herramientas de emulación o simulación para probar y depurar aplicaciones en diferentes plataformas sin la necesidad de tener acceso físico a todos los dispositivos.

Desempeño:

Optimización de Rendimiento: El ambiente de desarrollo busca optimizar el rendimiento de las aplicaciones para garantizar que sean eficientes y ejecuten de manera fluida en diversas plataformas.

Compilaciôn Cruzada:Permite la compilación cruzada, donde el código fuente escrito en un sistema operativo se puede compilar para ejecutarse en otros sistemas operativos o arquitecturas de hardware.

Distribución:

Empaquetado Uniforme:mFacilita el empaquetado uniforme de aplicaciones para diferentes plataformas, simplificando el proceso de distribución.

Tiendas de Aplicaciones: Se integra con tiendas de aplicaciones multiplataforma, permitiendo a los desarrolladores distribuir sus aplicaciones de manera centralizada.

Acrualizaciones Centrales:Facilita la implementación de actualizaciones y parches de manera centralizada, independientemente de la plataforma.

Conectividad:

Compatibilidad con Redes: Garantiza la compatibilidad con diferentes tipos de redes y tecnologías de comunicación para garantizar que las aplicaciones puedan conectarse eficientemente.

Gestión de Datos en la Nube: Facilita la integración con servicios en la nube para gestionar datos de manera eficiente, independientemente de la plataforma.

Soporte para Tecnologías Web: Permite el uso de tecnologías web como API REST, WebSocket, etc., para facilitar la comunicación entre la aplicación y los servidores, independientemente de la plataforma.

Actividad 2-2 " Programación de la funcionalidad de la aplicación móvil."

1. Inicio de Sesión: Usuario inicia sesión con credenciales válidas.

2. Registro:Nuevo usuario crea una cuenta en la aplicación.

3. Navegación: Usuario navega por diferentes secciones o páginas de la aplicación.

4. Interacción con Contenido: Usuario visualiza, crea, edita o elimina contenido dentro de la aplicación (puede incluir publicaciones, fotos, mensajes, etc.).

5. Notificaciones: Usuario recibe notificaciones push sobre eventos relevantes o actualizaciones.

6. Configuración de Perfil: Usuario ajusta la configuración de su perfil, como cambiar la foto, actualizar información personal, etc.

7. Compra o Transacción: Usuario realiza compras o transacciones dentro de la aplicación.

8. Búsqueda: Usuario realiza búsquedas dentro de la aplicación para encontrar información específica.

9. Compartir Contenido:

Usuario comparte contenido de la aplicación a través de redes sociales u otros canales.

10. Actividades en Tiempo Real: Usuario participa en actividades en tiempo real, como chats, videollamadas, o seguimiento de ubicación en vivo.

11. Gestión de Amigos o Contactos: Usuario agrega, elimina o gestiona sus conexiones dentro de la aplicación.

12. Feedback y Calificaciones: Usuario proporciona feedback sobre la aplicación o califica contenido.

13. Actualizaciones y Parches: La aplicación se actualiza automáticamente o notifica al usuario sobre nuevas versiones disponibles.

14. Eventos del Sistema: Captura de eventos del sistema como cambios en la conectividad, cambios de orientación del dispositivo, etc.

15. Cierre de Sesión: Usuario cierra sesión en la aplicación.

16. Errores y Excepciones: Manejo de errores y excepciones para garantizar la estabilidad de la aplicación.

Elementos de autenticación de la aplicación móvil

1. Inicio de Sesión (Login): Los usuarios ingresan sus credenciales para autenticarse en la aplicación.

Los usuarios pueden optar por iniciar sesión utilizando sus cuentas de redes sociales (por ejemplo, iniciar sesión con Google, Facebook, etc.).

2. Registro de Cuenta: Los usuarios crean una cuenta proporcionando información básica y estableciendo credenciales de acceso.

3. Recuperación de Contraseña: Mecanismos que permiten a los usuarios restablecer sus contraseñas en caso de olvido.

4. Autenticación de Dos Factores (2FA): Además de las credenciales, se utiliza una segunda capa de autenticación, como un código enviado por SMS, correo electrónico, o generado por una aplicación de autenticación.

5. Biometría: Utilización de características biométricas del usuario, como huellas dactilares, reconocimiento facial o escaneo de iris, para autenticarse.

6. Sesiones y Tokens: Se emite un token de acceso después de una autenticación exitosa, el cual se utiliza para autorizar las solicitudes subsiguientes.

Definición de un tiempo de duración para las sesiones de usuario antes de que deban volver a autenticarse.

7. Captchas y Desafíos Antifraude: Verificación de que el usuario es humano mediante la resolución de un desafío visual o de texto.

Implementación de medidas adicionales para detectar y prevenir actividades fraudulentas.

8. Protección contra Ataques: Mecanismos que bloquean temporalmente una cuenta después de varios intentos fallidos de inicio de sesión.Uso de captchas o desafíos después de varios intentos de inicio de sesión fallidos.

9. Registro de Actividad: Registro de eventos de inicio de sesión y cambios de contraseña para monitorear la actividad de la cuenta.

10. Políticas de Contraseña: Establecimiento de requisitos para contraseñas fuertes, como longitud mínima, caracteres especiales, etc.

11. Autenticación en Tiempo Real: Verificación continua de la identidad del usuario durante la sesión.

Actividad 2-3 “Generación de estructura de archivos y carpetas.”

La organización de carpetas y archivos en un proyecto que sigue un framework puede variar según el framework específico utilizado. Sin embargo, muchos frameworks modernos para desarrollo web o móvil siguen una estructura similar para facilitar la comprensión y el mantenimiento del código. Aquí proporcionaré una estructura común para un proyecto web, basado en un patrón MVC (Modelo-Vista-Controlador), que es utilizado por muchos frameworks web. Ten en cuenta que esto es solo un ejemplo general y puede variar según el framework y las necesidades específicas del proyecto.

/config: Contiene archivos de configuración del proyecto, como configuraciones de base de datos, variables de entorno, etc.

/controllers: Aquí se almacenan los controladores del MVC. Cada controlador maneja las interacciones entre el modelo y la vista.

/models: Contiene los modelos de datos que representan la lógica de negocio y la interacción con la base de datos.

/views: Almacena las plantillas de vistas, que pueden ser archivos HTML, EJS, Pug, etc.

/partials: Subcarpeta que contiene fragmentos de vistas reutilizables (encabezados, pies de página, etc.).

/public: Carpeta para archivos estáticos accesibles públicamente (CSS, imágenes, scripts del lado del cliente, etc.).

/css: Archivos de estilo.

/js: Archivos JavaScript del lado del cliente.

/routes: Contiene archivos de enrutamiento que dirigen las solicitudes HTTP a los controladores adecuados.

/tests: Carpeta para pruebas unitarias y de integración.

/utils: Archivos de utilidades y funciones auxiliares.

app.js: Archivo principal que inicializa la aplicación, configura middleware y define rutas.

package.json: Archivo que contiene la información del proyecto, dependencias y scripts de ejecución.

Actividad 2-4 “Generación de APP's multiplataforma.”

Despliegue de Interfaces Web:

Preparación del Entorno de Producción:

- Configurar un servidor web (por ejemplo, Apache, Nginx) y un servidor de aplicaciones si es necesario.
- Configurar una base de datos en el entorno de producción si la aplicación utiliza una.

Empaquetado de la Aplicación:

- Compilar y empaquetar la aplicación, asegurándose de incluir todos los archivos estáticos (HTML, CSS, JavaScript) y las dependencias.

Despliegue del Código Fuente:

- Subir el código fuente y los archivos empaquetados al servidor de producción a través de métodos como FTP, SCP o integración continua.

Configuración de Variables de Entorno:

- Configurar variables de entorno específicas del entorno de producción (por ejemplo, configuración de base de datos, claves API).

Gestión de Dependencias:

- Instalar las dependencias del proyecto en el entorno de producción utilizando un gestor de paquetes como npm o yarn para Node.js, por ejemplo.

Configuración del Servidor Web:

- Configurar el servidor web para redirigir las solicitudes al servidor de aplicaciones y manejar rutas.

Configuración de SSL (opcional):

- Configurar un certificado SSL para habilitar el protocolo HTTPS y garantizar la seguridad de la comunicación.

Reinicio del Servidor:

- Reiniciar el servidor web y, si es necesario, el servidor de aplicaciones para aplicar los cambios.

Despliegue de Interfaces Móviles:

Preparación del Entorno de Desarrollo:

- Configurar entornos de desarrollo para Android Studio (para aplicaciones Android) o Xcode (para aplicaciones iOS).

Empaquetado de la Aplicación:

- Compilar y empaquetar la aplicación para la plataforma específica (APK para Android, IPA para iOS).

Firma del Aplicativo (para iOS):

- Firmar la aplicación iOS con un certificado de desarrollador para permitir su instalación en dispositivos.

Despliegue en Tiendas de Aplicaciones:

- Subir la aplicación a las tiendas de aplicaciones relevantes (Google Play Store, Apple App Store).
- Completar los procesos de revisión y aprobación de la tienda de aplicaciones.

Despliegue de Interfaces de Escritorio:

Empaquetado de la Aplicación:

- Empaquetar la aplicación para la plataforma de escritorio específica (por ejemplo, un instalador MSI para Windows, un paquete DMG para macOS).

Firma y Certificación (si es necesario):

- Firmar y certificar la aplicación, especialmente si se trata de un software distribuido de manera comercial o pública.

Despliegue en Plataformas de Distribución:

- Publicar la aplicación en plataformas de distribución de software, como Microsoft Store, Mac App Store o distribución directa desde el sitio web.

Actualizaciones:

- Implementar un mecanismo de actualización automática si es necesario, para facilitar la entrega de nuevas versiones de la aplicación.

UNIDAD III Control de dispositivos de hardware abierto.

OBJETIVO

El alumno desarrollará aplicaciones multiplataforma para dispositivos de hardware abierto y administración de información de información en Bases de Datos.

ACTIVIDADES

Actividad 3-1 " Configuración de módulos de conectividad en el dispositivo móvil."

Actividad 3-2 " Desarrollo demétodos de envío y recepción de datos del hardware abierto."

Actividad 3-3 "Desarrollo de métodos de envío y recepción de datos a la Base de Datos."

RESULTADO DE APRENDIZAJE

Desarrolla aplicaciones para el control de hardware abierto y gestión de información en bases de datos que incluya:

- Monitoreo de dispositivos de hardware abierto.
- Control de dispositivos de hardware abierto.
- Envío de información a la base de datos.
- Consulta de información de la base de datos.

Actividad 3-1 " Configuración de módulos de conectividad en el dispositivo móvil."

Procesador (CPU): El procesador es el cerebro del dispositivo móvil y realiza todas las operaciones de cálculo y procesamiento de datos.

Memoria RAM: La memoria RAM (Random Access Memory) se utiliza para almacenar temporalmente datos y programas en ejecución, permitiendo un acceso más rápido que el almacenamiento a largo plazo.

Almacenamiento interno: Este módulo se refiere a la capacidad de almacenamiento a largo plazo del dispositivo, donde se guardan aplicaciones, archivos multimedia y otros datos.

Pantalla táctil: La pantalla táctil permite la interacción del usuario mediante toques, gestos y deslizamientos. Puede ser LCD, OLED, AMOLED, entre otros tipos.

Cámara: Los dispositivos móviles suelen contar con cámaras traseras y frontales para la captura de fotos y videos, así como para funciones como reconocimiento facial.

Sensores de movimiento: Incluyen acelerómetros, giroscopios y magnetómetros que permiten al dispositivo detectar el movimiento y la orientación.

Conectividad: Módulos como Wi-Fi, Bluetooth, GPS y NFC permiten la conexión a redes, dispositivos externos y servicios de ubicación.

Batería: Este módulo proporciona la fuente de energía para alimentar el dispositivo.

Altavoces y micrófono**:** Estos módulos permiten la reproducción de sonido y la captura de audio para llamadas y otras funciones multimedia.

Sistema operativo: Aunque no es un componente físico, el sistema operativo es esencial y proporciona la interfaz y la funcionalidad del dispositivo.

Conectividad de red: Los dispositivos móviles cuentan con módulos para conectividad celular, como 4G o 5G, que les permiten acceder a la red móvil para llamadas, mensajes y datos.

Puertos y conectores: Incluyen puertos de carga, puertos para auriculares, y otros conectores que permiten la conexión con accesorios y la carga de la batería.

Medios de configuración de los módulos disponibles en los dispositivos móviles

Configuración del sistema operativo: Los dispositivos móviles tienen menús de configuración integrados en su sistema operativo. Los usuarios pueden acceder a estos menús para ajustar configuraciones generales del dispositivo, como la fecha y la hora, el idioma, las cuentas de usuario, las actualizaciones de software y más.

Configuración de red: Para módulos como Wi-Fi, Bluetooth y redes celulares, los usuarios pueden acceder a la configuración de red para conectarse a redes Wi-Fi, emparejar dispositivos Bluetooth y gestionar configuraciones de datos móviles.

Configuración de la pantalla: Los usuarios pueden ajustar la configuración de la pantalla, como el brillo, la resolución, el tamaño del texto y la orientación en la configuración del dispositivo.

Configuración de sonido: A través de la configuración de sonido, los usuarios pueden ajustar el volumen del timbre, las notificaciones, la música y otros sonidos del dispositivo. También pueden gestionar la configuración del audio para auriculares y otros dispositivos de audio.

Configuración de la cámara: Algunos dispositivos permiten configurar opciones de la cámara, como la resolución de las fotos, la calidad del video, los modos de captura y otras configuraciones relacionadas con la fotografía y la grabación de video.

Configuración de aplicaciones: Muchas aplicaciones instaladas en dispositivos móviles tienen sus propias configuraciones. Los usuarios pueden personalizar preferencias específicas de cada aplicación a través de sus menús de configuración individuales.

Configuración de seguridad y privacidad: Los usuarios pueden ajustar configuraciones relacionadas con la seguridad y la privacidad, como el bloqueo de pantalla, las contraseñas, la autenticación biométrica, el control de permisos de aplicaciones y otras medidas de seguridad.

Configuración de energía y batería: Los dispositivos móviles suelen tener configuraciones relacionadas con el ahorro de energía y la gestión de la batería. Los usuarios pueden ajustar configuraciones para optimizar el rendimiento y la duración de la batería.

Configuración de accesibilidad: Se incluyen configuraciones que permiten a los usuarios personalizar la experiencia según sus necesidades específicas, como el tamaño del texto, la accesibilidad táctil, el reconocimiento de voz y otras opciones de accesibilidad.

Actividad 3-2 " Desarrollo demétodos de envío y recepción de datos del hardware abierto."

Sintaxis:

Lenguaje de Programación: Puede variar según la plataforma de hardware, pero a menudo se utiliza C, C++, Python o algún lenguaje específico para hardware, como Verilog o VHDL.

Programación a Nivel de Registro: En hardware abierto, a menudo se accede a los registros de hardware directamente para configurar y controlar el funcionamiento del hardware.

APIs y Librerías: Dependiendo del hardware, puede haber APIs y librerías específicas para interactuar con los periféricos y componentes de hardware.

Proceso:

Entender la Arquitectura del Hardware: Antes de desarrollar métodos de envío y recepción de datos, es crucial comprender la arquitectura del hardware abierto en el que se está trabajando. Esto incluye conocer la disposición de los registros, la interconexión de los componentes y los protocolos de comunicación.

Configuración de Periféricos: Muchos dispositivos en hardware abierto se comunican a través de periféricos como UART, SPI, I2C, etc. Se configuran estos periféricos según las necesidades del proyecto.

Programación de Controladores: Se desarrollan controladores de hardware para gestionar la comunicación y el intercambio de datos. Esto a menudo implica escribir código a nivel de registro para configurar y controlar los periféricos.

Implementación de Protocolos de Comunicación: Si es necesario, se implementan protocolos de comunicación específicos, como TCP/IP, para facilitar la transmisión y recepción de datos a través de interfaces de red.

Manejo de Interrupciones y Eventos: En entornos de hardware, es común manejar interrupciones y eventos. Se desarrolla código para gestionar estas interrupciones y responder a eventos específicos, como la llegada de nuevos datos.

Pruebas y Depuración: Dado que trabajar con hardware abierto implica configuración a nivel de registro y manipulación directa de hardware, las pruebas y la depuración son críticas para garantizar un funcionamiento correcto.

Optimización del Rendimiento:Se pueden realizar optimizaciones para mejorar el rendimiento del sistema, como la optimización de código y la gestión eficiente de la memoria.

Documentación: Es esencial documentar el código y el diseño del hardware para facilitar la colaboración y el mantenimiento futuro.

Actividad 3-3 “Desarrollo de métodos de envío y recepción de datos a la Base de Datos.”

Lenguajes de Consulta:

SQL (Structured Query Language): Es el lenguaje estándar para interactuar con bases de datos relacionales. SQL se utiliza para realizar operaciones como la inserción, actualización, selección y eliminación de datos.

Proceso:

1. Diseño de la Base de Datos: Antes de comenzar el desarrollo, se realiza un diseño de la base de datos que incluye la definición de tablas, relaciones entre tablas, campos y tipos de datos.
2. Creación de la Base de Datos: Se crea la base de datos utilizando un DBMS como MySQL, PostgreSQL, SQL Server, etc. Esto implica ejecutar scripts SQL para definir la estructura de la base de datos.
3. Conexión a la Base de Datos: En el desarrollo de aplicaciones, se establece una conexión con la base de datos. Esto generalmente se hace a través de bibliotecas o controladores específicos del lenguaje de programación que estás utilizando.
4. Desarrollo de Métodos de Envío (Inserción de Datos): Se escriben consultas SQL o se utilizan funciones específicas del lenguaje de programación para insertar datos en la base de datos. Por ejemplo, en SQL:

```sql
INSERT INTO tabla (columna1, columna2) VALUES (valor1, valor2);
```

5. Desarrollo de Métodos de Recepción (Consulta de Datos): Se escriben consultas SQL o se utilizan funciones específicas para recuperar datos de la base de datos. Por ejemplo, en SQL:

```sql
SELECT columna1, columna2 FROM tabla WHERE condicion;
```

6. Actualización y Eliminación de Datos: Se desarrollan métodos para actualizar y eliminar datos según sea necesario. Por ejemplo, en SQL:

```sql
UPDATE tabla SET columna1 = nuevo_valor WHERE condicion;
DELETE FROM tabla WHERE condicion;
```

7. Manejo de Transacciones: Se implementa el manejo de transacciones para garantizar la consistencia y la integridad de los datos. Las transacciones agrupan operaciones y aseguran que se realicen todas o ninguna.
8. Seguridad y Prevención de Inyección SQL: Se implementan prácticas de seguridad para prevenir ataques de inyección SQL, como el uso de consultas parametrizadas o declaraciones preparadas.
9. Pruebas y Optimización del Rendimiento: Se realizan pruebas exhaustivas para garantizar que los métodos de envío y recepción de datos funcionen correctamente. También se optimiza el rendimiento de las consultas y operaciones.
10. Documentación: Es fundamental documentar el esquema de la base de datos, así como los métodos de envío y recepción de datos para facilitar el mantenimiento y la colaboración.

II. BIBLIOGRAFÍAS

https://www.um.es/docencia/barzana/DAWEB/2017-18/daweb-tema-10-introduccion-css.html

https://developer.mozilla.org/es/docs/Learn/Forms/Basic_native_form_controls

https://www.anerbarrena.com/atributo-controls-html5-video-5295/

https://developer.mozilla.org/es/docs/Web/CSS/Using_CSS_custom_properties

https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Statements

https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Grammar_and_types

https://developer.mozilla.org/es/docs/Learn/JavaScript/Objects/Basics

https://creatuaplicacion.com/cuales-son-las-funciones-de-las-aplicaciones-moviles/

https://www.uv.es/fragar/html/html1302.html

https://aggregate.digital/es/products/solutions/mobile-device-management.html

https://docs.oracle.com/cd/E24842_01/html/820-2981/ipov-29.html