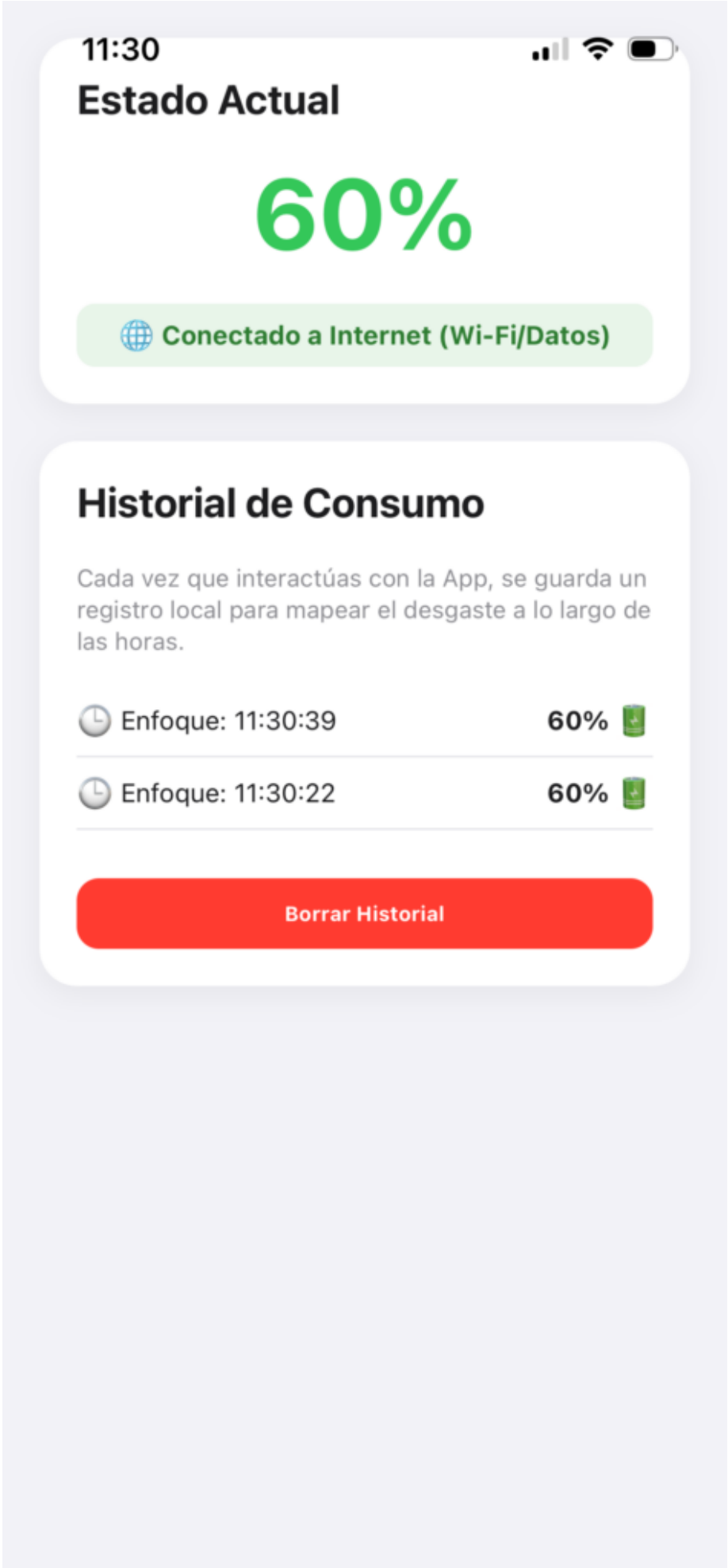
Guía definitiva: cómo crear un monitor de consumo de batería e Internet en iPhone (desde cero)


11:30
Estado Actual
60%
Conectado a Internet (Wi-Fi/Datos)
Historial de Consumo
Cada vez que interactúas con la App, se guarda un registro local para mapear el desgaste a lo largo de las horas.
Enfoque: 11:30:39
60%
Enfoque: 11:30:22
60%
Borrar Historial

Cuando desarrollamos aplicaciones móviles con tecnologías web (HTML, JavaScript y CSS) usando **Ionic Capacitor**, uno de los mayores desafíos es entender los límites del sistema operativo. Apple es sumamente estricto con la privacidad: **no permite rastrear redes secundarias (como Bluetooth o ubicaciones de antenas de telefonía) ni ejecutar procesos infinitos en segundo plano sin permisos explícitos.**

En este tutorial aprenderás a construir desde el absoluto cero un **Monitor de Consumo de Batería**. El enfoque técnico correcto para este escenario es registrar el estado del dispositivo cada vez que el usuario interactúa con la aplicación, almacenando de forma persistente un historial local en el teléfono (localStorage) para analizar el desgaste. Además, añadiremos detección de conectividad a internet sin requerir molestas alertas de autorización.

Requisitos previos

Para poder compilar este proyecto, necesitas tener instalado en tu Mac:

1. **Node.js:** Puedes descargarlo desde su sitio web oficial.
2. **Xcode:** La herramienta oficial de Apple (disponible gratis en la Mac App Store).

Paso 1: Crear el directorio y la estructura base

Abre la **Terminal** de tu Mac y ejecuta los siguientes comandos uno a uno para levantar los cimientos del proyecto:

```
# 1. Crear una carpeta limpia para nuestra aplicación
mkdir monitorbateria

# 2. Entrar a la carpeta recién creada
```

```
cd monitorbateria

# 3. Inicializar el entorno de Node.js (genera el archivo package.json)
npm init -y

# 4. Instalar las dependencias principales del núcleo de Capacitor
npm install @capacitor/core @capacitor/cli
```

Paso 2: Inicializar la configuración de la App

Ejecuta el siguiente comando en la terminal para definir la identidad de tu aplicación dentro del ecosistema de Apple:

```
npx cap init "MonitorBat" "com.tuusuario.monitor" --web-dir=www
```

- "MonitorBat": Nombre que aparecerá en la pantalla de inicio del iPhone.
- "com.tuusuario.monitor": Identificador único de la aplicación.
- --web-dir=www: Indica que todo nuestro código fuente web se alojará en la carpeta www.

Paso 3: Instalar el Plugin de Dispositivo

Para saltarnos el navegador interno de iOS y acceder a los datos reales de carga de la batería, instalamos el plugin oficial de hardware:

```
npm install @capacitor/device
```

Paso 4: Crear el código de la interfaz Web

Creamos la carpeta contenedora donde irá el diseño y la lógica de nuestra aplicación:

```
mkdir www
```

Ahora, crea un archivo llamado **index.html** dentro de esa carpeta www utilizando tu editor de código favorito (como VS Code) y pega el siguiente código completo:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Monitor de Batería Inteligente</title>
    <style>
        body {
            background-color: #f2f2f7;
            color: #1c1c1e;
            margin: 0;
            padding: 20px;
            display: flex;
            flex-direction: column;
            align-items: center;
        }
        .card {
            background: white;
            border-radius: 20px;
            padding: 20px;
            box-shadow: 0 4px 15px rgba(0,0,0,0.05);
            width: 100%;
            max-width: 400px;
            margin-bottom: 20px;
            box-sizing: border-box;
        }
        h1, h2 {
```

```
            margin-top: 0;
            color: #1c1c1e;
            font-size: 22px;
        }
        .battery-large {
            font-size: 54px;
            font-weight: bold;
            text-align: center;
            margin: 15px 0;
            color: #34c759;
        }
        .badge {
            display: block;
            padding: 8px 12px;
            border-radius: 10px;
            font-size: 14px;
            font-weight: bold;
            text-align: center;
            margin-top: 10px;
        }
        .badge-ok { background-color: #e8f5e9; color: #2e7d32;
}
          .badge-warn { background-color: #fff3e0; color:
#ef6c00; }
        .log-list {
            list-style: none;
            padding: 0;
            max-height: 220px;
            overflow-y: auto;
            margin-top: 15px;
        }
        .log-item {
            display: flex;
            justify-content: space-between;
            padding: 10px 0;
            border-bottom: 1px solid #e5e5ea;
            font-size: 14px;
        }
        .btn-clear {
            background-color: #ff3b30;
            color: white;
```

```
            border: none;
            padding: 12px 15px;
            border-radius: 12px;
            width: 100%;
            font-weight: bold;
            cursor: pointer;
            margin-top: 10px;
        }
        .btn-clear:active {
            background-color: #d32f2f;
        }
    </style>
</head>
<body>

    <div class="card">
        <h1>Estado Actual</h1>
        <div id="bat-level" class="battery-large">--%</div>
        <div>
             <span id="net-status" class="badge">Detectando
red...</span>
        </div>
    </div>

    <div class="card">
        <h2>Historial de Consumo</h2>
         <p style="font-size: 13px; color: #8e8e93; margin:
0;">Cada vez que interactúas con la App, se guarda un registro
local para mapear el desgaste a lo largo de las horas.</p>
        <ul id="log-container" class="log-list">
            </ul>
                            <button   class="btn-clear"
onclick="limpiarHistorial()">Borrar Historial</button>
    </div>

    <script>
        async function analizarDispositivo() {
            try {
                // Comprobamos la inyección global del puente
nativo de Capacitor
                            if  (window.Capacitor  &&
```

```
window.Capacitor.Plugins && window.Capacitor.Plugins.Device) {
                                        const Device =
window.Capacitor.Plugins.Device;
                                    const info = await
Device.getBatteryInfo();
                    // Convertir el float de iOS (0.0 a 1.0) a
formato de porcentaje (0% a 100%)
                    const level = Math.round(info.batteryLevel
* 100);
                         document.getElementById('bat-
level').textContent = level + '%';
                     // Cambiar el color visual del número
según el rango
                               const percentageElement =
document.getElementById('bat-level');
                                      if (level <= 20)
percentageElement.style.color = '#ff3b30';
                                  else if (level <= 50)
percentageElement.style.color = '#ff9500';
                      else percentageElement.style.color =
'#34c759';

                     // Detección de conectividad básica (Sin
requerir permisos de localización)
                    const isOnline = navigator.onLine;
                                       const netBadge =
document.getElementById('net-status');
                    if (isOnline) {
                        netBadge.textContent = "□ Conectado a
Internet (Wi-Fi/Datos)";
                       netBadge.className = "badge badge-ok";
                    } else {
                           netBadge.textContent = "⚠□ Sin
conexión a Internet";
                         netBadge.className = "badge badge-
warn";
                    }

                     // Guardar automáticamente este punto de
control en el almacenamiento del dispositivo
                               guardarEnHistorial(level,
```

```
info.isCharging);
                }
            } catch (error) {
                 console.error("Error al leer el hardware:",
error);
            }
        }

        function guardarEnHistorial(nivel, cargando) {
             // Recuperamos el historial existente o creamos
uno nuevo vacío
                                        let   historial   =
JSON.parse(localStorage.getItem('historial_bateria')) || [];
            const ahora = new Date();
                                  const  horaFormateada  =
ahora.toLocaleTimeString([], {hour: '2-digit', minute:'2-
digit', second:'2-digit'});
            const nuevoRegistro = {
                hora: horaFormateada,
                porcentaje: nivel,
                estado: cargando ? "⚡" : "🔋"
            };

              // Evitamos guardar duplicados exactos si se
actualiza en el mismo segundo
              if (historial.length === 0 || historial[0].hora
!== horaFormateada) {
                 historial.unshift(nuevoRegistro); // Insertar
al inicio de la lista
                   localStorage.setItem('historial_bateria',
JSON.stringify(historial));
                renderizarHistorial();
            }
        }

        function renderizarHistorial() {
                                     const   historial   =
JSON.parse(localStorage.getItem('historial_bateria')) || [];
             const container = document.getElementById('log-
container');
            container.innerHTML = "";
```

```
            historial.forEach(item => {
                const li = document.createElement('li');
                li.className = "log-item";
                       li.innerHTML = <span> Enfoque:
${item.hora}</span>       <strong>${item.porcentaje}%
${item.estado}</strong>;
                container.appendChild(li);
            });
        }

        function limpiarHistorial() {
            localStorage.removeItem('historial_bateria');
            renderizarHistorial();
        }

        // Iniciar la rutina cuando el DOM esté listo
        window.addEventListener('DOMContentLoaded', () => {
            setTimeout(() => {
                analizarDispositivo();
                renderizarHistorial();
            }, 500);
        });
    </script>
</body>
</html>
```

Paso 5: Agregar la plataforma iOS y Sincronizar

Vuelve a la terminal y ejecuta estos comandos para inyectar todo el entorno web dentro del contenedor de Apple:

```
# 1. Añadir el módulo de Capacitor para iOS
npm install @capacitor/ios

# 2. Agregar la plataforma nativa al directorio
npx cap add ios

# 3. Copiar el archivo index.html y configurar los plugins nativos en iOS
```

npx cap sync ios

Paso 6: Evitar errores de conexión física (Truco del Modo Avión)

Para que Xcode no genere errores de emparejamiento con tu dispositivo de prueba:

1. Desconecta el cable USB de tu iPhone.
2. Abre el centro de control del iPhone y **activa el Modo Avión**. Comprueba que el Wi-Fi y el Bluetooth se apaguen completamente.
3. Vuelve a conectar el cable USB a tu Mac.

Abre Xcode desde la terminal:

npx cap open ios

Paso 7: Compilar desde Xcode

1. **Firmar la aplicación (Gratis):** Haz clic en la carpeta azul raíz llamada **App** en la barra lateral izquierda. Ve a la pestaña **Signing & Capabilities** en el centro. En el desplegable de **Team**, selecciona tu **Apple ID**.
2. **Elegir destino:** En la barra superior de herramientas (junto al botón de Play), cambia el simulador genérico y **selecciona tu iPhone físico**.
3. **Compilar limpiamente:** Ve al menú superior, haz clic en **Product -> Clean Build Folder** y posteriormente presiona el botón **Play** (el triángulo equilátero de la esquina superior izquierda).

Paso 8: Confiar en el Desarrollador (Solo la primera vez)

Una vez que la aplicación se instale en tu pantalla de inicio, iOS bloqueará su apertura por motivos de seguridad con un cartel de desarrollador no confiable.

1. Abre **Ajustes** en tu iPhone y ve a **General**.
2. Entra en **Administración de dispositivos y VPN**.
3. Haz clic sobre tu correo electrónico asociado al Apple ID y selecciona «**Confiar**».

¡Listo! Ya puedes ejecutar tu aplicación. Cada vez que abras el monitor, registrará el consumo exacto y comprobará el estado de la red sin necesidad de pedir un solo permiso invasivo, cumpliendo estrictamente con las políticas de privacidad de Apple.