

**Reconstrucción visual de
rutas: cómo dibujar el
recorrido GPS de un iPhone en
un mapa en tiempo real y en
segundo plano**

15:15

RASTREADOR INTELIGENTE

Filtro de 1 Minuto + Cartografía Forense Activa

 INICIAR SEGUIMIENTO DE RUTA



EXPORTAR



IMPORTAR



LIMPIAR

Cargando registros y mapas...



HORA

COORDENADAS

VEL.

DIRECCIÓN / LUGAR
DE INTERÉS

Ningún punto registrado todavía

Un gran desafío al desarrollar aplicaciones híbridas para iPhone es el acceso continuo al hardware en segundo plano. Si intentas registrar el GPS de un usuario usando un simple temporizador de JavaScript (`setInterval`), notarás que iOS congela y suspende el proceso a los pocos segundos de bloquear la pantalla para proteger la batería.

Además, si intentas exportar la información recolectada mediante enlaces de descarga tradicionales (`data:text/json`), la capa de seguridad de Apple WebKit (Sandbox) bloqueará la descarga.

En este megapost técnico aprenderás a construir, desde cero, un **Rastreador de Rutas Pro con Cartografía Activa** capaz de:

- Conectarse de forma nativa al chip GPS del iPhone sin *root* ni *jailbreak*.
- Evitar la suspensión de iOS mediante los *Background Modes* de Apple.
- Filtrar las ubicaciones para procesar datos estrictamente **cada 1 minuto**.
- Pintar un **mapa interactivo en tiempo real** con la ruta trazada (sin APIs de pago).
- Traducir las coordenadas a direcciones reales (calles, comercios) mediante Geocodificación Inversa Gratuita.
- **Exportar e Importar ficheros JSON firmados digitalmente con SHA-256** utilizando el almacenamiento interno y la app *Archivos* de iOS.

▯▯ Parte 1: Inicialización del Proyecto en la Terminal

Abre la **Terminal** de tu Mac y ejecuta los siguientes comandos de forma ordenada para levantar la estructura limpia del proyecto, instalar el núcleo de Capacitor y añadir los plugins nativos necesarios para el GPS de fondo, los mapas, el sistema

de archivos y el menú de compartir de iOS:

1. Crear el directorio principal y acceder a él

```
mkdir tracker-ios-pro
```

```
cd tracker-ios-pro
```

2. Inicializar el entorno Node.js

```
npm init -y
```

3. Instalar dependencias del núcleo de Capacitor

```
npm install @capacitor/core @capacitor/cli
```

4. Inicializar la configuración base de la App

```
npx cap init "RutaPro" "com.jesusninoc.prueba" --web-dir=www
```

5. Instalar los componentes nativos oficiales y comunitarios requeridos

```
npm install @capacitor/ios
```

```
npm install @capacitor-community/background-geolocation
```

```
npm install @capacitor/filesystem @capacitor/share
```

⚙️ Parte 2: Configuración Crítica del Servidor de Capacitor

Para resolver de raíz el error habitual *módulo capacitor no detectado* o fallos en las peticiones del Sandbox (xpc_user_sessions), es obligatorio forzar a WebKit a levantar un esquema de red local seguro.

Abre el archivo **capacitor.config.json** en la raíz de tu proyecto y reemplaza su contenido por este:

```
{  
  "appId": "com.jesusninoc.prueba",  
  "appName": "RutaPro",  
  "webDir": "www",  
  "server": {  
    "iosScheme": "capacitor",  
    "hostname": "localhost"  
  }  
}
```

}

❑ Parte 3: El Código de la Interfaz Web e Integración de Mapas (index.html)

Crea una carpeta llamada `www/` en la raíz de tu proyecto:

```
mkdir www
```

Dentro de `www/`, crea tu archivo **`index.html`** e inyecta este código unificado. Este script cuenta con el motor cartográfico **Leaflet**, persistencia de memoria mediante `localStorage` y un validador criptográfico SHA-256 para blindar la **cadena de custodia** de la prueba:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Rastreador Pro con Mapas e Integración
Forense</title>
    <script src="capacitor.js"></script>

                                <link          rel="stylesheet"
href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css" />
                                <script
src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>
>

    <style>
        body {
            background-color: #0f172a;
            color: #ffffff;
            margin: 0;
            padding: 15px;
            box-sizing: border-box;
        }
    </style>
</head>
<body>
```

```
h1 {
    font-size: 22px;
    color: #38bdf8;
    text-transform: uppercase;
    text-align: center;
    margin-bottom: 5px;
}
.subtitle {
    font-size: 12px;
    color: #94a3b8;
    text-align: center;
    margin-bottom: 20px;
}
.btn-main {
    width: 100%;
    max-width: 400px;
    border: none;
    border-radius: 12px;
    padding: 18px;
    font-size: 15px;
    font-weight: bold;
    color: white;
    background-color: #0284c7;
    cursor: pointer;
    display: block;
    margin: 0 auto 15px auto;
    box-shadow: 0 4px 12px rgba(2, 132, 199, 0.3);
}
.actions-panel {
    display: flex;
    justify-content: center;
    gap: 10px;
    max-width: 400px;
    margin: 0 auto 20px auto;
}
.btn-tool {
    flex: 1;
    border: none;
    border-radius: 8px;
    padding: 10px;
    font-size: 12px;
```

```

        font-weight: bold;
        color: white;
        cursor: pointer;
    }
    .btn-export { background-color: #16a34a; }
    .btn-import { background-color: #eab308; color:
#0f172a; }
    .btn-clear { background-color: #dc2626; }

    .status {
        text-align: center;
        font-size: 13px;
        color: #e2e8f0;
        margin-bottom: 15px;
    }
    /* Contenedor del Mapa */
    #map {
        width: 100%;
        max-width: 600px;
        height: 300px;
        margin: 0 auto 20px auto;
        border-radius: 12px;
        border: 2px solid #334155;
        z-index: 1;
    }

    .table-container {
        width: 100%;
        max-width: 600px;
        margin: 0 auto;
        overflow-x: auto;
        background-color: #1e293b;
        border: 1px solid #334155;
        border-radius: 12px;
    }
    table {
        width: 100%;
        border-collapse: collapse;
        font-size: 12px;
        text-align: left;
    }

```

```

    th, td {
        padding: 10px 12px;
        border-bottom: 1px solid #334155;
    }
    th {
        background-color: #0f172a;
        color: #38bdf8;
        text-transform: uppercase;
        font-size: 11px;
    }
    .text-geo {
        color: #10b981;
        font-weight: bold;
    }
    #file-input { display: none; }
</style>
</head>
<body>

    <h1>☐ Rastreador Inteligente</h1>
    <div class="subtitle">Filtro de 1 Minuto + Cartografía
Forense Activa</div>

        <button            class="btn-main"
onclick="iniciarRastradorCompleto()">
    ☐ INICIAR SEGUIMIENTO DE RUTA
    </button>
    <div class="actions-panel">
        <button class="btn-tool btn-export"
onclick="exportarHistorialNativo()">☐ EXPORTAR</button>
        <button class="btn-tool btn-import"
onclick="document.getElementById('file-input').click()">☐
IMPORTAR</button>
        <button class="btn-tool btn-clear"
onclick="borrarHistorial()">☐☐ LIMPIAR</button>
    </div>

    <input type="file" id="file-input" accept=".json"
onchange="importarHistorialNativo(event)">

    <div id="status-log" class="status">Cargando registros y

```

```
mapas...</div>
```

```
<div id="map"></div>
```

```
<div class="table-container">
```

```
<table>
```

```
<thead>
```

```
<tr>
```

```
<th>Hora</th>
```

```
<th>Coordenadas</th>
```

```
<th>Vel.</th>
```

```
<th>Dirección / Lugar de Interés</th>
```

```
</tr>
```

```
</thead>
```

```
<tbody id="ruta-log-body"></tbody>
```

```
</table>
```

```
</div>
```

```
<script>
```

```
let historialPuntos = [];
```

```
let ultimoTiempoRegistrado = 0;
```

```
let mapaInstancia = null;
```

```
let capaLineaRuta = null;
```

```
let marcadorPosicionActual = null;
```

```
window.addEventListener('DOMContentLoaded', () => {  
    inicializarMapaVisual();
```

```
const guardado =  
localStorage.getItem('historial_gps_pro');
```

```
if (guardado) {
```

```
    historialPuntos = JSON.parse(guardado);
```

```
    renderizarTabla();
```

```
    actualizarMapaYLineas();
```

```
    document.getElementById('status-  
log').textContent = Memoria: ${historialPuntos.length} puntos  
cargados.;
```

```
    } else {
```

```
        mostrarTablaVacía();
```

```
    }
```

```
});
```

```

function inicializarMapaVisual() {
    mapaInstancia = L.map('map').setView([40.4167,
-3.7037], 6);
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.pn
g', {
    attribution: '© OpenStreetMap contributors'
}).addTo(mapaInstancia);

    capaLineaRuta = L.polyline([], { color: '#38bdf8',
weight: 4 }).addTo(mapaInstancia);
}

function obtenerHoraActual() {
    return new Date().toString().split(' ')[0];
}

function mostrarTablaVacia() {
    document.getElementById('ruta-log-body').innerHTML
=
        <tr><td colspan="4" style="text-align: center;
color: #64748b;">Ningún punto registrado todavía</td></tr>;
}

function renderizarTabla() {
    const tablaBody = document.getElementById('ruta-
log-body');
    if (historialPuntos.length === 0) {
        mostrarTablaVacia();
        return;
    }
    tablaBody.innerHTML = "";
    historialPuntos.forEach(punto => {
        const fila = document.createElement('tr');
        fila.innerHTML =
            <td>${punto.hora}</td>
            <td style="">${punto.lat.toFixed(4)},
${punto.lon.toFixed(4)}</td>
            <td>${punto.velocidad} km/h</td>
            <td class="text-
geo">${punto.direccion}</td>
        ;
    });
}

```

```

        tablaBody.appendChild(fila);
    });
}

function actualizarMapaYLineas() {
    if (historialPuntos.length === 0) {
        capaLineaRuta.setLatLngs([]);
        if (marcadorPosicionActual)
            mapaInstancia.removeLayer(marcadorPosicionActual);
        return;
    }

    const coordenadasRuta =
[...historialPuntos].reverse().map(punto => [punto.lat,
punto.lon]);
    capaLineaRuta.setLatLngs(coordenadasRuta);

    const ultimaPosicion = [historialPuntos[0].lat,
historialPuntos[0].lon];

    if (marcadorPosicionActual) {
        marcadorPosicionActual.setLatLng(ultimaPosicion);
    } else {
        marcadorPosicionActual =
L.marker(ultimaPosicion).addTo(mapaInstancia);
    }

    mapaInstancia.setView(ultimaPosicion, 16);
}

async function obtenerDireccionTexto(lat, lon) {
    try {
        const respuesta = await
fetch(https://nominatim.openstreetmap.org/reverse?format=json&
lat=${lat}&lon=${lon}&zoom=18&addressdetails=1, {
            headers: { 'User-Agent':
'AppRutaProGps/1.0' }
        });
        const datos = await respuesta.json();
        if (datos && datos.display_name) {
            const calle = datos.address.road || "";

```

```

        const numero = datos.address.house_number
        || "";
        const ciudad = datos.address.city ||
datos.address.town || datos.address.village || "";
        const puntoInteres = datos.name || "";
        if (puntoInteres && puntoInteres !==
calle) return `${puntoInteres} (${calle}, ${ciudad});
        return `${calle} ${numero},
${ciudad}.trim() || datos.display_name;
    }
    return "Dirección no identificada";
} catch (err) {
    return "Error al traducir dirección";
}
}

async function iniciarRastradorCompleto() {
    const log = document.getElementById('status-log');
    if (typeof window.Capacitor === 'undefined' ||
!window.Capacitor.Plugins.BackgroundGeolocation) {
        log.textContent = "❏ Inicializando puente
nativo. Reintente.";
        return;
    }

    const BackgroundGeolocation =
window.Capacitor.Plugins.BackgroundGeolocation;
    try {
        log.textContent = "Sincronizando servicio de
fondo...";
        await BackgroundGeolocation.addWatcher(
            {
                backgroundTitle: "Seguimiento
Inteligente Activo",
                backgroundMessage: "Procesando
ubicación cada minuto en segundo plano.",
                requestPermissions: true,
                stale: false,
                distanceFilter: 0
            },
            async function callback(location, error) {
                if (error) return;

```

```

        if (location) {
            const ahoraMilisegundos =
Date.now();
            if (ahoraMilisegundos -
ultimoTiempoRegistrado >= 60000 || ultimoTiempoRegistrado ===
0) {
                log.textContent = "⏱ Filtro 1
Min superado. Traduciendo...";
                ultimoTiempoRegistrado =
ahoraMilisegundos;
                const hora =
obtenerHoraActual();
                const velocidadKmh =
location.speed && location.speed > 0 ?
Math.round(location.speed * 3.6) : 0;
                const direccionTexto = await
obtenerDireccionTexto(location.latitude, location.longitude);
                const nuevoPunto = {
                    hora: hora,
                    lat: location.latitude,
                    lon: location.longitude,
                    velocidad: velocidadKmh,
                    direccion: direccionTexto
                };
                historialPuntos.unshift(nuevoPunto);
                localStorage.setItem('historial_gps_pro',
JSON.stringify(historialPuntos));
                renderizarTabla();
                actualizarMapaYLineas();
                log.textContent = "⏱ Historial
y Mapa actualizados.";
            }
        }
    }
    );
    log.textContent = "⏱ Rastreador activado (1
Minuto).";
} catch (err) {
    log.textContent = "⏱ Error en Background
Tracker.";
}

```

```

    }

    // ALGORITMO CRIPTOGRÁFICO DE CADENA DE CUSTODIA
    (SHA-256)
    async function calcularHashForense(texto) {
        const encoder = new TextEncoder();
        const data = encoder.encode(texto);
        const hashBuffer = await
crypto.subtle.digest('SHA-256', data);
        const hashArray = Array.from(new
Uint8Array(hashBuffer));
        return hashArray.map(b =>
b.toString(16).padStart(2, '0')).join('');
    }

    async function exportarHistorialNativo() {
        const log = document.getElementById('status-log');
        if (historialPuntos.length === 0) {
            alert("No hay información para exportar.");
            return;
        }

        if (typeof window.Capacitor === 'undefined' ||
!window.Capacitor.Plugins.FileSystem) {
            const dataStr =
"data:text/json;charset=utf-8," +
encodeURIComponent(JSON.stringify(historialPuntos, null, 2));
            const downloadAnchor =
document.createElement('a');
            downloadAnchor.setAttribute("href", dataStr);
            downloadAnchor.setAttribute("download",
"historial.json");
            downloadAnchor.click();
            return;
        }

        const FileSystem =
window.Capacitor.Plugins.FileSystem;
        const Share = window.Capacitor.Plugins.Share;
        const nombreArchivo =
historial_gps_${Date.now()}.json;

```

```

        const cadenaJson = JSON.stringify(historialPuntos,
null, 2);

        try {
            log.textContent = "Generando archivo y firma
hash...";

            const hashEvidencia = await
calcularHashForense(cadenaJson);

            const result = await Filesystem.writeFile({
                path: nombreArchivo,
                data: cadenaJson,
                directory: 'CACHE',
                encoding: 'utf8'
            });

            log.textContent = "Abriendo menú de compartir
de iOS...";

            await Share.share({
                title: 'Exportación Forense GPS',
                text: " EVIDENCIA DIGITAL INTEGRAL\nHash
SHA-256 de origen verificado:\n${hashEvidencia},
                url: result.uri,
                dialogTitle: 'Guardar o enviar evidencia'
            });

            log.textContent = " Fichero exportado y
firmado.";
        } catch (err) {
            log.textContent = " Error al exportar archivo
en iOS.";
        }
    }

    function importarHistorialNativo(event) {
        const log = document.getElementById('status-log');
        const archivo = event.target.files[0];
        if (!archivo) return;

        log.textContent = "Leyendo archivo

```

```

seleccionado...";
    const lector = new FileReader();
    lector.onload = function(e) {
        try {
            const datosImportados =
JSON.parse(e.target.result);
            if (Array.isArray(datosImportados)) {
                historialPuntos = datosImportados;
                localStorage.setItem('historial_gps_pro',
                JSON.stringify(historialPuntos));
                renderizarTabla();
                actualizarMapaYLineas();
                log.textContent = " Importado con
éxito: ${datosImportados.length} puntos mapeados.;
            } else {
                alert("Estructura JSON inválida.");
            }
        } catch (err) {
            alert("Error al parsear el JSON.");
        }
    };
    lector.readAsText(archivo);
    event.target.value = "";
}

function borrarHistorial() {
    if (confirm("¿Borrar permanentemente todo el
historial y limpiar el mapa?")) {
        historialPuntos = [];
        localStorage.removeItem('historial_gps_pro');
        ultimoTiempoRegistrado = 0;
        renderizarTabla();
        actualizarMapaYLineas();
        document.getElementById('status-
log').textContent = " Historial y mapas eliminados.";
    }
}
</script>
</body>
</html>

```

☐☐ Parte 4: Compilación y Vinculación Estructural

Regresa a la **Terminal** para enlazar tu código e inyectar las dependencias nativas de Swift en Xcode. Usaremos comandos de actualización profunda para enlazar de forma transparente los nuevos módulos de archivos e impedir el error de paquetes perdidos Missing package product 'CapApp-SPM':

```
# 1. Agregar el contenedor de la plataforma iOS nativa  
npx cap add ios
```

```
# 2. Sincronizar el HTML nuevo y las dependencias de los  
plugins de Node a Xcode  
npx cap sync ios
```

```
# 3. Forzar el refresco completo de los paquetes de  
dependencias locales de Apple  
npx cap update ios
```

☐ Parte 5: Configuración de Capacidades y Privacidad en Xcode

Lanza Xcode desde la terminal ejecutando: `npx cap open ios`.

En la barra lateral izquierda de Xcode, selecciona el proyecto raíz azul (**App**).

Entra a la pestaña intermedia **Signing & Capabilities**, pulsa en el botón superior + **Capability**, escribe **Background Modes** en el buscador y presiona doble clic.

En el listado que se desplegará abajo, marca de forma obligatoria estas dos casillas para evitar la congelación de hilos del sistema operativo:

- **Location updates**
- **Background fetch**

Dirígete a la pestaña continua llamada **Info**. Añade tres filas nuevas haciendo clic en el botón (+) para declarar los strings de privacidad exigidos estrictamente en las revisiones de seguridad de Apple:

- **Privacy - Location When In Use Usage Description** -> *Necesitamos tu ubicación en tiempo real para trazar tu ruta cartográfica activa.*
- **Privacy - Location Always and When In Use Usage Description** -> *Necesitamos el GPS continuo en segundo plano para registrar las coordenadas y direcciones por las que pasas.*
- **Privacy - Location Always Usage Description** -> *Esta aplicación requiere comunicación permanente con el chip GPS de fondo para alimentar la telemetría del historial.*

□ Parte 6: Despliegue Limpio en el iPhone Físico

1. Activa el **Modo Avión** en tu iPhone y asegúrate de inhabilitar manualmente el Wi-Fi y el Bluetooth desde la aplicación de Ajustes del sistema (esto previene fallos aleatorios de comunicación de red local con el Sandbox en Xcode).
2. Conecta el iPhone a tu Mac usando el cable USB y selecciónalo en el menú desplegable de dispositivos activos de la esquina superior izquierda de Xcode.
3. Ejecuta una limpieza profunda de caché en Xcode: menú superior **Product** -> **Clean Build Folder** (o presiona el atajo de teclado Cmd + Shift + K).
4. Haz clic en el botón de **Play** (el triángulo de ejecución).
5. Al abrir la app en tu teléfono, pulsa el botón azul, acepta el permiso de localización seleccionando «Permitir al usar la app».

6. Finalmente, ve a **Ajustes > Privacidad y seguridad > Localización > RutaPro** en tu iPhone y cambia la marca de forma manual a la opción **Siempre**.

¡Listo! Ya tienes una solución profesional de seguimiento, con mapas integrados y blindada forensemente, lista para tus clases de máster.