Index Gps Acciona

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0,
user-scalable=no">
    <title>Monitor GPS | Acciona</title>
    <link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css" />
    <meta name="apple-mobile-web-app-capable" content="yes">
    <meta name="apple-mobile-web-app-status-bar-style" content="black-translucent">
    <meta name="theme-color" content="#f44336">
    <meta name="mobile-web-app-capable" content="yes">

    <script src="https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.18.5/xlsx.full.min.js"></script>
    <style>
        :root {
            --white: #ffffff;
            --red-light: #ffebee;
            --red-primary: #f44336;
            --red-dark: #d32f2f;
            --gray-light: #f5f5f5;
            --online-color: #27ae60;
            --offline-color: #95a5a6;
            --user-position-color: #3498db;
            --blue-primary: #2196f3;
            --blue-light: #e3f2fd;
            --blue-dark: #1976d2;
            --route1-color: #e74c3c;
            --route2-color: #9b59b6;
            --dashboard-bg: #f8f9fa;
            --dashboard-card: #ffffff;
            --dashboard-text: #333333;
            --dashboard-border: #e0e0e0;
            --shift1-color: #3498db;
            --shift2-color: #9b59b6;
            --shift3-color: #e67e22;
    --transparency: 0.35;

    --uv-low: #27ae60;
    --uv-moderate: #f39c12;
    --uv-high: #e67e22;
    --uv-very-high: #e74c3c;
    --uv-extreme: #9b59b6;

    --distance-close: #27ae60;
    --distance-medium: #f39c12;
    --distance-far: #e74c3c;
}
* {
    -webkit-tap-highlight-color: transparent;
    touch-action: manipulation;
}
body {
    margin: 0;
    padding: 0;
    font-family: 'Segoe UI', sans-serif;
    overflow: hidden;
    background-color: #2c3e50;
    touch-action: pan-x pan-y;
}
#map {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100vh;
    z-index: 1;
}

.weather-widget {
    position: fixed;
    top: 120px;
    left: 15px;
    z-index: 1000;
    background: rgba(255, 255, 255, var(--transparency));
    padding: 8px;
    border-radius: 50%;
    box-shadow: 0 4px 12px rgba(0,0,0,0.15);
    display: flex;
    align-items: center;
    justify-content: center;
    width: 30px;
    height: 30px;
    backdrop-filter: blur(5px);
    cursor: pointer;
    transition: all 0.3s ease;
}
.weather-widget.expanded {
    border-radius: 12px;
    width: auto;
    height: auto;
    background: rgba(255, 255, 255, 0.85);
    min-width: 220px;
    padding: 15px;
}
.weather-icon {
    width: 34px;
    height: 34px;
    display: flex;
    align-items: center;
    justify-content: center;
    font-size: 28px;
    transition: all 0.3s ease;
}
.weather-widget.expanded .weather-icon {
    margin-right: 12px;
}
.weather-info {
    display: none;
    flex-direction: column;
    opacity: 0;
    transition: opacity 0.3s ease;
```

```
}
.weather-widget.expanded .weather-info {
    display: flex;
    opacity: 1;
}
.weather-temp {
    font-size: 18px;
    font-weight: bold;
    color: #000;
    margin-bottom: 4px;
}
.weather-desc {
    font-size: 12px;
    color: #000;
    margin-bottom: 4px;
    text-transform: capitalize;
}
.weather-location {
    font-size: 10px;
    color: #000;
    margin-top: 2px;
    font-weight: 500;
}
.weather-updating {
    font-size: 9px;
    color: #666;
    margin-top: 8px;
    font-style: italic;
}
.weather-feels-like {
    font-size: 10px;
    color: #666;
    margin-top: 2px;
}
.weather-uv {
    display: flex;
    align-items: center;
    margin-top: 8px;
    padding: 5px;
    border-radius: 4px;
    font-size: 11px;
    font-weight: 500;
}
.uv-low {
    background-color: rgba(39, 174, 96, 0.2);
    color: var(--uv-low);
}
.uv-moderate {
    background-color: rgba(243, 156, 18, 0.2);
    color: var(--uv-moderate);
}
.uv-high {
    background-color: rgba(230, 126, 34, 0.2);
    color: var(--uv-high);
}
.uv-very-high {
    background-color: rgba(231, 76, 60, 0.2);
    color: var(--uv-very-high);
}
.uv-extreme {
    background-color: rgba(155, 89, 182, 0.2);
    color: var(--uv-extreme);
}
.weather-warning {
    margin-top: 8px;
    padding: 6px;
    border-radius: 4px;
    font-size: 10px;
    background-color: rgba(231, 76, 60, 0.2);
    color: #d32f2f;
    display: none;
    line-height: 1.3;
    white-space: normal;
    text-align: left;
}
.weather-sources {
    font-size: 9px;
    color: #666;
    margin-top: 8px;
    border-top: 1px solid #eee;
    padding-top: 8px;
}
.weather-loading {
    display: none;
    font-size: 10px;
    color: #666;
    margin-top: 5px;
}
.weather-widget.loading .weather-loading {
    display: block;
}
.weather-details {
    display: grid;
    grid-template-columns: repeat(2, 1fr);
    gap: 5px;
    margin-top: 8px;
    font-size: 10px;
}
.weather-detail-item {
    display: flex;
    align-items: center;
}
.weather-detail-icon {
    margin-right: 4px;
    font-size: 12px;
}

@media (min-width: 768px) {
    .weather-widget {
        top: 130px;
        left: 15px;
    }
}
```

```
.control-panel-container {
    position: fixed;
    top: 15px;
    right: 0;
    z-index: 1000;
    display: flex;
    overflow: hidden;
    pointer-events: none;
}
.control-panel {
    background: rgba(255, 255, 255, 0.85);
    padding: 15px;
    border-radius: 12px 0 0 12px;
    box-shadow: 0 5px 15px rgba(0,0,0,0.1);
    width: 280px;
    transform: translateX(100%);
    transition: transform 0.3s ease;
    border: 1px solid var(--red-light);
    position: relative;
    right: -100%;
    max-height: 80vh;
    overflow-y: auto;
    -webkit-overflow-scrolling: touch;
    pointer-events: auto;
    backdrop-filter: blur(5px);
}
.control-panel.visible {
    transform: translateX(0);
    right: 0;
}
.toggle-panel {
    background: rgba(255,255,255,var(--transparency));
    border: none;
    width: 40px;
    height: 60px;
    border-radius: 8px 0 0 8px;
    cursor: pointer;
    display: flex;
```

```css
    align-items: center;
    justify-content: center;
    box-shadow: -3px 0 5px rgba(0,0,0,0.1);
    margin-left: 5px;
    flex-shrink: 0;
    pointer-events: auto;
    backdrop-filter: blur(5px);
}
.toggle-panel:hover {
    background: var(--red-light);
}
.hamburger-icon {
    display: flex;
    flex-direction: column;
    gap: 4px;
    width: 20px;
    height: 16px;
}
.hamburger-line {
    height: 2px;
    background-color: var(--red-dark);
    transition: all 0.3s ease;
}
.control-panel.visible ~ .toggle-panel .hamburger-line:nth-child(1) {
    transform: translateY(6px) rotate(45deg);
}
.control-panel.visible ~ .toggle-panel .hamburger-line:nth-child(2) {
    opacity: 0;
}
.control-panel.visible ~ .toggle-panel .hamburger-line:nth-child(3) {
    transform: translateY(-6px) rotate(-45deg);
}
.vehicle-card {
    margin-bottom: 12px;
    padding-bottom: 12px;
    border-bottom: 1px solid var(--red-light);
}
.vehicle-card:last-child {
    border-bottom: none;
}
.vehicle-name {
    font-weight: 600;
    color: #000000;
    display: flex;
    align-items: center;
}
.vehicle-status {
    display: flex;
    justify-content: space-between;
    font-size: 14px;
    margin-top: 5px;
}
.vehicle-distance {
    display: flex;
    justify-content: space-between;
    font-size: 12px;
    margin-top: 3px;
    color: #666;
}
.distance-indicator {
    display: flex;
    align-items: center;
    font-weight: 500;
}
.distance-close {
    color: var(--distance-close);
}
.distance-medium {
    color: var(--distance-medium);
}
.distance-far {
    color: var(--distance-far);
}
.distance-trend {
    margin-left: 5px;
    font-size: 14px;
```

```
}
.trend-approaching {
    color: var(--distance-close);
}
.trend-moving-away {
    color: var(--distance-far);
}
.trend-stable {
    color: #666;
}
.online { color: var(--online-color); }
.offline { color: #000000; }
.logout-btn {
    background: var(--red-primary);
    color: white;
    border: none;
    padding: 10px;
    width: 100%;
    border-radius: 6px;
    cursor: pointer;
    font-weight: 600;
    margin-top: 15px;
    transition: background 0.3s;
}
.logout-btn:hover {
    background: var(--red-dark);
}
.diagnostic-panel-container {
    display: none;
}
.user-icon {
    background-size: 70%;
    background-repeat: no-repeat;
    background-position: center;
    width: 32px;
    height: 32px;
    border-radius: 50%;
    border: 2px solid white;
```

```
            box-shadow: 0 0 5px rgba(0,0,0,0.5);
            margin-right: 8px;
            background-image: url('data:image/svg+xml;utf8,<svg xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 24 24" fill="white"><path d="M12 12c2.21 0 4-1.79 4-4s-1.79-4-4-4-4 1.79-4 4 1.79
4 4 4zm0 2c-2.67 0-8 1.34-8 4v2h16v-2c0-2.66-5.33-4-8-4z"/></svg>');
        }
        .user-icon.online {
            background-color: var(--online-color);
        }
        .user-icon.offline {
            background-color: var(--offline-color);
            opacity: 0.7;
        }
        .vehicle-icon {
            background-size: 70%;
            background-repeat: no-repeat;
            background-position: center;
            width: 32px;
            height: 32px;
            border-radius: 50%;
            border: 2px solid white;
            box-shadow: 0 0 5px rgba(0,0,0,0.5);
            margin-right: 8px;
        }
        .vehicle-icon.online {
            background-color: var(--online-color);
            background-image: url('data:image/svg+xml;utf8,<svg xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 24 24" fill="white"><path d="M18.92 6.01C18.72 5.42 18.16 5 17.5 5h-11c-.66
0-1.21.42-1.42 1.01L3 12v8c0 .55.45 1 1 1h1c.55 0 1-.45 1-1v-1h12v1c0 .55.45 1 1 1h1c.55 0 1-.45
1-1v-8l-2.08-5.99zM6.85 7h10.29l1.08 3.11H5.77L6.85 7zM19 17H5v-5h14v5z"/><circle cx="7.5"
cy="14.5" r="1.5"/><circle cx="16.5" cy="14.5" r="1.5"/></svg>');
        }
        .vehicle-icon.offline {
            background-color: var(--offline-color);
            background-image: url('data:image/svg+xml;utf8,<svg xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 24 24" fill="white"><path d="M18.92 6.01C18.72 5.42 18.16 5 17.5 5h-11c-.66
0-1.21.42-1.42 1.01L3 12v8c0 .55.45 1 1 1h1c.55 0 1-.45 1-1v-1h12v1c0 .55.45 1 1 1h1c.55 0 1-.45
1-1v-8l-2.08-5.99zM6.85 7h10.29l1.08 3.11H5.77L6.85 7zM19 17H5v-5h14v5z"/><circle cx="7.5"
cy="14.5" r="1.5"/><circle cx="16.5" cy="14.5" r="1.5"/></svg>');
            opacity: 0.7;
        }
        .vehicle-marker {
            background-size: 70%;
            background-repeat: no-repeat;
            background-position: center;
            width: 32px;
            height: 32px;
            border-radius: 50%;
            border: 2px solid white;
            box-shadow: 0 0 5px rgba(0,0,0,0.5);
        }
        .vehicle-marker.online {
            background-color: #27ae60;
            background-image: url('data:image/svg+xml;utf8,<svg xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 24 24" fill="white"><path d="M18.92 6.01C18.72 5.42 18.16 5 17.5 5h-11c-.66
0-1.21.42-1.42 1.01L3 12v8c0 .55.45 1 1 1h1c.55 0 1-.45 1-1v-1h12v1c0 .55.45 1 1 1h1c.55 0 1-.45
1-1v-8l-2.08-5.99zM6.85 7h10.29l1.08 3.11H5.77L6.85 7zM19 17H5v-5h14v5z"/><circle cx="7.5"
cy="14.5" r="1.5"/><circle cx="16.5" cy="14.5" r="1.5"/></svg>');
        }
        .vehicle-marker.offline {
            background-color: #95a5a6;
            background-image: url('data:image/svg+xml;utf8,<svg xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 24 24" fill="white"><path d="M18.92 6.01C18.72 5.42 18.16 5 17.5 5h-11c-.66
0-1.21.42-1.42 1.01L3 12v8c0 .55.45 1 1 1h1c.55 0 1-.45 1-1v-1h12v1c0 .55.45 1 1 1h1c.55 0 1-.45
1-1v-8l-2.08-5.99zM6.85 7h10.29l1.08 3.11H5.77L6.85 7zM19 17H5v-5h14v5z"/><circle cx="7.5"
cy="14.5" r="1.5"/><circle cx="16.5" cy="14.5" r="1.5"/></svg>');
            opacity: 0.7;
        }
        .vehicle-label {
            position: absolute;
            bottom: -10px;
            left: 50%;
            transform: translateX(-50%);
            white-space: nowrap;
            font-size: 12px;
            font-weight: bold;
```

```
    color: #333;
    text-shadow: 0 0 3px white;
    pointer-events: none;
}
.refresh-container {
    position: fixed;
    bottom: 10px;
    left: 0;
    right: 0;
    display: flex;
    justify-content: center;
    z-index: 1000;
    pointer-events: none;
}
.refresh-btn {
    background: var(--red-primary);
    color: white;
    border: none;
    padding: 10px 20px;
    border-radius: 6px;
    cursor: pointer;
    font-weight: 600;
    display: flex;
    align-items: center;
    gap: 8px;
    box-shadow: 0 2px 10px rgba(0,0,0,0.2);
    transition: background 0.3s, transform 0.2s;
    pointer-events: auto;
}
.refresh-btn:hover {
    background: var(--red-dark);
    transform: translateY(-2px);
}
.refresh-btn:active {
    transform: translateY(0);
}
.refresh-icon {
    width: 16px;
    height: 16px;
    transition: transform 0.5s;
}
.refresh-btn.loading .refresh-icon {
    animation: spin 1s linear infinite;
}
@keyframes spin {
    from { transform: rotate(0deg); }
    to { transform: rotate(360deg); }
}
.map-provider-selector {
    position: fixed;
    bottom: 57px;
    left: 10px;
    z-index: 1000;
    background: rgba(255, 255, 255, var(--transparency));
    padding: 6px;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0,0,0,0.2);
    font-size: 10px;
    backdrop-filter: blur(5px);
}
.map-provider-selector select {
    padding: 5px;
    border-radius: 3px;
    border: none;
    background-color: transparent;
}
.status-notification {
    position: fixed;
    top: 10px;
    left: 50%;
    transform: translateX(-50%);
    background: rgba(0, 0, 0, 0.7);
    color: white;
    padding: 10px 15px;
    border-radius: 5px;
    z-index: 1001;
    display: none;
    font-size: 14px;
}
.current-user-indicator {
    position: fixed;
    top: 10px;
    left: 10px;
    z-index: 1000;
    background: rgba(255, 255, 255, var(--transparency));
    padding: 10px 15px;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0,0,0,0.2);
    font-size: 14px;
    font-weight: bold;
    color: #000000;
    backdrop-filter: blur(5px);
}

@keyframes pulse {
    0% { transform: scale(0.8); opacity: 0.6; }
    70% { transform: scale(2.5); opacity: 0; }
    100% { transform: scale(0.8); opacity: 0; }
}
.emergency-locate-btn {
    position: fixed;
    bottom: 20px;
    right: 15px;
    z-index: 1000;
    background: rgba(52, 152, 219, 0.7);
    color: white;
    border: none;
    width: 46px;
    height: 46px;
    border-radius: 50%;
    cursor: pointer;
    font-weight: bold;
    box-shadow: 0 4px 10px rgba(0,0,0,0.3);
    display: flex;
    align-items: center;
    justify-content: center;
    font-size: 24px;
}
.accuracy-circle {
    fill: rgba(52, 152, 219, 0.2);
    stroke: rgba(52, 152, 219, 0.5);
    stroke-width: 1;
}
.leaflet-control-zoom {
    margin-top: 80px !important;
}
.leaflet-bar a {
    width: 30px !important;
    height: 30px !important;
    line-height: 30px !important;
    font-size: 18px !important;
    background: rgba(255, 255, 255, var(--transparency)) !important;
    backdrop-filter: blur(5px);
}
.leaflet-container {
    touch-action: manipulation;
}
.map-touch-overlay {
    position: absolute;
    top: 0;
    right: 0;
    width: 50px;
    height: 100%;
    z-index: 999;
    display: none;
}
.control-panel.visible ~ .map-touch-overlay {
    display: block;
}
.route-selector {
    position: fixed;
    top: 60px;
    left: 10px;
    z-index: 1000;
    background: rgba(255, 255, 255, var(--transparency));
    padding: 6px;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0,0,0,0.2);
    font-size: 10px;
    backdrop-filter: blur(5px);
 }
.route-selector select {
    padding: 5px;
    border-radius: 3px;
    border: none;
    margin-top: 5px;
    width: 100%;
    background-color: transparent;
 }

 .leaflet-bottom.leaflet-right {
    bottom: 160px;
 }
 .leaflet-control-attribution {
    display: none !important;
 }

 .vehicle-number-modal {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: rgba(0,0,0,0.8);
    display: flex;
    align-items: center;
    justify-content: center;
    z-index: 2000;
 }
 .vehicle-number-form {
    background: white;
    padding: 30px;
    border-radius: 15px;
    width: 90%;
    max-width: 400px;
    text-align: center;
    box-shadow: 0 10px 30px rgba(0,0,0,0.3);
}
.vehicle-number-form h2 {
    color: var(--red-dark);
    margin-bottom: 20px;
}
.vehicle-number-input {
    width: 100%;
    padding: 15px;
    margin: 15px 0;
    border: 2px solid var(--gray-light);
    border-radius: 8px;
    font-size: 18px;
    text-align: center;
}
.vehicle-number-input:focus {
    border-color: var(--red-primary);
    outline: none;
    box-shadow: 0 0 0 3px rgba(244, 67, 54, 0.2);
}
.vehicle-number-submit {
    background: var(--red-primary);
    color: white;
    border: none;
    padding: 15px 30px;
    border-radius: 8px;
    cursor: pointer;
    font-size: 16px;
    font-weight: 600;
    transition: all 0.3s;
}
.vehicle-number-submit:hover {
```

```css
    background: var(--red-dark);
    transform: translateY(-2px);
    box-shadow: 0 5px 15px rgba(211, 47, 47, 0.3);
}

.admin-dashboard {
    display: none;
    background: rgba(255, 255, 255, var(--transparency));
    border-radius: 8px;
    padding: 15px;
    margin: 15px 0;
    box-shadow: none;
    border: none;
}
.admin-dashboard h4 {
    color: var(--red-dark);
    margin-top: 0;
    margin-bottom: 15px;
    padding-bottom: 10px;
    border-bottom: 1px solid var(--dashboard-border);
}
.dashboard-tabs {
    display: flex;
    margin-bottom: 15px;
    border-bottom: 1px solid var(--dashboard-border);
}
.dashboard-tab {
    padding: 8px 15px;
    cursor: pointer;
    border-bottom: 2px solid transparent;
    font-weight: 500;
}
.dashboard-tab.active {
    border-bottom-color: var(--red-primary);
    color: var(--red-dark);
}
.dashboard-content {
    display: none;
```

```css
}
.dashboard-content.active {
    display: block;
}
.lap-counter {
    margin-bottom: 15px;
    padding: 10px;
    background: var(--dashboard-bg);
    border-radius: 6px;
}
.lap-counter h5 {
    margin: 0 0 8px 0;
    color: var(--dashboard-text);
    font-size: 14px;
}
.lap-stats {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 8px;
}
.lap-stat {
    text-align: center;
    padding: 8px;
    border-radius: 4px;
    background: var(--dashboard-card);
    box-shadow: 0 1px 3px rgba(0,0,0,0.1);
}
.lap-stat.shift1 {
    border-top: 3px solid var(--shift1-color);
}
.lap-stat.shift2 {
    border-top: 3px solid var(--shift2-color);
}
.lap-stat.shift3 {
    border-top: 3px solid var(--shift3-color);
}
.lap-count {
    font-size: 18px;
    font-weight: bold;
    margin-bottom: 3px;
}
.lap-label {
    font-size: 11px;
    color: black;
}
.dashboard-metrics {
    display: grid;
    grid-template-columns: repeat(2, 1fr);
    gap: 10px;
    margin-top: 15px;
}
.metric-card {
    padding: 10px;
    background: var(--dashboard-bg);
    border-radius: 6px;
    text-align: center;
}
.metric-value {
    font-size: 16px;
    font-weight: bold;
    margin-bottom: 5px;
}
.metric-label {
    font-size: 12px;
    color: #666;
}

.download-btn {
    background: var(--blue-primary);
    color: white;
    border: none;
    padding: 8px 15px;
    border-radius: 6px;
    cursor: pointer;
    font-weight: 600;
    margin-top: 10px;
```

```
            display: flex;
            align-items: center;
            gap: 5px;
            transition: background 0.3s;
        }
        .download-btn:hover {
            background: var(--blue-dark);
        }
        #locationStatus {
            font-size: 12px;
            padding: 4px 8px;
            border-radius: 18px;
        }
        .export-excel-btn {
            background: #2ecc71;
            color: white;
            border: none;
            padding: 10px;
            width: 100%;
            border-radius: 6px;
            cursor: pointer;
            font-weight: 600;
            margin-top: 15px;
            transition: background 0.3s;
            display: flex;
            align-items: center;
            justify-content: center;
            gap: 8px;
            box-shadow: 0 2px 10px rgba(0, 0, 0, 0.2);
        }

        .export-excel-btn:hover {
            background: #27ae60;
        }
    </style>
</head>
<body>
    <div id="map"></div>
```

```
<div class="map-touch-overlay" id="mapTouchOverlay"></div>

<div class="weather-widget" id="weatherWidget">
  <div class="weather-icon" id="weatherIcon">🌡️</div>
  <div class="weather-info">
    <div class="weather-temp" id="weatherTemp">--°C</div>
    <div class="weather-desc" id="weatherDesc">Cargando datos...</div>
    <div class="weather-feels-like" id="weatherFeelsLike">Sensación: --°C</div>
    <div class="weather-uv" id="weatherUv">
      <span>Índice UV: --</span>
    </div>
    <div class="weather-warning" id="weatherWarning"></div>
    <div class="weather-details">
      <div class="weather-detail-item">
        <span class="weather-detail-icon"> 💧 </span>
        <span id="weatherHumidity">--%</span>
      </div>
      <div class="weather-detail-item">
        <span class="weather-detail-icon">🌬️</span>
        <span id="weatherWind">-- km/h</span>
      </div>
      <div class="weather-detail-item">
        <span class="weather-detail-icon">☁️</span>
        <span id="weatherClouds">--%</span>
      </div>
      <div class="weather-detail-item">
        <span class="weather-detail-icon">🔆</span>
        <span id="weatherPressure">-- hPa</span>
      </div>
    </div>
    <div class="weather-location">Pudahuel, Santiago</div>
    <div class="weather-updating" id="weatherUpdating"></div>
    <div class="weather-sources" id="weatherSources">Fuente: VisualCrossing</div>
    <div class="weather-loading">Actualizando datos...</div>
  </div>
</div>

<div class="status-notification" id="statusNotification"></div>
```

```
    <div class="current-user-indicator" id="currentUserIndicator">
        <span id="locationStatus">Buscando tu ubicación...</span>
    </div>

    <div class="diagnostic-panel-container">
        <div class="diagnostic-panel" id="diagnosticPanel">
            <h3 style="margin-top:0;color:var(--blue-dark);">📊 Diagnóstico GPS</h3>

            <div class="diagnostic-info">
                <div class="diagnostic-item">
                    <span>Estado GPS:</span>
                    <span id="gpsStatus" class="diagnostic-value warning">Comprobando...</span>
                </div>
                <div class="diagnostic-item">
                    <span>Precisión:</span>
                    <span id="accuracyValue" class="diagnostic-value">--</span>
                </div>
                <div class="diagnostic-item">
                    <span>Última actualización:</span>
                    <span id="lastUpdateValue" class="diagnostic-value">--</span>
                </div>
                <div class="diagnostic-item">
                    <span>Método:</span>
                    <span id="methodValue" class="diagnostic-value">--</span>
                </div>
            </div>

            <button id="forceLocationBtn" style="margin-top: 15px; width: 100%; padding: 8px;
background: var(--blue-primary); color: white; border: none; border-radius: 6px; cursor: pointer;
font-weight: 600;">
                Forzar Ubicación
            </button>
        </div>

        <button class="toggle-diagnostic-panel" id="toggleDiagnosticPanel">
            <div class="diagnostic-hamburger-icon">
                <div class="diagnostic-hamburger-line"></div>
                <div class="diagnostic-hamburger-line"></div>
                <div class="diagnostic-hamburger-line"></div>
            </div>
        </button>
    </div>

    <div class="route-selector">
        <label for="routeSelector">Ruta:</label>
        <select id="routeSelector">
            <option value="none">Sin ruta</option>
            <option value="route1">Ruta 1</option>
            <option value="route2">Ruta 2</option>
        </select>
    </div>

    <button class="emergency-locate-btn" id="emergencyLocateBtn" title="Centrar en mi
    ubicación">👤</button>

    <div class="control-panel-container">
        <div class="control-panel" id="controlPanel">
            <h3 style="margin-top:0;color:var(--red-dark);">🚐 Flota Acciona</h3>

            <div class="admin-dashboard" id="adminDashboard">
                <h4>📊 Dashboard Administrador</h4>

                <div class="dashboard-tabs">
                    <div class="dashboard-tab active" data-tab="laps">Vueltas</div>
                    <div class="dashboard-tab" data-tab="metrics">Data</div>
                    <div class="dashboard-tab" data-tab="performance">Global</div>
                </div>

                <div class="dashboard-content active" id="lapsContent">
                    <div class="lap-counter" id="movil1Laps">
                        <h5>Móvil 1</h5>
                        <div class="lap-stats">
                            <div class="lap-stat shift1">
                                <div class="lap-count" id="movil1Shift1">0</div>
                                <div class="lap-label">06:00 - 13:00</div>
```

```html
        </div>
        <div class="lap-stat shift2">
            <div class="lap-count" id="movil1Shift2">0</div>
            <div class="lap-label">13:00 - 22:00</div>
        </div>
        <div class="lap-stat shift3">
            <div class="lap-count" id="movil1Shift3">0</div>
            <div class="lap-label">22:00 - 06:00</div>
        </div>
    </div>
</div>

<div class="lap-counter" id="movil2Laps">
    <h5>Móvil 2</h5>
    <div class="lap-stats">
        <div class="lap-stat shift1">
            <div class="lap-count" id="movil2Shift1">0</div>
            <div class="lap-label">06:00 - 13:00</div>
        </div>
        <div class="lap-stat shift2">
            <div class="lap-count" id="movil2Shift2">0</div>
            <div class="lap-label">13:00 - 22:00</div>
        </div>
        <div class="lap-stat shift3">
            <div class="lap-count" id="movil2Shift3">0</div>
            <div class="lap-label">22:00 - 06:00</div>
        </div>
    </div>
</div>

<div class="lap-counter" id="movil3Laps">
    <h5>Móvil 3</h5>
    <div class="lap-stats">
        <div class="lap-stat shift1">
            <div class="lap-count" id="movil3Shift1">0</div>
            <div class="lap-label">06:00 - 13:00</div>
        </div>
        <div class="lap-stat shift2">
```

```
            <div class="lap-count" id="movil3Shift2">0</div>
            <div class="lap-label">13:00 - 22:00</div>
        </div>
        <div class="lap-stat shift3">
            <div class="lap-count" id="movil3Shift3">0</div>
            <div class="lap-label">22:00 - 06:00</div>
        </div>
    </div>
</div>

<div class="lap-counter" id="movil4Laps">
    <h5>Móvil 4</h5>
    <div class="lap-stats">
        <div class="lap-stat shift1">
            <div class="lap-count" id="movil4Shift1">0</div>
            <div class="lap-label">06:00 - 13:00</div>
        </div>
        <div class="lap-stat shift2">
            <div class="lap-count" id="movil4Shift2">0</div>
            <div class="lap-label">13:00 - 22:00</div>
        </div>
        <div class="lap-stat shift3">
            <div class="lap-count" id="movil4Shift3">0</div>
            <div class="lap-label">22:00 - 06:00</div>
        </div>
    </div>
</div>

<div class="lap-counter" id="movil5Laps">
    <h5>Móvil 5</h5>
    <div class="lap-stats">
        <div class="lap-stat shift1">
            <div class="lap-count" id="movil5Shift1">0</div>
            <div class="lap-label">06:00 - 13:00</div>
        </div>
        <div class="lap-stat shift2">
            <div class="lap-count" id="movil5Shift2">0</div>
            <div class="lap-label">13:00 - 22:00</div>
            </div>
            <div class="lap-stat shift3">
                <div class="lap-count" id="movil5Shift3">0</div>
                <div class="lap-label">22:00 - 06:00</div>
            </div>
        </div>
    </div>

    <div class="lap-counter" id="movil6Laps">
        <h5>Móvil 6</h5>
        <div class="lap-stats">
            <div class="lap-stat shift1">
                <div class="lap-count" id="movil6Shift1">0</div>
                <div class="lap-label">06:00 - 13:00</div>
            </div>
            <div class="lap-stat shift2">
                <div class="lap-count" id="movil6Shift2">0</div>
                <div class="lap-label">13:00 - 22:00</div>
            </div>
            <div class="lap-stat shift3">
                <div class="lap-count" id="movil6Shift3">0</div>
                <div class="lap-label">22:00 - 06:00</div>
            </div>
        </div>
    </div>
</div>

<div class="dashboard-content" id="metricsContent">
    <div class="dashboard-metrics">
        <div class="metric-card">
            <div class="metric-value" id="totalLaps">0</div>
            <div class="metric-label">Total Vueltas Hoy</div>
        </div>
        <div class="metric-card">
            <div class="metric-value" id="activeVehicles">0</div>
            <div class="metric-label">Vehículos Activos</div>
        </div>
        <div class="metric-card">
```

```
                    <div class="metric-value" id="avgLapTime">--</div>
                    <div class="metric-label">Tiempo Promedio</div>
                </div>
                <div class="metric-card">
                    <div class="metric-value" id="lastUpdate">--</div>
                    <div class="metric-label">Última Actualización</div>
                </div>
            </div>

            <button class="export-excel-btn" id="downloadExcelBtn">
                <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"
fill="currentColor" viewBox="0 0 16 16">
                    <path d="M5.884 6.68a.5.5 0 1 0-.768.64L7.349 10l-2.233 2.68a.5.5 0 0 0
.768.64L8 10.781l2.116 2.54a.5.5 0 0 0 .768-.641L8.651 10l2.233-2.68a.5.5 0 0 0-.768-.64L8
9.219l-2.116-2.54z"/>
                    <path d="M14 14V4.5L9.5 0H4a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h8a2 2 0 0 0
2-2zM9.5 3A1.5 1.5 0 0 0 11 4.5h2V14a1 1 0 0 1-1 1H4a1 1 0 0 1-1-1V2a1 1 0 0 1 1-1h5.5v2z"/>
                </svg>
                Descargar ult. Semana (Excel)
            </button>
        </div>

        <div class="dashboard-content" id="performanceContent">
            <div class="dashboard-metrics">
                <div class="metric-card">
                    <div class="metric-value" id="totalDistance">0 km</div>
                    <div class="metric-label">Distancia Total</div>
                </div>
                <div class="metric-card">
                    <div class="metric-value" id="avgSpeed">0 km/h</div>
                    <div class="metric-label">Velocidad Promedio</div>
                </div>
                <div class="metric-card">
                    <div class="metric-value" id="fuelEfficiency">--</div>
                    <div class="metric-label">Eficiencia Combustible</div>
                </div>
                <div class="metric-card">
                    <div class="metric-value" id="operatingHours">0 h</div>
```

```html
                <div class="metric-label">Horas Operativas</div>
            </div>
        </div>
    </div>
</div>

<div class="vehicle-card">
    <div class="vehicle-name">
        <div id="current-user-icon" class="user-icon offline"></div>
        <span id="current-user-name">Mi Ubicación</span>
    </div>
    <div class="vehicle-status">
        <span>Estado:</span>
        <span id="current-user-status" class="offline">Desconectado</span>
    </div>
    <div class="vehicle-status">
        <span>Coordenadas:</span>
        <span id="current-user-coords">--</span>
    </div>
</div>

<div id="other-vehicles-container" style="display: none;">
    <h4 style="margin: 15px 0 10px 0; color: var(--red-dark);">Carruseles:</h4>

    <div class="vehicle-card">
        <div class="vehicle-name">
            <div id="movil1-icon" class="vehicle-icon offline"></div>
            <span id="movil1-name">Móvil 1</span>
        </div>
        <div class="vehicle-status">
            <span>Estado:</span>
            <span id="movil1-status" class="offline">Desconectado</span>
        </div>
        <div class="vehicle-distance" id="movil1-distance">
            <span>Distancia:</span>
            <span class="distance-indicator">--</span>
        </div>
    </div>
```

```
<div class="vehicle-card">
    <div class="vehicle-name">
        <div id="movil2-icon" class="vehicle-icon offline"></div>
        <span id="movil2-name">Móvil 2</span>
    </div>
    <div class="vehicle-status">
        <span>Estado:</span>
        <span id="movil2-status" class="offline">Desconectado</span>
    </div>
    <div class="vehicle-distance" id="movil2-distance">
        <span>Distancia:</span>
        <span class="distance-indicator">--</span>
    </div>
</div>

<div class="vehicle-card">
    <div class="vehicle-name">
        <div id="movil3-icon" class="vehicle-icon offline"></div>
        <span id="movil3-name">Móvil 3</span>
    </div>
    <div class="vehicle-status">
        <span>Estado:</span>
        <span id="movil3-status" class="offline">Desconectado</span>
    </div>
    <div class="vehicle-distance" id="movil3-distance">
        <span>Distancia:</span>
        <span class="distance-indicator">--</span>
    </div>
</div>

<div class="vehicle-card">
    <div class="vehicle-name">
        <div id="movil4-icon" class="vehicle-icon offline"></div>
        <span id="movil4-name">Móvil 4</span>
    </div>
    <div class="vehicle-status">
        <span>Estado:</span>
```

```html
            <span id="movil4-status" class="offline">Desconectado</span>
        </div>
        <div class="vehicle-distance" id="movil4-distance">
            <span>Distancia:</span>
            <span class="distance-indicator">--</span>
         </div>
    </div>

    <div class="vehicle-card">
        <div class="vehicle-name">
            <div id="movil5-icon" class="vehicle-icon offline"></div>
            <span id="movil5-name">Móvil 5</span>
        </div>
        <div class="vehicle-status">
            <span>Estado:</span>
            <span id="movil5-status" class="offline">Desconectado</span>
        </div>
        <div class="vehicle-distance" id="movil5-distance">
            <span>Distancia:</span>
            <span class="distance-indicator">--</span>
        </div>
    </div>

    <div class="vehicle-card">
        <div class="vehicle-name">
            <div id="movil6-icon" class="vehicle-icon offline"></div>
            <span id="movil6-name">Móvil 6</span>
        </div>
        <div class="vehicle-status">
            <span>Estado:</span>
            <span id="movil6-status" class="offline">Desconectado</span>
        </div>
        <div class="vehicle-distance" id="movil6-distance">
            <span>Distancia:</span>
            <span class="distance-indicator">--</span>
        </div>
    </div>
</div>
```

```
            <button class="logout-btn" id="logoutBtn">Cerrar Sesión</button>
        </div>

        <button class="toggle-panel" id="togglePanel">
            <div class="hamburger-icon">
                <div class="hamburger-line"></div>
                <div class="hamburger-line"></div>
                <div class="hamburger-line"></div>
            </div>
        </button>
    </div>

    <div class="refresh-container" id="refreshContainer">
        <button class="refresh-btn" id="refreshBtn">
            <svg class="refresh-icon" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24"
fill="white">
                <path d="M17.65 6.35C16.2 4.9 14.21 4 12 4c-4.42 0-7.99 3.58-7.99 8s3.57 8 7.99 8c3.73
0 6.84-2.55 7.73-6h-2.08c-.82 2.33-3.04 4-5.65 4-3.31 0-6-2.69-6-6s2.69-6 6-6c1.66 0 3.14.69
4.22 1.78L13 11h7V4l-2.35 2.35z"/>
            </svg>
            Actualizar
        </button>
    </div>

    <div class="map-provider-selector">
        <label for="mapProvider">Mapa:</label>
        <select id="mapProvider">
            <option value="esri">Satelital (Recomendado)</option>
            <option value="carto">Cartográfico</option>
            <option value="osm">Híbrido</option>
        </select>
    </div>

    <div class="vehicle-number-modal" id="vehicleNumberModal">
        <div class="vehicle-number-form">
            <h2>Ingrese número de vehículo</h2>
            <input type="text" class="vehicle-number-input" id="vehicleNumberInput" placeholder="Ej:
```

```
6142" maxlength="10">
            <button class="vehicle-number-submit" id="vehicleNumberSubmit">Continuar</button>
        </div>
    </div>

    <script src="https://www.gstatic.com/firebasejs/9.0.0/firebase-app-compat.js"></script>
    <script src="https://www.gstatic.com/firebasejs/9.0.0/firebase-auth-compat.js"></script>
    <script src="https://www.gstatic.com/firebasejs/9.0.0/firebase-database-compat.js"></script>
    <script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.18.5/xlsx.full.min.js"></script>

    <script>
        const firebaseConfig = {
            apiKey: "AIzaSyDT85qotm1EpAJzJzCFeoNwGyo-VKhM6dk",
            authDomain: "gpsaccionascl2025.firebaseapp.com",
            databaseURL: "https://gpsaccionascl2025-default-rtdb.firebaseio.com",
            projectId: "gpsaccionascl2025",
            storageBucket: "gpsaccionascl2025.appspot.com",
            messagingSenderId: "742246618721",
            appId: "1:742246618721:web:72d0046987416800f07c20"
        };

        const app = firebase.initializeApp(firebaseConfig);
        const database = firebase.database();

        const SCL_COORDS = [-33.3931, -70.7858];
        const INACTIVE_COORDS = [-33.409995, -70.788647];
        const START_POINT = [-33.409455, -70.789421];
        const DETECTION_RADIUS = 100;
        const ZOOM_INICIAL = 14;
        let map;
        let currentTileLayer;
        const markers = {};
        const lastUpdateTimes = {};
        const vehicleCache = {};
        let gpsInterval;
        let wakeLock = null;
        let backgroundGeolocationWatchId = null;
        const TEN_MINUTES = 10 * 60 * 1000;
        const UPDATE_INTERVAL = 30000;
        let currentUserId = null;
        let isViewer = false;
        let userPositionMarker = null;
        let userAccuracyCircle = null;
        let lastKnownPosition = null;
        let locationUpdateInterval = null;
        let viewerUpdateInterval = null;
        let currentRouteLayer = null;
        let vehicleNames = {};

        const positionHistory = {
            Movil1: [],
            Movil2: [],
            Movil3: [],
            Movil4: [],
            Movil5: [],
            Movil6: []
        };

        const PUDALUEL_COORDS = [-33.3931, -70.7858];

        let weatherUpdateInterval = null;

        let lapCounters = {
            Movil1: { shift1: 0, shift2: 0, shift3: 0, lastLapTime: null, isNearStart: false, lastResetDate:
null },
            Movil2: { shift1: 0, shift2: 0, shift3: 0, lastLapTime: null, isNearStart: false, lastResetDate:
null },
            Movil3: { shift1: 0, shift2: 0, shift3: 0, lastLapTime: null, isNearStart: false, lastResetDate:
null },
            Movil4: { shift1: 0, shift2: 0, shift3: 0, lastLapTime: null, isNearStart: false, lastResetDate:
null },
            Movil5: { shift1: 0, shift2: 0, shift3: 0, lastLapTime: null, isNearStart: false, lastResetDate:
null },
            Movil6: { shift1: 0, shift2: 0, shift3: 0, lastLapTime: null, isNearStart: false, lastResetDate:
null }
```

```
};
let startPointMarker = null;

const route1Coordinates = [
    [-33.4095444, -70.7893019],
    [-33.4092635, -70.7895518],
    [-33.404626, -70.7897795],
    [-33.4045659, -70.7889574],
    [-33.3965734, -70.7893183],
    [-33.3966737, -70.7910012],
    [-33.3970831, -70.790958],
    [-33.3971273, -70.7919677],
    [-33.4013289, -70.7917774],
    [-33.401446, -70.795415],
    [-33.399824, -70.7955015],
    [-33.3998119, -70.7961745],
    [-33.3963198, -70.796343],
    [-33.3960964, -70.7894296],
    [-33.3966985, -70.7893961],
    [-33.3967748, -70.7909972],
    [-33.3971721, -70.7909347],
    [-33.3972203, -70.792002],
    [-33.4045345, -70.7915771],
    [-33.4044984, -70.7897934],
    [-33.40931, -70.7895175],
    [-33.4096593, -70.7891232]
];
const route2Coordinates = [
    [-33.4096489, -70.7892732],
    [-33.4093416, -70.7895579],
    [-33.4046013, -70.7898303],
    [-33.4045362, -70.7889762],
    [-33.3966243, -70.7893156],
    [-33.396535, -70.7894658],
    [-33.3960629, -70.7895151],
    [-33.3963403, -70.7977141],
    [-33.3964389, -70.7977093],
    [-33.3961615, -70.7895498],
        [-33.3966124, -70.7895158],
        [-33.3966597, -70.7909835],
        [-33.3970632, -70.7911581],
        [-33.3970881, -70.7919946],
        [-33.4044896, -70.7915298],
        [-33.4044823, -70.789894],
        [-33.4092579, -70.7896023],
        [-33.4096104, -70.7892531]
    ];
    const mapProviders = {
        esri: {
            url: 'https://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer/tile/
{z}/{y}/{x}',
            attribution: 'Tiles &copy; Esri &mdash; Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye,
Getmapping, Aerogrid, IGN, IGP, UPR-EGP, and the GIS User Community',
            maxZoom: 19
        },
        carto: {
            url: 'https://{s}.basemaps.cartocdn.com/light_all/{z}/{x}/{y}{r}.png',
            attribution: '&copy; <a href="https://www.openstreetmap.org/
copyright">OpenStreetMap</a> contributors, &copy; <a href="https://carto.com/
attributions">CARTO</a>',
            maxZoom: 20
        },
        osm: {
            url: 'https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png',
            attribution: '&copy; <a href="https://www.openstreetmap.org/
copyright">OpenStreetMap</a> contributors',
            maxZoom: 19
        }
    };

    function mapVisualCrossingIconToEmoji(iconName) {
        const iconMap = {
            'clear-day': '☀️',
            'clear-night': '🌙',
            'partly-cloudy-day': '⛅',
            'partly-cloudy-night': '🌙',
        'cloudy': '☁️',
        'rain': '🌧️',
        'snow': '❄️',
        'sleet': '🌨️',
        'wind': '💨',
        'fog': '🌫️',
        'thunderstorm': '⛈️',
        'hail': '🌨️',
        'tornado': '🌪️'
    };
    return iconMap[iconName] || '🌡️';
}

function translateWeatherCondition(condition) {
    if (!condition) return 'Condiciones variables';

    const translationMap = {
        'partly cloudy': 'Parcialmente nublado',
        'partly-cloudy': 'Parcialmente nublado',
        'partly-cloudy-day': 'Parcialmente nublado',
        'partly-cloudy-night': 'Parcialmente nublado',
        'clear': 'Despejado',
        'clear day': 'Despejado',
        'clear-night': 'Despejado',
        'sunny': 'Soleado',
        'mostly sunny': 'Mayormente soleado',
        'cloudy': 'Nublado',
        'overcast': 'Cubierto',
        'rain': 'Lluvia',
        'light rain': 'Lluvia ligera',
        'heavy rain': 'Lluvia intensa',
        'showers': 'Chubascos',
        'drizzle': 'Llovizna',
        'snow': 'Nieve',
        'light snow': 'Nieve ligera',
        'heavy snow': 'Nieve intensa',
        'sleet': 'Aguanieve',
        'wind': 'Ventoso',
            'windy': 'Ventoso',
            'fog': 'Niebla',
            'foggy': 'Neblinoso',
            'mist': 'Neblina',
            'haze': 'Bruma',
            'thunderstorm': 'Tormenta eléctrica',
            'thunder': 'Truenos',
            'hail': 'Granizo',
            'tornado': 'Tornado'
        };

        const normalizedCondition = condition.toLowerCase().replace(/-/g, ' ').trim();
        return translationMap[normalizedCondition] || condition;
    }

    const VISUAL_CROSSING_API = {
        name: "VisualCrossing",
        url: (lat, lon) => `https://weather.visualcrossing.com/VisualCrossingWebServices/rest/
services/timeline/${lat}%2C${lon}?
unitGroup=us&key=WHNF6WFYLLBQL3CVLTT62R34P&contentType=json`,
        parser: data => {
            if (!data || !data.currentConditions) return null;
            return {
                temp: Math.round((data.currentConditions.temp - 32) * 5/9),
                feels_like: Math.round((data.currentConditions.feelslike - 32) * 5/9),
                description: data.currentConditions.conditions,
                icon: data.currentConditions.icon,
                humidity: data.currentConditions.humidity,
                wind: data.currentConditions.windspeed * 1.60934,
                uv: data.currentConditions.uvindex,
                pressure: data.currentConditions.pressure,
                cloudcover: data.currentConditions.cloudcover,
                source: "VisualCrossing"
            };
        }
    };

    const OPEN_METEO_API = {
        name: "Open-Meteo",
        url: (lat, lon) => `https://api.open-meteo.com/v1/forecast?latitude=${lat}&longitude=${lon}
&current=temperature_2m,apparent_temperature,precipitation,rain,showers,snowfall,weather_code
,wind_speed_10m,relative_humidity_2m&hourly=temperature_2m,relative_humidity_2m,wind_spee-
d_10m&timezone=auto`,
        parser: data => {
            if (!data || !data.current) return null;
            return {
                temp: Math.round(data.current.temperature_2m),
                feels_like: Math.round(data.current.apparent_temperature),
                description: getWeatherCodeDescription(data.current.weather_code),
                icon: getWeatherCodeIcon(data.current.weather_code),
                humidity: data.current.relative_humidity_2m,
                wind: data.current.wind_speed_10m,
                uv: 0,
                pressure: 0,
                cloudcover: 0,
                source: "Open-Meteo"
            };
        }
    };

    function getWeatherCodeDescription(code) {
        const weatherCodes = {
            0: "Despejado",
            1: "Principalmente despejado",
            2: "Parcialmente nublado",
            3: "Nublado",
            45: "Niebla",
            48: "Niebla escarchada",
            51: "Llovizna ligera",
            53: "Llovizna moderada",
            55: "Llovizna densa",
            56: "Llovizna helada ligera",
            57: "Llovizna helada densa",
            61: "Lluvia ligera",
            63: "Lluvia moderada",
            65: "Lluvia intensa",
            66: "Lluvia helada ligera",
            67: "Lluvia helada intensa",
            71: "Nieve ligera",
            73: "Nieve moderada",
            75: "Nieve intensa",
            77: "Granos de nieve",
            80: "Chubascos ligeros",
            81: "Chubascos moderados",
            82: "Chubascos violentos",
            85: "Nevadas ligeras",
            86: "Nevadas intensas",
            95: "Tormenta eléctrica",
            96: "Tormenta eléctrica con granizo ligero",
            99: "Tormenta eléctrica con granizo intenso"
        };
        return weatherCodes[code] || "Condiciones variables";
    }

    function getWeatherCodeIcon(code) {
        const iconMap = {
            0: "☀️",
            1: "🌤️",
            2: "⛅",
            3: "☁️",
            45: "🌫️",
            48: "🌫️",
            51: "🌧️",
            53: "🌧️",
            55: "🌧️",
            56: "🌧️",
            57: "🌧️",
            61: "🌧️",
            63: "🌧️",
            65: "🌧️",
            66: "🌧️",
            67: "🌧️",
            71: "❄️",
            73: "❄️",
            75: "❄️",
            77: "❄️",
            80: "🌦️",
            81: "🌦️",
            82: "🌦️",
            85: "❄️",
            86: "❄️",
            95: "⛈️",
            96: "⛈️",
            99: "⛈️"
        };
         return iconMap[code] || "🌡️";
    }

    firebase.auth().onAuthStateChanged((user) => {
        if (!user) {
            window.location.href = 'login.html';
        } else {
            const auth = JSON.parse(localStorage.getItem('gpsAuth'));
            if (!auth) window.location.href = 'login.html';

            currentUserId = auth.username;
            isViewer = auth.isViewer;

            const savedVehicleNames = localStorage.getItem('vehicleNames');
            if (savedVehicleNames) {
                vehicleNames = JSON.parse(savedVehicleNames);
            }

            initMap();
            setupUI();

            startWeatherUpdates();

            if (!isViewer && !auth.vehicleNumber) {
                document.getElementById('vehicleNumberModal').style.display = 'flex';
            } else {
                document.getElementById('vehicleNumberModal').style.display = 'none';
```

```
                setupAfterVehicleNumber();
            }
        }
    });

    function startWeatherUpdates() {
        getWeatherData();
        weatherUpdateInterval = setInterval(getWeatherData, 600000);
        updateNextUpdateTime();
    }

    function updateNextUpdateTime() {
        const now = new Date();
        const nextUpdate = new Date(now.getTime() + 600000);
        const nextUpdateStr = nextUpdate.toLocaleTimeString('es-CL', {hour: '2-digit', minute:'2-
digit'});

        document.getElementById('weatherUpdating').textContent = `Próxima actualización: $
{nextUpdateStr}`;
    }

    async function getWeatherData() {
        const lat = PUDALUEL_COORDS[0];
        const lon = PUDALUEL_COORDS[1];

        document.getElementById('weatherWidget').classList.add('loading');
        document.getElementById('weatherDesc').textContent = 'Actualizando...';
        document.getElementById('weatherSources').textContent = 'Consultando VisualCrossing...';

        try {
            const response = await fetch(VISUAL_CROSSING_API.url(lat, lon));

            if (!response.ok) {
                throw new Error(`Error en VisualCrossing: ${response.status}`);
            }

            const data = await response.json();
            const weatherData = VISUAL_CROSSING_API.parser(data);
            if (!weatherData) {
                throw new Error("No se pudieron obtener datos del clima");
            }

            updateWeatherWidget(weatherData);

        } catch (error) {
            console.error('Error obteniendo datos del clima de VisualCrossing:', error);

            try {
                document.getElementById('weatherSources').textContent = 'Consultando Open-
Meteo...';

                const response = await fetch(OPEN_METEO_API.url(lat, lon));

                if (!response.ok) {
                    throw new Error(`Error en Open-Meteo: ${response.status}`);
                }

                const data = await response.json();
                const weatherData = OPEN_METEO_API.parser(data);

                if (!weatherData) {
                    throw new Error("No se pudieron obtener datos del clima");
                }

                updateWeatherWidget(weatherData);

            } catch (fallbackError) {
                console.error('Error obteniendo datos del clima de Open-Meteo:', fallbackError);
                document.getElementById('weatherDesc').textContent = 'Info clima, intermitente.';
                document.getElementById('weatherTemp').textContent = '--°C';
                document.getElementById('weatherFeelsLike').textContent = 'Sensación: --°C';
                document.getElementById('weatherIcon').textContent = '☀️';
                document.getElementById('weatherSources').textContent = 'Error al consultar
servicios';
            }
```

```
    } finally {
        document.getElementById('weatherWidget').classList.remove('loading');
        updateNextUpdateTime();
    }
}

function updateWeatherWidget(weatherData) {
    if (!weatherData) {
        document.getElementById('weatherDesc').textContent = 'Error al cargar';
        document.getElementById('weatherTemp').textContent = '--°C';
        document.getElementById('weatherFeelsLike').textContent = 'Sensación: --°C';
        document.getElementById('weatherIcon').textContent = '☀️';
        document.getElementById('weatherSources').textContent = 'Error en servicios';
        return;
    }

    const tempElement = document.getElementById('weatherTemp');
    const descElement = document.getElementById('weatherDesc');
    const feelsLikeElement = document.getElementById('weatherFeelsLike');
    const iconElement = document.getElementById('weatherIcon');
    const uvElement = document.getElementById('weatherUv');
    const warningElement = document.getElementById('weatherWarning');
    const sourcesElement = document.getElementById('weatherSources');
    const humidityElement = document.getElementById('weatherHumidity');
    const windElement = document.getElementById('weatherWind');
    const cloudsElement = document.getElementById('weatherClouds');
    const pressureElement = document.getElementById('weatherPressure');

    tempElement.textContent = `${weatherData.temp}°C`;
    descElement.textContent = weatherData.description;
    iconElement.textContent = mapVisualCrossingIconToEmoji(weatherData.icon);
    feelsLikeElement.textContent = `Sensación: ${weatherData.feels_like}°C`;

    if (weatherData.humidity !== undefined) {
        humidityElement.textContent = `${Math.round(weatherData.humidity)}%`;
    } else {
        humidityElement.textContent = '--%';
    }
```

```
if (weatherData.wind !== undefined) {
    windElement.textContent = `${Math.round(weatherData.wind)} km/h`;
} else {
    windElement.textContent = '-- km/h';
}

if (weatherData.cloudcover !== undefined) {
    cloudsElement.textContent = `${Math.round(weatherData.cloudcover)}%`;
} else {
    cloudsElement.textContent = '--%';
}

if (weatherData.pressure !== undefined) {
    pressureElement.textContent = `${Math.round(weatherData.pressure)} hPa`;
} else {
    pressureElement.textContent = '-- hPa';
}

if (weatherData.uv !== undefined && weatherData.uv > 0) {
    const uvIndex = weatherData.uv;
    uvElement.innerHTML = `<span>Índice UV: ${uvIndex} - ${getUvLevel(uvIndex)}</span>`;
    uvElement.className = `weather-uv ${getUvClass(uvIndex)}`;

    if (uvIndex >= 6) {
        const warningLines = getUvWarning(uvIndex);
        warningElement.style.display = 'block';
        warningElement.innerHTML = warningLines.join('<br>');
    } else {
        warningElement.style.display = 'none';
    }
} else {
    uvElement.innerHTML = '<span>Índice UV: No disponible</span>';
    uvElement.className = 'weather-uv';
    warningElement.style.display = 'none';
}

sourcesElement.textContent = `Fuente: ${weatherData.source}`;
```

```
}

function getUvLevel(uvIndex) {
    if (uvIndex >= 0 && uvIndex <= 2) return 'Bajo';
    if (uvIndex >= 3 && uvIndex <= 5) return 'Moderado';
    if (uvIndex >= 6 && uvIndex <= 7) return 'Alto';
    if (uvIndex >= 8 && uvIndex <= 10) return 'Muy Alto';
    return 'Extremo';
}

function getUvClass(uvIndex) {
    if (uvIndex >= 0 && uvIndex <= 2) return 'uv-low';
    if (uvIndex >= 3 && uvIndex <= 5) return 'uv-moderate';
    if (uvIndex >= 6 && uvIndex <= 7) return 'uv-high';
    if (uvIndex >= 8 && uvIndex <= 10) return 'uv-very-high';
    return 'uv-extreme';
}

function getUvWarning(uvIndex) {
    if (uvIndex >= 6 && uvIndex <= 7) {
        return ['⚠️ Protección necesaria.', 'Use ERP, gorro y protector solar.'];
    } else if (uvIndex >= 8 && uvIndex <= 10) {
        return ['⚠️ Protección necesaria.', 'Use ERP, gorro y protector solar.'];
    } else if (uvIndex > 10) {
        return ['⚠️ ¡Extremo! Tome precauciones.', 'Use ERP, gorro y protector solar.'];
    }
    return [];
}

function toggleWeatherWidget() {
    const widget = document.getElementById('weatherWidget');
    widget.classList.toggle('expanded');
}

function setupAfterVehicleNumber() {
    const auth = JSON.parse(localStorage.getItem('gpsAuth'));
    if (auth.vehicleNumber) {
        vehicleNames[currentUserId] = auth.vehicleNumber;
```

```
            localStorage.setItem('vehicleNames', JSON.stringify(vehicleNames));
        }

        const displayName = (currentUserId === 'Funcionarios') ? 'Yo' :
(vehicleNames[currentUserId] || currentUserId);
        document.getElementById('current-user-name').textContent = displayName;

        if (currentUserId === 'Funcionarios') {
            document.getElementById('refreshContainer').style.display = 'none';
        }

        if (currentUserId === 'Admin') {
    document.getElementById('adminDashboard').style.display = 'block';
    loadLapCounters();
    setupDashboardTabs();

    document.getElementById('downloadExcelBtn').addEventListener('click', downloadExcelData);
    document.getElementById('downloadExcelBtn').style.display = 'flex';

    setTimeout(() => {
        document.getElementById('refreshBtn').click();
    }, 5000);
} else if (currentUserId.startsWith('Movil')) {
    document.getElementById('refreshContainer').style.display = 'none';
}

        if (isViewer || currentUserId === 'Funcionarios' || currentUserId === 'Admin') {
            document.getElementById('other-vehicles-container').style.display = 'block';
            updateVehicleNamesInUI();
            initializeAllMarkers();
            startMonitoring();
        } else {
            initializeCurrentUserMarker();
            startGPSTracking(currentUserId);
        }

        document.getElementById('weatherWidget').addEventListener('click',
toggleWeatherWidget);
```

```
}

function setupDashboardTabs() {
    const tabs = document.querySelectorAll('.dashboard-tab');
    tabs.forEach(tab => {
        tab.addEventListener('click', () => {
            tabs.forEach(t => t.classList.remove('active'));
            tab.classList.add('active');

            document.querySelectorAll('.dashboard-content').forEach(content => {
                content.classList.remove('active');
            });

            const tabId = tab.getAttribute('data-tab');
            document.getElementById(`${tabId}Content`).classList.add('active');
        });
    });
}

function loadLapCounters() {
    const savedLapCounters = localStorage.getItem('lapCounters');
    if (savedLapCounters) {
        const savedData = JSON.parse(savedLapCounters);

        const today = new Date().toDateString();
        Object.keys(savedData).forEach(vehicle => {
            if (savedData[vehicle].lastResetDate !== today) {
                savedData[vehicle].shift1 = 0;
                savedData[vehicle].shift2 = 0;
                savedData[vehicle].shift3 = 0;
                savedData[vehicle].lastResetDate = today;
            }
        });

        lapCounters = savedData;
        Object.keys(lapCounters).forEach(vehicle => {
            updateLapCounterUI(vehicle);
        });
        } else {
            const today = new Date().toDateString();
            Object.keys(lapCounters).forEach(vehicle => {
                lapCounters[vehicle].lastResetDate = today;
            });
        }
        updateMetrics();
    }

    function saveLapCounters() {
        localStorage.setItem('lapCounters', JSON.stringify(lapCounters));
    }

    function updateLapCounterUI(vehicle) {
        const counter = lapCounters[vehicle];
        document.getElementById(`${vehicle.toLowerCase()}Shift1`).textContent = counter.shift1;
        document.getElementById(`${vehicle.toLowerCase()}Shift2`).textContent = counter.shift2;
        document.getElementById(`${vehicle.toLowerCase()}Shift3`).textContent = counter.shift3;
    }

    function updateMetrics() {
        let totalLaps = 0;
        Object.keys(lapCounters).forEach(vehicle => {
            totalLaps += lapCounters[vehicle].shift1 + lapCounters[vehicle].shift2 +
lapCounters[vehicle].shift3;
        });
        document.getElementById('totalLaps').textContent = totalLaps;

        let activeCount = 0;
        Object.keys(vehicleCache).forEach(vehicle => {
            if (vehicleCache[vehicle] && vehicleCache[vehicle].status === 'online') {
                const lastUpdate = vehicleCache[vehicle].lastUpdate || 0;
                const isOnline = (Date.now() - lastUpdate) < TEN_MINUTES;
                if (isOnline) {
                    activeCount++;
                }
            }
        });
```

```
            document.getElementById('activeVehicles').textContent = activeCount;

            const now = new Date();
            const chileTime = new Date(now.toLocaleString("en-US", {timeZone: "America/Santiago"}));
            document.getElementById('lastUpdate').textContent = chileTime.toLocaleTimeString('es-
CL');

            let totalDistance = 0;
            Object.keys(lapCounters).forEach(vehicle => {
                totalDistance += (lapCounters[vehicle].shift1 + lapCounters[vehicle].shift2 +
lapCounters[vehicle].shift3) * 15;
            });
            document.getElementById('totalDistance').textContent = totalDistance + ' km';

            document.getElementById('avgSpeed').textContent = '20 km/h';
            document.getElementById('fuelEfficiency').textContent = '5.2 km/L';
            document.getElementById('operatingHours').textContent = '8.5 h';
        }

        function downloadExcelData() {
            const coordinateData = getCoordinateDataForLast10Days();

            if (coordinateData.length === 0) {
                showNotification("No hay datos para exportar", 3000);
                return;
            }

            const wb = XLSX.utils.book_new();
            const ws = XLSX.utils.json_to_sheet(coordinateData);
            XLSX.utils.book_append_sheet(wb, ws, "Coordenadas");

            const today = new Date();
            const fileName = `coordenadas_${today.getDate()}-${today.getMonth()+1}-$
{today.getFullYear()}.xlsx`;

            XLSX.writeFile(wb, fileName);

            showNotification("Datos descargados exitosamente", 3000);
```

```
    }

    function getCoordinateDataForLast10Days() {
        const storedData = localStorage.getItem('coordinateData');

        if (!storedData) return [];

        const allData = JSON.parse(storedData);

        const tenDaysAgo = new Date();
        tenDaysAgo.setDate(tenDaysAgo.getDate() - 10);

        const filteredData = allData.filter(item => {
            return new Date(item.timestamp) >= tenDaysAgo;
        });

        return filteredData.map(item => {
            const date = new Date(item.timestamp);
            const chileDate = new Date(date.toLocaleString("en-US", {timeZone: "America/
Santiago"}));

            return {
                'Número de Equipo': item.vehicleName,
                'ID del Vehículo': item.vehicleId,
                'Latitud': item.latitude,
                'Longitud': item.longitude,
                'Estado': item.status,
                'Número de Vueltas': item.laps,
                'Fecha': chileDate.toLocaleDateString('es-CL'),
                'Hora': chileDate.toLocaleTimeString('es-CL')
            };
        });
    }

    function storeCoordinateData(vehicleId, coords, status, laps) {
        const storedData = localStorage.getItem('coordinateData');
        let coordinateData = storedData ? JSON.parse(storedData) : [];
        coordinateData.push({
            vehicleId: vehicleId,
            vehicleName: vehicleNames[vehicleId] || vehicleId,
            latitude: coords[0],
            longitude: coords[1],
            status: status,
            laps: laps || 0,
            timestamp: new Date().toISOString()
        });

        const tenDaysAgo = new Date();
        tenDaysAgo.setDate(tenDaysAgo.getDate() - 10);

        coordinateData = coordinateData.filter(item => {
            return new Date(item.timestamp) >= tenDaysAgo;
        });

        localStorage.setItem('coordinateData', JSON.stringify(coordinateData));
    }

    function checkLapCompletion(vehicle, coords) {
        const distance = calculateDistance(
            coords[0], coords[1],
            START_POINT[0], START_POINT[1]
        );

        if (distance <= DETECTION_RADIUS && !lapCounters[vehicle].isNearStart) {
            lapCounters[vehicle].isNearStart = true;
        }
        else if (distance > DETECTION_RADIUS && lapCounters[vehicle].isNearStart) {
            lapCounters[vehicle].isNearStart = false;
            registerLap(vehicle);
        }
    }

    function calculateDistance(lat1, lon1, lat2, lon2) {
        const R = 6371000;
        const dLat = (lat2 - lat1) * Math.PI / 180;
        const dLon = (lon2 - lon1) * Math.PI / 180;
        const a =
            Math.sin(dLat/2) * Math.sin(dLat/2) +
            Math.cos(lat1 * Math.PI / 180) * Math.cos(lat2 * Math.PI / 180) *
            Math.sin(dLon/2) * Math.sin(dLon/2);
        const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
        return R * c;
    }

    function registerLap(vehicle) {
        const now = new Date();
        const hour = now.getHours();
        let shift;

        let adjustedHour = hour;
        let currentDate = now.toDateString();

        if (hour >= 22) {
            shift = 'shift3';
        } else if (hour < 6) {
            shift = 'shift3';
            currentDate = new Date(now.getTime() - 6 * 60 * 60 * 1000).toDateString();
        } else if (hour >= 6 && hour < 13) {
            shift = 'shift1';
        } else {
            shift = 'shift2';
        }

        if (lapCounters[vehicle].lastResetDate !== currentDate) {
            lapCounters[vehicle].shift1 = 0;
            lapCounters[vehicle].shift2 = 0;
            lapCounters[vehicle].shift3 = 0;
            lapCounters[vehicle].lastResetDate = currentDate;
        }

        lapCounters[vehicle][shift]++;
        lapCounters[vehicle].lastLapTime = now.getTime();
        updateLapCounterUI(vehicle);
        updateMetrics();
        saveLapCounters();

        showNotification(`¡${vehicle} completó una vuelta en el turno ${shift}!`, 5000);
    }

    function updateVehicleNamesInUI() {
        for (let i = 1; i <= 6; i++) {
            const mobileId = 'Movil' + i;
            const element = document.getElementById(`${mobileId.toLowerCase()}-name`);
            if (element && vehicleNames[mobileId]) {
                element.textContent = vehicleNames[mobileId];
            }
        }
    }

    function initMap() {
        map = L.map('map', {
            zoomControl: false,
            tap: false,
            dragging: true,
            preferCanvas: true,
            fadeAnimation: false,
            markerZoomAnimation: false
        }).setView(SCL_COORDS, ZOOM_INICIAL);

        L.control.zoom({
            position: 'bottomright'
        }).addTo(map);

        const savedProvider = localStorage.getItem('mapProvider') || 'esri';
        document.getElementById('mapProvider').value = savedProvider;
        setMapProvider(savedProvider);

        L.marker(SCL_COORDS)
            .addTo(map)
            .bindPopup('✈️ Aeropuerto SCL')
            .openPopup();

        createUserPositionMarker(SCL_COORDS);
    }

    function setMapProvider(providerKey) {
        if (currentTileLayer) {
            map.removeLayer(currentTileLayer);
        }

        const provider = mapProviders[providerKey];
        currentTileLayer = L.tileLayer(provider.url, {
            attribution: provider.attribution,
            maxZoom: provider.maxZoom,
            updateWhenIdle: true,
            reuseTiles: true,
            detectRetina: false
        }).addTo(map);

        localStorage.setItem('mapProvider', providerKey);
    }

    function setupUI() {
        document.getElementById('logoutBtn').addEventListener('click', logout);
        const panel = document.getElementById('controlPanel');
        const toggleBtn = document.getElementById('togglePanel');

        toggleBtn.addEventListener('click', function() {
            panel.classList.toggle('visible');
        });

        const refreshBtn = document.getElementById('refreshBtn');
        refreshBtn.addEventListener('click', function() {
            refreshBtn.classList.add('loading');

            if (isViewer || currentUserId === 'Funcionarios' || currentUserId === 'Admin') {
                fetchVehicleData();
            }
            setTimeout(() => {
                const isPanelVisible = panel.classList.contains('visible');
                localStorage.setItem('panelState', isPanelVisible ? 'open' : 'closed');
                refreshBtn.classList.remove('loading');
            }, 500);
        });

        document.getElementById('mapProvider').addEventListener('change', function(e) {
            setMapProvider(e.target.value);
        });

        document.getElementById('routeSelector').addEventListener('change', function(e) {
            showRoute(e.target.value);
        });

        document.getElementById('forceLocationBtn').addEventListener('click', function() {
            forceLocationUpdate();
        });

        document.getElementById('emergencyLocateBtn').addEventListener('click', function() {
            centerOnUserLocation();
        });

        document.getElementById('vehicleNumberSubmit').addEventListener('click', function() {
            const vehicleNumber = document.getElementById('vehicleNumberInput').value.trim();

            if (!vehicleNumber) {
                alert('Por favor ingrese un número de vehículo');
                return;
            }

            const auth = JSON.parse(localStorage.getItem('gpsAuth'));
            auth.vehicleNumber = vehicleNumber;
            localStorage.setItem('gpsAuth', JSON.stringify(auth));

            document.getElementById('vehicleNumberModal').style.display = 'none';
            setupAfterVehicleNumber();
```

```
        });

        document.getElementById('vehicleNumberInput').addEventListener('keypress', function(e) {
            if (e.key === 'Enter') {
                document.getElementById('vehicleNumberSubmit').click();
            }
        });

        document.addEventListener('visibilitychange', handleVisibilityChange);

        if ('wakeLock' in navigator) {
            document.addEventListener('visibilitychange', async () => {
                if (document.visibilityState === 'visible') {
                    await requestWakeLock();
                }
            });
        }

        const mapTouchOverlay = document.getElementById('mapTouchOverlay');
        mapTouchOverlay.addEventListener('click', function() {
            panel.classList.remove('visible');
        });
    }

    function showRoute(routeId) {
        if (currentRouteLayer) {
            map.removeLayer(currentRouteLayer);
            currentRouteLayer = null;
        }

        if (routeId === 'none') return;
        let coordinates;
        let color;

        if (routeId === 'route1') {
            coordinates = route1Coordinates;
            color = '#e74c3c';
        } else if (routeId === 'route2') {
```

```
        coordinates = route2Coordinates;
        color = '#9b59b6';
    }

    currentRouteLayer = L.polyline(coordinates, {
        color: color,
        weight: 4,
        opacity: 0.7,
        smoothFactor: 1
    }).addTo(map);

    const bounds = L.latLngBounds(coordinates);
    map.fitBounds(bounds, { padding: [50, 50] });
}

function handleVisibilityChange() {
    const notification = document.getElementById('statusNotification');
    if (document.visibilityState === 'visible') {
        notification.textContent = 'Aplicación en primer plano';
        notification.style.display = 'block';
        setTimeout(() => { notification.style.display = 'none'; }, 3000);
        if (!isViewer) {
            forceLocationUpdate();
        }
    } else {
        notification.textContent = 'Aplicación en segundo plano - El GPS sigue activo';
        notification.style.display = 'block';
        setTimeout(() => { notification.style.display = 'none'; }, 3000);
    }
}

async function requestWakeLock() {
    try {
        if ('wakeLock' in navigator) {
            wakeLock = await navigator.wakeLock.request('screen');
            wakeLock.addEventListener('release', () => {
            });
        }
        } catch (err) {
        }
    }

    function initializeAllMarkers() {
        const mobileIds = ['Movil1', 'Movil2', 'Movil3', 'Movil4', 'Movil5', 'Movil6'];
        mobileIds.forEach(mobileId => {
            createMarker(mobileId, INACTIVE_COORDS, false);
            lastUpdateTimes[mobileId] = Date.now();
        });

        if (currentUserId === 'Admin') {
            fetchVehicleData();
        }
    }

    function initializeCurrentUserMarker() {
        createMarker(currentUserId, INACTIVE_COORDS, false);
        lastUpdateTimes[currentUserId] = Date.now();
    }

    function createUserPositionMarker(coords) {
        if (userPositionMarker) {
            map.removeLayer(userPositionMarker);
        }
        if (userAccuracyCircle) {
            map.removeLayer(userAccuracyCircle);
        }

        const userIcon = L.divIcon({
            className: 'user-position-marker',
            html: `
                <div style="background-color: #3498db; width: 20px; height: 20px; border-radius:
50%; border: 3px solid white; box-shadow: 0 0 10px rgba(0,0,0,0.5);"></div>
                <div class="pulse-circle"></div>
            `,
            iconSize: [20, 20],
            iconAnchor: [10, 10]
        });

        userPositionMarker = L.marker(coords, { icon: userIcon, zIndexOffset: 1000 })
            .addTo(map)
            .bindPopup('<b>Tu ubicación actual</b>');

        map.setView(coords, 16);
    }

    function updateUserPositionMarker(coords, accuracy) {
        if (!userPositionMarker) {
            createUserPositionMarker(coords);
        } else {
            userPositionMarker.setLatLng(coords);
        }

        if (userAccuracyCircle) {
            map.removeLayer(userAccuracyCircle);
        }

        if (accuracy && accuracy < 1000) {
            userAccuracyCircle = L.circle(coords, {
                radius: accuracy,
                className: 'accuracy-circle'
            }).addTo(map);
        }

        document.getElementById('current-user-coords').textContent =
            `${coords[0].toFixed(6)}, ${coords[1].toFixed(6)}`;
        document.getElementById('accuracyValue').textContent =
            accuracy ? `${accuracy.toFixed(1)} metros` : 'Desconocida';

        const now = new Date();
        const chileTime = new Date(now.toLocaleString("en-US", {timeZone: "America/Santiago"}));
        document.getElementById('lastUpdateValue').textContent =
chileTime.toLocaleTimeString('es-CL');

        const locationStatus = document.getElementById('locationStatus');
        if (accuracy && accuracy < 1000) {
            updateDiagnostic('good', 'Ubicación actualizada', accuracy);
            locationStatus.textContent = 'Ubicación activa';
            locationStatus.style.color = 'green';
        } else if (accuracy) {
            updateDiagnostic('warning', 'Precisión limitada', accuracy);
            locationStatus.textContent = 'Precisión limitada';
            locationStatus.style.color = 'orange';
        } else {
            updateDiagnostic('bad', 'Sin datos de precisión');
            locationStatus.textContent = 'Ubicación inactiva';
            locationStatus.style.color = 'red';
        }
    }

    function startGPSTracking(mobileId) {
        if (locationUpdateInterval) clearInterval(locationUpdateInterval);

        requestWakeLock();

        updatePosition(mobileId);

        locationUpdateInterval = setInterval(() => {
            updatePosition(mobileId);
        }, 60000);

        setupBackgroundGeolocation(mobileId);

        setTimeout(() => updatePosition(mobileId), 2000);
    }

    function setupBackgroundGeolocation(mobileId) {
        if ('geolocation' in navigator) {
            const geoOptions = {
                enableHighAccuracy: true,
                maximumAge: 10000,
                timeout: 10000
            };
            if (backgroundGeolocationWatchId !== null) {
                navigator.geolocation.clearWatch(backgroundGeolocationWatchId);
            }

            backgroundGeolocationWatchId = navigator.geolocation.watchPosition(
                (position) => {
                    const coords = [
                        position.coords.latitude,
                        position.coords.longitude
                    ];
                    const accuracy = position.coords.accuracy;

                    updateDiagnostic('good', 'watchPosition', accuracy);
                    savePosition(mobileId, coords, accuracy);
                },
                (error) => {
                    updateDiagnostic('bad', `Error: ${getGeolocationErrorText(error)}`);
                    updateStatus(mobileId, 'offline');
                },
                geoOptions
            );
        } else {
            updateDiagnostic('bad', 'Geolocalización no soportada');
        }
    }

    function updatePosition(mobileId) {
        if (navigator.geolocation) {
            navigator.geolocation.getCurrentPosition(
                position => {
                    const coords = [
                        position.coords.latitude,
                        position.coords.longitude
                    ];
                    const accuracy = position.coords.accuracy;

                    updateDiagnostic('good', 'getCurrentPosition', accuracy);
                    savePosition(mobileId, coords, accuracy);
                },
                error => {
                    navigator.geolocation.getCurrentPosition(
                        position => {
                            const coords = [
                                position.coords.latitude,
                                position.coords.longitude
                            ];
                            const accuracy = position.coords.accuracy;

                            updateDiagnostic('warning', 'Método alternativo', accuracy);
                            savePosition(mobileId, coords, accuracy);
                        },
                        error2 => {
                            console.error('Error GPS con método alternativo:', error2);
                            updateDiagnostic('bad', `Error: ${getGeolocationErrorText(error2)}`);
                            updateStatus(mobileId, 'offline');

                            if (lastKnownPosition) {
                                updateDiagnostic('warning', 'Usando última posición conocida');
                                savePosition(mobileId, lastKnownPosition, 1000);
                            }
                        },
                        {
                            enableHighAccuracy: false,
                            timeout: 15000,
                            maximumAge: 300000
                        }
                    );
                },
                {
                    enableHighAccuracy: true,
                    timeout: 10000,
                    maximumAge: 0
                }
            );
        } else {
                updateDiagnostic('bad', 'Geolocalización no soportada');
                alert("Tu navegador no soporta geolocalización");
            }
        }

        function forceLocationUpdate() {
            if (!currentUserId) return;
            updateDiagnostic('warning', 'Forzando actualización...');

            navigator.geolocation.getCurrentPosition(
                position => {
                    const coords = [
                        position.coords.latitude,
                        position.coords.longitude
                    ];
                    const accuracy = position.coords.accuracy;

                    updateDiagnostic('good', 'Forzado exitoso', accuracy);
                    savePosition(currentUserId, coords, accuracy);

                    showNotification('Ubicación actualizada forzosamente', 3000);
                },
                error => {
                    updateDiagnostic('bad', `Forzado fallido: ${getGeolocationErrorText(error)}`);
                    showNotification('Error al forzar ubicación', 3000);
                },
                {
                    enableHighAccuracy: true,
                    timeout: 20000,
                    maximumAge: 0
                }
            );
        }

        function centerOnUserLocation() {
            if (lastKnownPosition) {
                map.setView(lastKnownPosition, 16);
                showNotification('Centrado en tu ubicación', 2000);
                } else {
                    updatePosition(currentUserId);
                    showNotification('Buscando tu ubicación...', 2000);
                }
            }

            function getGeolocationErrorText(error) {
                switch(error.code) {
                    case error.PERMISSION_DENIED:
                        return "Permiso denegado";
                    case error.POSITION_UNAVAILABLE:
                        return "Posición no disponible";
                    case error.TIMEOUT:
                        return "Tiempo agotado";
                    default:
                        return "Error desconocido";
                }
            }

            function updateDiagnostic(status, method, accuracy = null) {
                const statusElement = document.getElementById('gpsStatus');
                const methodElement = document.getElementById('methodValue');

                statusElement.textContent = status === 'good' ? 'ACTIVO' :
                                            status === 'warning' ? 'LIMITADO' : 'INACTIVO';
                statusElement.className = `diagnostic-value ${status}`;

                methodElement.textContent = method;

                if (accuracy !== null) {
                    document.getElementById('accuracyValue').textContent = `${accuracy.toFixed(1)}
        metros`;
                }

                const now = new Date();
                const chileTime = new Date(now.toLocaleString("en-US", {timeZone: "America/Santiago"}));
                document.getElementById('lastUpdateValue').textContent =
        chileTime.toLocaleTimeString('es-CL');
```

```
    const locationStatus = document.getElementById('locationStatus');
    if (status === 'good') {
        locationStatus.textContent = 'Ubicación activa';
        locationStatus.style.color = 'green';
    } else if (status === 'warning') {
        locationStatus.textContent = 'Ubicación limitada';
        locationStatus.style.color = 'orange';
    } else {
        locationStatus.textContent = 'Ubicación inactiva';
        locationStatus.style.color = 'red';
    }
}

function showNotification(message, duration) {
    const notification = document.getElementById('statusNotification');
    notification.textContent = message;
    notification.style.display = 'block';
    setTimeout(() => { notification.style.display = 'none'; }, duration);
}

function savePosition(mobileId, coords, accuracy) {
    const now = Date.now();
    lastKnownPosition = coords;

    updateUserPositionMarker(coords, accuracy);

    if (!isViewer && mobileId !== 'Funcionarios') {
        database.ref(`vehicles/${mobileId}`).set({
            coords: {
                lat: coords[0],
                lng: coords[1]
            },
            status: 'online',
            lastUpdate: now,
            accuracy: accuracy,
            displayName: vehicleNames[mobileId] || mobileId
        });
```

```
            }

            lastUpdateTimes[mobileId] = now;
            updateMarker(mobileId, coords, true);
            updateStatus(mobileId, 'online');

            if (currentUserId === 'Admin') {
                checkLapCompletion(mobileId, coords);

                const laps = lapCounters[mobileId] ?
                    lapCounters[mobileId].shift1 + lapCounters[mobileId].shift2 +
lapCounters[mobileId].shift3 : 0;
                storeCoordinateData(mobileId, coords, 'online', laps);
            }

            if (mobileId === currentUserId) {
                map.setView(coords, Math.max(map.getZoom(), 16));
            }
        }

        function startMonitoring() {
            fetchVehicleData();
            viewerUpdateInterval = setInterval(fetchVehicleData, UPDATE_INTERVAL);
        }

        function fetchVehicleData() {
            database.ref('vehicles').once('value')
                .then(snapshot => {
                    const vehiclesData = snapshot.val() || {};
                    let newDataFound = false;

                    Object.keys(vehiclesData).forEach(mobileId => {
                        const vehicle = vehiclesData[mobileId];
                        if (vehicle && vehicle.lastUpdate) {
                            if (!vehicleCache[mobileId] || vehicle.lastUpdate >
vehicleCache[mobileId].lastUpdate) {
                                vehicleCache[mobileId] = vehicle;
                                newDataFound = true;
                                if (currentUserId === 'Admin') {
                                    const coords = [vehicle.coords.lat, vehicle.coords.lng];
                                    checkLapCompletion(mobileId, coords);

                                    const laps = lapCounters[mobileId] ?
                                        lapCounters[mobileId].shift1 + lapCounters[mobileId].shift2 +
lapCounters[mobileId].shift3 : 0;
                                    storeCoordinateData(mobileId, coords, vehicle.status, laps);
                                }
                            }
                        }
                    });

                    if (newDataFound) {
                        updateMarkersFromCache();
                    } else {
                        console.log('No hay datos nuevos, usando caché.');
                        checkConnectionStatus();
                    }
                })
                .catch(error => {
                    console.error("Error al obtener datos de la flota:", error);
                    checkConnectionStatus();
                });
        }

        function updateMarkersFromCache() {
            const now = Date.now();
            Object.keys(vehicleCache).forEach(mobileId => {
                const vehicle = vehicleCache[mobileId];
                if (vehicle) {
                    const isOnline = (now - vehicle.lastUpdate) < TEN_MINUTES;
                    const coords = isOnline ? [vehicle.coords.lat, vehicle.coords.lng] : INACTIVE_COORDS;

                    if (vehicle.displayName && vehicle.displayName !== mobileId) {
                        vehicleNames[mobileId] = vehicle.displayName;
                        localStorage.setItem('vehicleNames', JSON.stringify(vehicleNames));
```

```
                    const nameElement = document.getElementById(`${mobileId.toLowerCase()}-
name`);
                    if (nameElement) {
                        nameElement.textContent = vehicle.displayName;
                    }
                }

                updateMarker(mobileId, coords, isOnline);
                updateStatus(mobileId, isOnline ? 'online' : 'offline');

                if (currentUserId === 'Funcionarios' && lastKnownPosition && isOnline) {
                    calculateAndDisplayDistance(mobileId, coords);
                }
            }
        });

        updateVehicleNamesInUI();

        if (currentUserId === 'Admin') {
            updateMetrics();
        }
    }

    function calculateAndDisplayDistance(mobileId, mobileCoords) {
        if (!lastKnownPosition) return;

        const distance = calculateDistance(
            lastKnownPosition[0], lastKnownPosition[1],
            mobileCoords[0], mobileCoords[1]
        );

        const estimatedTimeMinutes = Math.round((distance / 1000) / 30 * 60);

        const trend = calculateDistanceTrend(mobileId, mobileCoords);

        updateDistanceUI(mobileId, distance, estimatedTimeMinutes, trend);
    }
```

```
function calculateDistanceTrend(mobileId, currentCoords) {
    positionHistory[mobileId].push({
        coords: currentCoords,
        timestamp: Date.now()
    });

    if (positionHistory[mobileId].length > 5) {
        positionHistory[mobileId].shift();
    }

    if (positionHistory[mobileId].length < 2 || !lastKnownPosition) {
        return 'stable';
    }

    const previousPosition = positionHistory[mobileId][positionHistory[mobileId].length - 2];
    const previousDistance = calculateDistance(
        lastKnownPosition[0], lastKnownPosition[1],
        previousPosition.coords[0], previousPosition.coords[1]
    );

    const currentDistance = calculateDistance(
        lastKnownPosition[0], lastKnownPosition[1],
        currentCoords[0], currentCoords[1]
    );

    if (currentDistance < previousDistance - 10) {
        return 'approaching';
    } else if (currentDistance > previousDistance + 10) {
        return 'moving-away';
    } else {
        return 'stable';
    }
}

function updateDistanceUI(mobileId, distance, estimatedTimeMinutes, trend) {
    const distanceElement = document.getElementById(`${mobileId.toLowerCase()}-distance`);
    if (!distanceElement) return;
```

```
        const distanceIndicator = distanceElement.querySelector('.distance-indicator');
        if (!distanceIndicator) return;

        let formattedDistance;
        let distanceClass;

        if (distance < 1000) {
            formattedDistance = `${Math.round(distance)} m`;
            distanceClass = 'distance-close';
        } else {
            formattedDistance = `${(distance / 1000).toFixed(1)} km`;
            if (distance < 5000) {
                distanceClass = 'distance-medium';
            } else {
                distanceClass = 'distance-far';
            }
        }

       let trendIcon = '';
        if (trend === 'approaching') {
            trendIcon = '↓';
        } else if (trend === 'moving-away') {
            trendIcon = '↑';
        }

        distanceIndicator.innerHTML = `
            ${formattedDistance}
            <span class="distance-trend ${trend !== 'stable' ? 'trend-' + trend : ''}">${trendIcon}</
span>
            ${estimatedTimeMinutes > 0 ? `<br><small>~${estimatedTimeMinutes} min</small>` : ''}
        `;
        distanceIndicator.className = `distance-indicator ${distanceClass}`;
    }

    function checkConnectionStatus() {
        const now = Date.now();
        Object.keys(vehicleCache).forEach(mobileId => {
            const vehicle = vehicleCache[mobileId];
        if (vehicle) {
            const lastUpdate = vehicle.lastUpdate || 0;
            const isOnline = (now - lastUpdate) < TEN_MINUTES;

            if (!isOnline) {
                updateStatus(mobileId, 'offline');
                if (markers[mobileId]) {
                    const icon = L.divIcon({
                        className: 'vehicle-marker offline',
                        html: `
                            <div class="vehicle-marker offline"></div>
                            <div class="vehicle-label">${vehicleNames[mobileId] || mobileId}</div>
                        `,
                        iconSize: [32, 42],
                        iconAnchor: [16, 16],
                        popupAnchor: [0, -16]
                    });

                    markers[mobileId]
                        .setIcon(icon)
                        .setLatLng(INACTIVE_COORDS)
                        .setPopupContent(`<b>${vehicleNames[mobileId] || mobileId}</
b><br>Desconectado<br>Última actualización: ${new Date(lastUpdate).toLocaleTimeString()}`);
                }

                if (currentUserId === 'Funcionarios') {
                    const distanceElement = document.getElementById(`${mobileId.toLowerCase()}-
distance`);
                    if (distanceElement) {
                        const distanceIndicator = distanceElement.querySelector('.distance-indicator');
                        if (distanceIndicator) {
                            distanceIndicator.textContent = '--';
                            distanceIndicator.className = 'distance-indicator';
                        }
                    }
                }
            }
        }
```

```
    });
}

function createMarker(mobileId, coords, isOnline) {
    if (markers[mobileId]) return;
    const displayName = vehicleNames[mobileId] || mobileId;

    const container = L.divIcon({
        className: 'vehicle-marker-container',
        html: `
            <div class="vehicle-marker ${isOnline ? 'online' : 'offline'}"></div>
            <div class="vehicle-label">${displayName}</div>
        `,
        iconSize: [32, 42],
        iconAnchor: [16, 16],
        popupAnchor: [0, -16]
    });

    markers[mobileId] = L.marker(coords, {
        icon: container,
        title: displayName
    }).addTo(map)
    .bindPopup(`<b>${displayName}</b><br>${isOnline ? 'En ruta' : 'Desconectado'}`);
}

function updateMarker(mobileId, coords, isOnline) {
    const displayName = vehicleNames[mobileId] || mobileId;

    if (!markers[mobileId]) {
        createMarker(mobileId, coords, isOnline);
    } else {
        const container = L.divIcon({
            className: 'vehicle-marker-container',
            html: `
                <div class="vehicle-marker ${isOnline ? 'online' : 'offline'}"></div>
                <div class="vehicle-label">${displayName}</div>
            `,
            iconSize: [32, 42],
                iconAnchor: [16, 16],
                popupAnchor: [0, 16]
            });

            markers[mobileId]
                .setLatLng(coords)
                .setIcon(container)
                .setPopupContent(`<b>${displayName}</b><br>${isOnline ? 'En ruta' :
'Desconectado'}<br>Última actualización: ${new Date().toLocaleTimeString()}`);
        }
    }

    function updateStatus(mobileId, status) {
        if (mobileId === currentUserId) {
            const element = document.getElementById('current-user-status');
            const iconElement = document.getElementById('current-user-icon');

            if (element) {
                element.textContent = status === 'online' ? 'En ruta' : 'Desconectado';
                element.className = status === 'online' ? 'online' : 'offline';
            }

            if (iconElement) {
                iconElement.className = `user-icon ${status === 'online' ? 'online' : 'offline'}`;
            }
        }
        else if (isViewer || currentUserId === 'Funcionarios') {
            const element = document.getElementById(`${mobileId.toLowerCase()}-status`);
            const iconElement = document.getElementById(`${mobileId.toLowerCase()}-icon`);

            if (element) {
                element.textContent = status === 'online' ? 'En ruta' : 'Desconectado';
                element.className = status === 'online' ? 'online' : 'offline';
            }

            if (iconElement) {
                iconElement.className = `vehicle-icon ${status === 'online' ? 'online' : 'offline'}`;
            }
    }
}

function logout() {
    if (gpsInterval) clearInterval(gpsInterval);
    if (locationUpdateInterval) clearInterval(locationUpdateInterval);
    if (viewerUpdateInterval) clearInterval(viewerUpdateInterval);
    if (weatherUpdateInterval) clearInterval(weatherUpdateInterval);

    if (backgroundGeolocationWatchId !== null) {
        navigator.geolocation.clearWatch(backgroundGeolocationWatchId);
    }

    if (wakeLock !== null) {
        wakeLock.release().then(() => {
            wakeLock = null;
        });
    }

    if (currentUserId && currentUserId.startsWith('Movil')) {
        firebase.database().ref(`activeUsers/${currentUserId}`).update({
            isActive: false,
            logoutTime: new Date().toISOString(),
            lastSeen: new Date().toISOString()
        }).then(() => {
            console.log('Estado actualizado en Firebase: usuario desconectado');
        }).catch((error) => {
            console.error('Error al actualizar estado en Firebase:', error);
        });
    }

    database.ref('vehicles').off();

    firebase.auth().signOut().then(() => {
        console.log('Sesión cerrada exitosamente');
    }).catch((error) => {
        console.error('Error al cerrar sesión:', error);
    });
```

```
            localStorage.removeItem('gpsAuth');
            localStorage.removeItem('panelState');

            window.location.href = 'login.html';
        }
    </script>
</body>
</html>
```