👋 Welcome

Live Location Tracking in Flutter with

`flutter_background_geolocation package`

🙋 Who Am I?

- Louis Eggermont, intern @ Get Driven
- Student MCT @ Howest
- Next web development
- Flutter dev experience from lessons, now applying it in internship
- Building location tracking feature from PoC to integration

🚗 About Get Driven

- Mobility platform with personal chauffeurs (in your own car!)
- Active in 🇧🇪 Belgium and 🇩🇪 Germany
- Driver and customer app

❓ The Problem I'm Trying to Solve

Why background location?

- Real-time driver visibility = customer trust
- Prevent joyriding, delays, “where is my driver?” moments
- App must update location even when closed or screen off

> Without background location, updates stop — breaking the experience.

🎯 Use Case

> A customer books a ride.
> A driver drives their car.
> The customer wants to see where the driver is — live.

We need:

- Continuous tracking during a ride
- Location history shown on a map
- Option to request current location
- Location updates even when app is terminated

👤 The Driver's Role

- Starts/stops ride
- Shares frequent location updates.
- Responds to customer requests for current location
- App must keep tracking even when closed.

🙋 The Customer’s Role

- Books a ride.
- Follows the driver’s route in-app.
- Can request current location.
- Views location history (if ride is tracked).

🔌 Package Evaluation

- ❌ `geolocator` – no background
- ❌ `background_locator` – unmaintained
- ⚙️ `background_location_tracker` - Simple, open-source solution
- 🔧 `flutter_background_service` – for custom handling
- ✅ `flutter_background_geolocation` – robust, cross-platform, headless, reliable

📦 flutter_background_geolocation

What?

- A powerful Flutter package for **background and headless geolocation**
- Built by **Transistor Software**
- Actively maintained and used in production apps

> 🛠️ Current version:** `v4.16.9` (April 2025)

🌟 Main Features

- ✅ Background tracking – works even if app is killed
- ✅ Motion-based tracking (moves only when needed)
- ✅ Battery optimization: pauses when stationary
- ✅ Geofencing support
- ✅ SQLite persistence (offline)
- ✅ Flexible HTTP layer (headless)
- ✅ Cross-platform support (same API for Android & iOS)

> Designed to be production-grade and highly configurable.

🔐 Licensing

💸 Licensing

- **Closed-source** (not open-source)
- **Production requires a paid license**
- Free for development/testing
- Optional dashboard for analytics, tracking, and remote config

📌 Why We Chose It

- Most **reliable** package for long-term background tracking
- Solves edge cases like:
  - iOS background mode quirks
  - Android app kill scenarios
  - Data buffering and offline syncing
  - Battery efficiency via motion/activity detection

⚠️ Requires native setup:

- Background modes (iOS), permissions, and service declarations

⚙️ Android Installation Flow

1. Update `AndroidManifest.xml`
2. Add dependencies to root `build.gradle`
3. Apply plugin script in `app/build.gradle`
4. Add license key to `AndroidManifest`

> 🧠 This setup ensures native Android compatibility, background tracking, and license validation.

❗ Why This Setup Is Needed

- Flutter doesn't auto-handle native Gradle + manifest merges
- License key required for production builds
- Some permissions are now injected via **merged manifest**

✅ From v4.16.9+, permissions are injected automatically if omitted

⚠️ Manual control is still recommended for clarity and customization

📲 Permission Flow Overview

`locationAuthorizationRequest: "Always"`

1. The OS shows a **"While using the app"** location prompt
2. The package may request an upgrade to **"Always Allow"** (depending on platform)
3. If denied, the SDK can show a **fallback dialog** to open app settings

> 🧠 The SDK automatically handles most flows, including dialogs and permission fallbacks.

🙋 Manually Requesting Permission

Use `requestPermission()` for custom flows:

```
int status = await BackgroundGeolocation.requestPermission();

if (status == ProviderChangeEvent.AUTHORIZATION_STATUS_ALWAYS) {
  print("✅ Authorized Always");
} else {
  print("⚠️ Limited or denied: $status");
}
```

🧠 The package already prompts the user on:

- `start()`
- `startGeofences()`
- `getCurrentPosition()`

Live demo

📍 Simulating Location: Lockito

Why Lockito?

- Simulates GPS movement for testing real-world routes
- Lets us trace **laps around Spa-Francorchamps** 🏎️
- Useful for testing background tracking

⚠️ Must enable Lockito in Settings as mock location app
⚠️ Manually set `isMoving: true`

🧩 Code: geolocation config

```
bg.BackgroundGeolocation.ready(
    bg.Config(
      desiredAccuracy: bg.Config.DESIRED_ACCURACY_HIGH,
      geofenceModeHighAccuracy: true, // High accuracy for geofencing

      enableHeadless: true, // Enables headless mode for background tracking
      stopOnTerminate: false, // Continue tracking when the app is terminated
      startOnBoot: true, // Automatically start tracking on device boot

      distanceFilter: 0, // Time-based tracking (distance filter disabled)
      locationUpdateInterval: 10000, // Location update interval (10 seconds)
      fastestLocationUpdateInterval: 5000, // Fastest interval (5 seconds)

      debug: false, // Disable debug notifications
      isMoving: true, // Sets pace to prevent automatic stopping of tracking
      disableStopDetection: true, // Prevent automatic stopping of tracking
      stopTimeout: 0, // Disable automatic stop timeout
    ),
  );
```

🧩 Code: Start and stop tracking

```
bg.BackgroundGeolocation.start();

bg.BackgroundGeolocation.stop();
```

🧩 Code: onLocation listener

```
bg.BackgroundGeolocation.onLocation((bg.Location location) {
  // 📍 When a new GPS location is received:
  // → Handle the location update (store, sync, or notify)
});
```

```
{
  "lat": 50.44328,
  "lon": 5.96588,
  "timestamp": 1680451200,
  "battery": 85.0,
  "speed": 12.5
}
```

Headless Tasks

A special **background function** that runs even when your Flutter app is **completely killed.**

✅ Useful for:

- Responding to silent push requests
- Handling background location updates
- Persisting data (e.g., to Hive or Firestore)
- Maintaining continuity in background

> When the app is terminated, the plugin can **still receive location updates** via Android's **headless mode.**

✅ Config Recap

```
bg.BackgroundGeolocation.ready(Config(
  enableHeadless: true,
  stopOnTerminate: false,
  startOnBoot: true,
));
```

🧩 Code: headless tasks

```
void main() async {

  runApp(const ProviderScope(child: MyApp()));

  bg.BackgroundGeolocation.registerHeadlessTask(headlessTask);
}
```

```
@pragma('vm:entry-point')
void headlessTask(bg.HeadlessEvent event) async {
  if (event.name == bg.Event.LOCATION) {
    // Do something with location (e.g., save or send locaton)
  }
}
```

❌ Limitations of Headless Tasks

> ⚠️ Headless tasks are powerful, but have **strict boundaries**:

- 🧱 **No UI access** – you cannot show dialogs, alerts, or navigate routes
- ❗ Must be **registered in** `main.dart` only
- ♻️ You can **only register one** global headless function (switch case)
- 🧵 Heavy or long-running operations may be **killed by the OS**
- 🛌 May **not be triggered immediately** (depends on OS state, permissions)

> 💡 Use it to persist data, sync to server, or trigger logic on events like:LOCATION, MOTIONCHANGE, TERMINATE, GEOFENCE, HEARTBEAT, etc.

🔐 iOS? Not Quite.

- iOS **does not support headless Dart tasks**
- Instead, the entire app is background-launched (cold-started)
- You must handle iOS logic via normal app lifecycle + background modes

iOS Background Tracking Simplified

iOS doesn't let apps run freely in background, Instead:

1. App sleeps when closed
2. iOS wakes it up automatically when:
    - Significant location change detected (~500m movement)
    - Periodic safety checks (every 15-30 mins)
3. App quickly sends location
4. Goes back to sleep

What Users Experience:

- 🔵 Blue status bar appears briefly when tracking
- Location updates less frequent when app closed (~5-15 mins)
- Updates become real-time when app opens
- Battery impact minimized by iOS

> **Trust iOS Magic:**
> iOS handles wake-ups and package automatically resumes tracking when awoken

🧰 Supporting Tools (I used)

- **Sendbird** – messaging layer for real-time location
- **Riverpod** – state management
- **Hive** – offline buffering
- **Lockito** – GPS simulation

🌐 HTTP Layer Integration

Server Communication Made Easy

✅ Features:

- **Batch uploads** when network available
- **Retry mechanism** for failed requests
- **Custom HTTP methods** (POST/PUT/PATCH)
- **Custom JSON template** support

> Ensures reliable delivery even with spotty connections.

Automatic location syncing with your backend:

```
bg.Config(
  url: "https://your-api.com/locations",
  headers: {
    "Authorization": "Bearer YOUR_TOKEN",
    "Content-Type": "application/json"
  },
  autoSync: true,
  autoSyncThreshold: 5 // Sync after 5 locations
);
```

💾 SQLite Persistence

Offline-First Location Storage

Key capabilities:

- Store locations when offline 📴
- Query locations by date/radius 🗺️
- Automatic purging of old data 🗑️
- Direct database access via:

```
List<Location> locations = await bg.BackgroundGeolocation.getLocations();
```

> Critical for maintaining location history during network outages.

Local database configuration:

```
bg.Config(
  reset: false, // Maintain database between launches
  maxDaysToPersist: 30, // Keep data for 30 days
  maxRecordsToPersist: 10000, // Storage limit
  locationsOrderDirection: "DESC" // Query order
);
```

⚠️ Challenges Faced

- Sendbird SDK doesn’t support background mode → needed REST workaround

🚀 What's Next?

- Customer can request location
- Sendbird REST middleware via backend
- Background response improvements
- UI polish → show battery/speed
- Integration in production Driver app

📚 Further Reading & Resources

- `flutter_background_geolocation` Documentation
- Repo of POC and presentation
- Lockito (Android GPS Simulation)

> For questions, feel free to reach out or explore more on my GitHub 👇

🙋 Q&A Time!

> Thanks for listening!
> GitHub: **https://github.com/LouisEggermont**
> Slack: Flutter Belgium