Suivi cyclistes temps réel — Présentation dev team (variantes)

Auteur : Dr Hamid MADANI drmdh@msn.com **Date** : 2026-06-05 **Statut** : PROPOSITION technique à discuter **Audience** : développeurs / intégrateurs / ops **Liés** : référence complète `../ARCHITECTURES.md`, slides dev `SLIDES-DEV-VARIANTES.md`

> Synthèse technique pour la revue d'architecture : composition d'écosystème, 5 variantes, et points d'intégration. Le détail exhaustif est dans `../ARCHITECTURES.md`.

---

1. Principe directeur

`@mostajs/geo` = **seule brique géo** (cœur pur, serveur **ou** device). Tout le reste est **délégué** (DEVRULES §10) : config, persistance, transport, temps réel, vidéo.

```
config ─┐
        ├─ data-plug ─► orm (DB interchangeable)
geo ────┤
        ├─ net (REST/SSE/WS, fan-out)
        └─ SFU/MCU studio.amia.fr (vidéo)
```

2. Modules & rôles

Module	API utilisée	Rôle
`@mostajs/config`	`getEnv`, `getEnvNumber`	env + profils `MOSTA_ENV`
`@mostajs/data-plug`	`registerSchemas`, `getDialect()`, `upsert/create`	persistance interchangeable (orm/net)
`@mostajs/orm`	`EntitySchema`	entités `Cyclist/Position/Checkpoint/Crossing`
`@mostajs/net`	`startServer()`	publication temps réel multi-protocole
`@mostajs/geo`	`distance/bearing/nearest/isWithin/geohashEncode`	calculs géo

Code partagé : `schema.mjs`, `common/course.mjs` (`computeGeo()`, `simulatePeloton()`).

3. Les 5 variantes (résumé)

Var.	Entrée	Transport positions	Vidéo	Spécificité technique
A	`tracker.mjs`	net SSE	SFU viewer	le plus simple ; calcul géo serveur
B	`gateway.mjs`	net (via moto 4G)	SFU (caméra moto)	filtrage par `distance()` / `nearest()` (coureurs du rayon)
C	`ingest-mqtt.mjs`	MQTT → net SSE/WS	SFU→**MCU**→**HLS/CDN**	découplage broker + egress (`egress-notes.md`)
D	`edge-agent.mjs`	net (mode net)	—	geo **sur device** + store-and-forward (retry)
B+C	`docker-compose.yml`	MQTT + SSE	SFU + HLS	orchestration mqtt+net+ingest+PostGIS+egress

4. Points d'intégration clés

- **Persistance interchangeable** : `MOSTA_DATA=orm` (SQLite/PostgreSQL/PostGIS/Mongo) ou `net` — **sans changer le code applicatif**.
- **Temps réel** : SSE (simple) / WS (bidir) / MQTT (IoT, QoS, offline) selon la variante.
- **Vidéo** : SFU `studio.amia.fr/sfu` (régie, <0,5 s) ; MCU `studio.amia.fr/mcu` (mixage) ; egress **HLS/LL-HLS** pour le grand public.
- **Edge** : `@mostajs/geo` est un cœur **pur sans dépendance** → `computeGeo()` tourne aussi sur device (variante D).
- **Pièges geo** : GeoJSON `[lon,lat]` ≠ `{lat,lon}` ; `provider osm` = Nominatim 1 req/s ; position = donnée perso.

5. Exécuter / tester

```
cp .env.example .env
npm i @mostajs/config @mostajs/data-plug @mostajs/orm @mostajs/geo
node variants/A-device-direct/tracker.mjs            # A (simulateur runnable)
PUBLISH=1 node variants/C-fanout-massif/ingest-mqtt.mjs   # C (broker MQTT requis)
node variants/D-edge-offline/edge-agent.mjs          # D (coupures réseau simulées)
```

Dashboards web : `variants/A-device-direct/web/` et `variants/C-fanout-massif/web/`.

6. Décision rapide

PoC → **A** · course route → **B** · grand public → **C** · zones blanches → **D** · épreuve complète → **B+C**.

Proposition technique — Dr Hamid MADANI drmdh@msn.com — 2026-06-05. Détail : `../ARCHITECTURES.md`.