

# AVANCE — Guide d'installation

Plateforme de transport adapté & médical — panneau d'administration (web) + API mobile chauffeur.

Laravel 13 (PHP 8.3+) · Blade + TailwindCSS/Vuexy · Sanctum · Leaflet/OpenStreetMap

## 1. Prérequis

| Outil           | Version / note                                                                                                                                        |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| PHP             | <b>8.3+</b> (extensions : pdo, mbstring, openssl, tokenizer, xml, ctype, json, curl, fileinfo, et <code>pdo_sqlite</code> ou <code>pdo_mysql</code> ) |
| Composer        | 2.x                                                                                                                                                   |
| Node.js + npm   | Node 18+ (le build utilise Vite 8)                                                                                                                    |
| Base de données | <b>SQLite</b> (par défaut, aucun serveur requis) — ou MySQL 8 / MariaDB                                                                               |

Sur macOS : `brew install php composer node` . Sur Ubuntu : `apt install php8.3 php8.3-sqlite3 php8.3-mbstring php8.3-xml php8.3-curl composer nodejs npm` .

## 2. Installation

**Étape 1** — Dépendances PHP :

```
composer install
```

**Étape 2** — Dépendances front-end :

```
npm install
```

**Étape 3** — Fichier d'environnement + clé d'application :

```
cp .env.example .env
php artisan key:generate
```

**Étape 4** — Base de données.

**Option A** — SQLite (le plus simple, recommandé en local)

```
touch database/database.sqlite
```

Dans `.env` , mettre :

```
DB_CONNECTION=sqlite
DB_DATABASE=/chemin/absolu/vers/database/database.sqlite
```

(Commentez ou supprimez les lignes DB\_HOST / DB\_PORT / DB\_USERNAME / DB\_PASSWORD.)

### Option B — MySQL / MariaDB

Créez une base `avance`, puis dans `.env` :

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=avance
DB_USERNAME=root
DB_PASSWORD=motdepasse
```

**Étape 5** — Migrations + données de départ (crée le compte admin) :

```
php artisan migrate --seed
```

**Étape 6** — Lien de stockage (images/avatars) — **indispensable** :

```
php artisan storage:link
```

**Étape 7** — Compiler les assets front-end :

```
npm run build
```

## 3. Lancer l'application

Le plus simple (serveur PHP seul) :

```
php artisan serve
```

L'application est disponible sur `http://127.0.0.1:8000`.

Ou tout en une commande (serveur + file d'attente + logs + Vite) :

```
composer dev
```

## 4. Accès par défaut

| Élément       | Valeur                                   |
|---------------|------------------------------------------|
| Panneau admin | <code>http://127.0.0.1:8000/login</code> |

Courriel admin

admin@example.com

Mot de passe

password

**Sécurité :** changez ce mot de passe immédiatement (page *Mon Profil*) et ne lancez jamais le seeder tel quel en production.

## 5. Documentation API (Swagger)

| Surface              | URL                       |
|----------------------|---------------------------|
| API Admin            | /api/documentation        |
| API Mobile Chauffeur | /api/documentation/driver |

Régénérer les specs après modification des annotations :

```
php artisan l5-swagger:generate --all
```

## 6. Tests

```
php artisan test
```

Les tests utilisent une base SQLite en mémoire — aucune configuration supplémentaire nécessaire.

## 7. Dépannage & points d'attention

### Le build front-end échoue (binaire natif introuvable)

Vite 8 utilise un binaire natif (rolldown) propre à la plateforme. Si `npm run build` se plaint d'un binding manquant :

```
rm -rf node_modules package-lock.json
npm install
npm run build
```

### Les images/avatars renvoient une erreur 403 ou ne s'affichent pas

Le lien symbolique de stockage est manquant. Vérifiez que `public/storage` est bien un *lien* (et non un dossier), sinon :

```
rm -rf public/storage
php artisan storage:link
```

## Les permissions « `command not found` » sur binaires locaux

Si `./vendor/bin/*` ou `node_modules/.bin/*` ne sont pas exécutables, préfixez par l'interpréteur : `php vendor/bin/pint`, ou `chmod +x` sur le dossier concerné.

## Carte & calcul de distance

La carte des courses utilise des services publics gratuits d'OpenStreetMap : **Nominatim** (géocodage) et **OSRM** (itinéraire voiture + distance). Ils ont des limites d'usage : pour un fort trafic en production, hébergez vos propres instances ou utilisez un fournisseur avec clé API (à modifier dans `resources/views/partials/ride-map.blade.php`).

## E-mails & notifications

En local, `MAIL_MAILER=log` (les e-mails vont dans les logs). Les notifications sont mises en file d'attente : lancez un worker ( `php artisan queue:work`, inclus dans `composer dev` ) pour qu'elles partent réellement. En production, configurez un vrai SMTP.

## 8. Liste de vérification — Production

- Générer une nouvelle `APP_KEY` et mettre `APP_ENV=production`, `APP_DEBUG=false`.
- Configurer une vraie base de données (MySQL/PostgreSQL) et un vrai `APP_URL` (HTTPS).
- Configurer un vrai serveur SMTP ( `MAIL_*` ) et exécuter un worker de file d'attente supervisé.
- Changer le mot de passe admin par défaut ; ne pas exécuter les seeders de démonstration.
- Optimiser : `php artisan config:cache route:cache view:cache` et `npm run build`.
- Servir `public/` derrière Nginx/Apache (pas `artisan serve`).

**Résumé express :** `composer install` → `npm install` → `cp .env.example .env` → `php artisan key:generate` → (SQLite) `touch database/database.sqlite` → `php artisan migrate --seed` → `php artisan storage:link` → `npm run build` → `php artisan serve`.