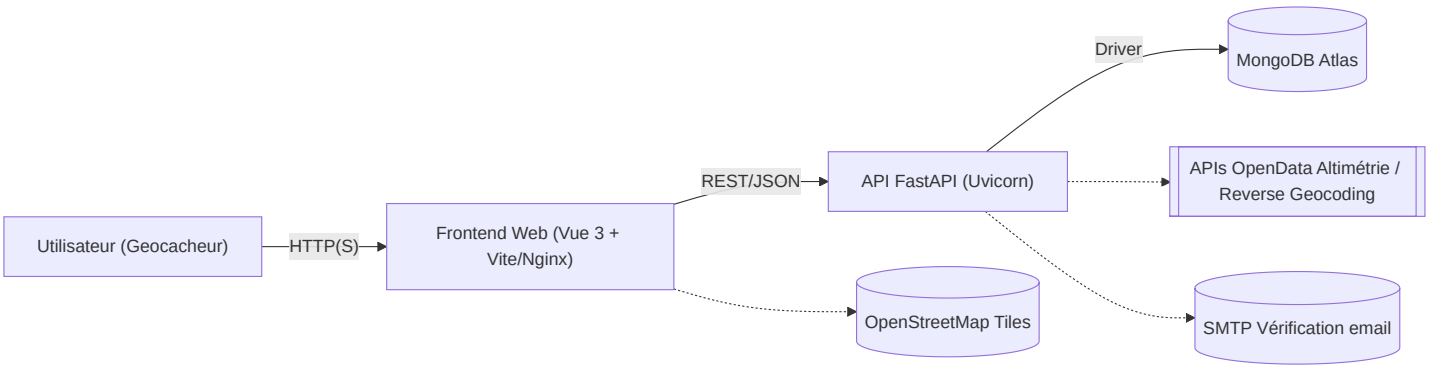
5. Architecture logicielle

Version 1.0 — basée sur l'API exposée (OpenAPI), l'arborescence backend, et les fichiers Docker.

5.1 Vue d'ensemble (Contexte)



- Application **web mobile-first** : frontend Vue communiquant avec une **API REST** FastAPI.
- Données persistées dans **MongoDB Atlas** (modèle documentaire).
- Intégrations externes : **OpenStreetMap** (tuiles), **APIs OpenData** (altimétrie, commune), **SMTP** (validation email).

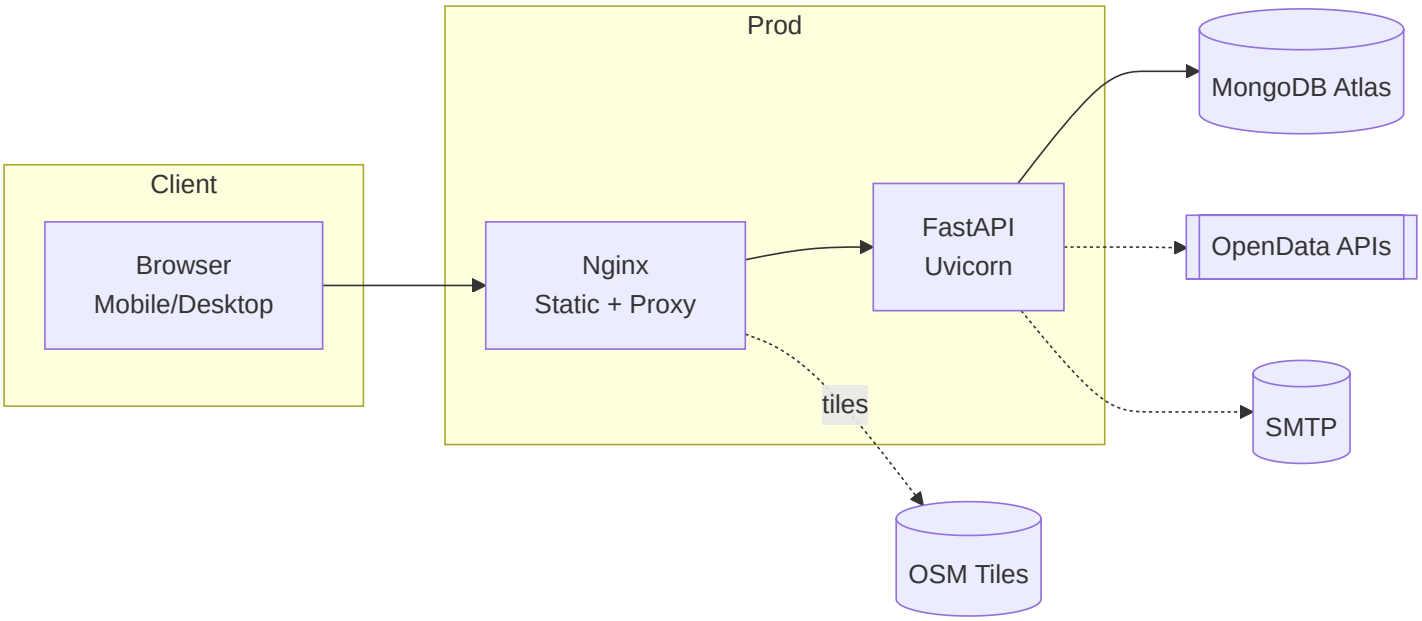
5.2 Vue conteneurs (environnements Dev/Prod)

Dev (docker-compose)

- **frontend** : Vite (HMR) sur `5173/tcp`.
- **backend** : FastAPI/Uvicorn sur `8000/tcp`.
- **MongoDB** : Atlas (DB managée) — accès via variables d'environnement.

Prod

- **frontend** : build statique servi par **Nginx**.
- **backend** : Uvicorn derrière reverse proxy (Nginx/ingress selon hébergeur).
- **CI/CD** : GitHub Actions (tests → build images → déploiement).



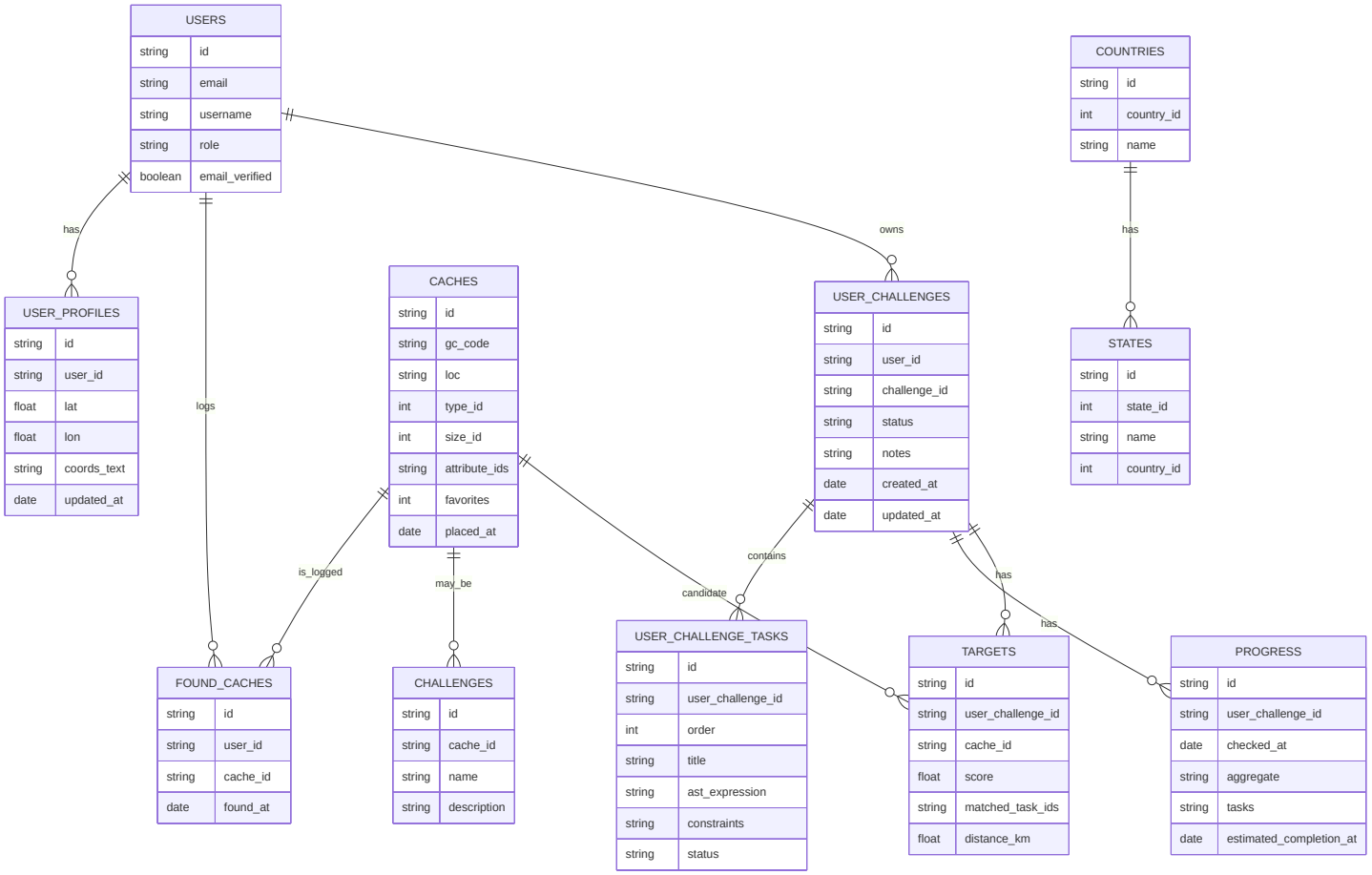
5.3 Composants backend (modules principaux)

Organisation logique issue du code et du contrat OpenAPI :

- **API / Routes**
  - `auth` : inscription, login, refresh, vérification email, renvoi code.
  - `caches` : upload GPX/ZIP, recherche par filtres, BBox / rayon, get par GC/ID.
  - `caches_elevation` : backfill altimétrie (admin).
  - `challenges` : (re)construction depuis caches *challenge* (admin).
  - `my-challenges` : lister/synchroniser/patcher mes *UserChallenges*.
  - `my-challenge-tasks` : lire/remplacer/valider les **tâches** d'un *UserChallenge*.
  - `my-challenge-progress` : lire/évaluer snapshots de progression.
  - `targets` : calculer/lister/effacer les **targets** (caches utiles).
  - `my_profile` : localisation utilisateur (get/put).
  - `maintenance` : endpoints réservés (diagnostic/maintenance).
- **Services**
  - `gpx_importer` : parsing GPX multi-namespaces, upsert caches, dédoublonnage.
  - `challenge_autocreate` : détection *challenge caches* → upsert `challenges`.
  - `user_challenges` : liste par utilisateur, sync `pending`.
  - `user_challenge_tasks` : validation/PUT complet des tâches (AST + contraintes).
  - `progress` : évaluation, snapshots, projections (courbes cumulées, estimations).
  - `targets` : agrégation caches satisfaisant ≥1 tâche, scoring, filtrage géo.
  - `elevation_retrieval` : backfill altimétrie (quotas, pagination, dry-run).
  - `providers` : adaptateurs vers APIs OpenData (altimétrie, reverse geocoding).
  - `referentials_cache` : référentiels (types, tailles, attributs, pays/états).
  - `query_builder` : construction de filtres Mongo à partir de règles/AST.
- **Core / Infra**

- `core/settings` : configuration (env vars, secrets).
- `core/security` : JWT + refresh, rôles `user` / `admin`, règles password, bcrypt.
- `db/mongodb` : connexion, index, seeds.

5.4 Modèle logique des données (MongoDB)



Index créés (via seeding)

- **users**
  - `username` (unique, collation insensible à la casse)
  - `email` (unique, collation insensible à la casse)
  - `is_active`, `is_verified`
  - `location` (2dsphere)
- **countries**
  - `name` (unique)
  - `code` (unique, partial string)
- **states**
  - `(country_id, name)` (unique)
  - `(country_id, code)` (unique, partial string)
  - `country_id`

- **cache_attributes**
  - `cache_attribute_id` (unique)
  - `txt` (unique, partial string)
  - `name`
- **cache_sizes**
  - `name` (unique)
  - `code` (unique, partial string)
- **cache_types**
  - `name` (unique)
  - `code` (unique, partial string)
- **caches**
  - `GC` (unique)
  - `type_id`, `size_id`, `country_id`, `state_id`, `(country_id, state_id)`
  - `difficulty`, `terrain`, `placed_at`
  - `title + description_html` (index texte)
  - `loc` (2dsphere)
  - `(attributes.attribute_doc_id, attributes.is_positive)`
  - `(type_id, size_id)`
  - `(difficulty, terrain)`
- **found_caches**
  - `(user_id, cache_id)` (unique)
  - `(user_id, found_date)`
  - `cache_id`
- **challenges**
  - `cache_id` (unique)
  - `name + description` (index texte)
- **user_challenges**
  - `(user_id, challenge_id)` (unique)
  - `(user_id, status, updated_at)`
  - `user_id`, `challenge_id`, `status`
- **user_challenge_tasks**
  - `(user_challenge_id, order)`
  - `(user_challenge_id, status)`
  - `user_challenge_id`
  - `last_evaluated_at`
- **progress**

- `(user_challenge_id, checked_at)` (unique)

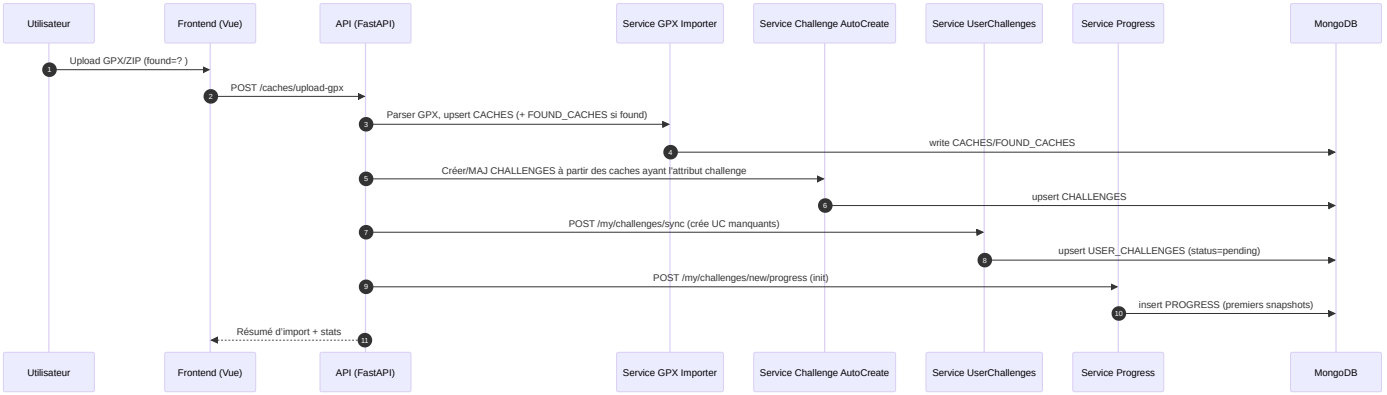
- **targets**
  - `(user_challenge_id, cache_id)` (unique)
  - `(user_challenge_id, satisfies_task_ids)`
  - `(user_challenge_id, primary_task_id)`
  - `cache_id`
  - `(user_id, score)`
  - `(user_id, user_challenge_id, score)`
  - `loc` (2dsphere)
  - `(updated_at, created_at)`

5.5 Sécurité (vue architecture)

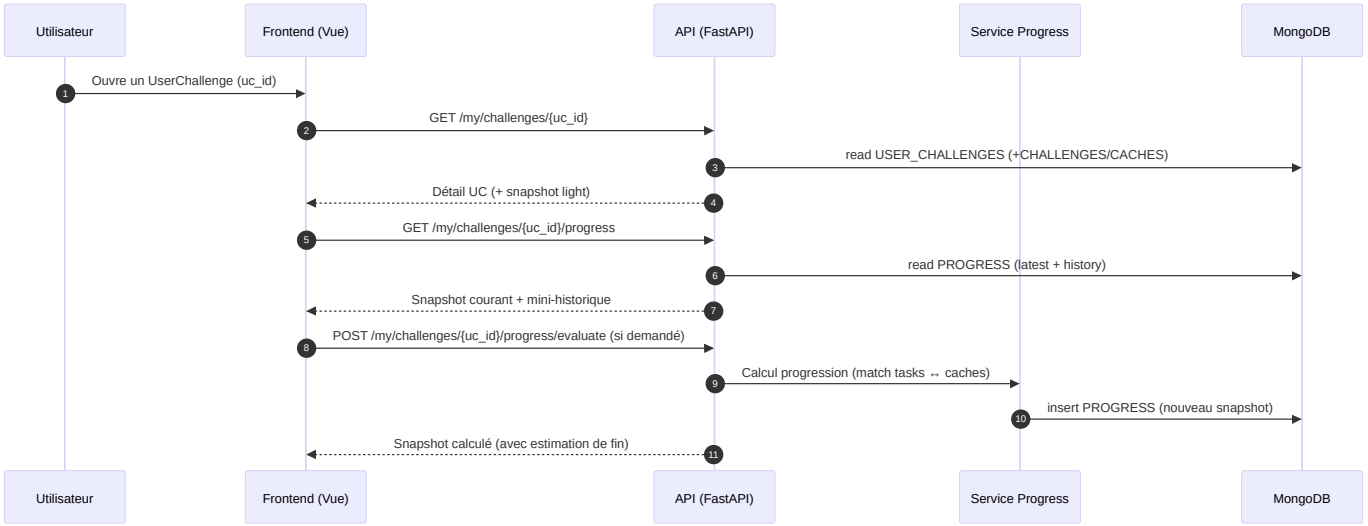
- **Auth** : OAuth2 password flow (login) → **JWT** access + **refresh** (durées distinctes).
- **Rôles** : `user` / `admin` (endpoints d'admin protégés).
- **Comptes** : règles password strictes, **bcrypt + sel variable**, vérification email par lien **temporaire**.
- **Front** : stockage tokens (access en mémoire, refresh en stockage sécurisé si nécessaire), gardes de routes.
- **Back** : contrôle systématique `userId` côté serveur (aucune confiance dans le client), CORS configuré.
- **APIs externes** : clés en variables d'environnement ; timeouts, retries, backoff ; respect des quotas.

5.6 Séquences clés (UML séquence)

5.6.1 Import GPX → sync challenges → premiers snapshots



5.6.2 Consultation d'un challenge et projection



5.7 Performance & cache

- **Clustering** des marqueurs sur carte, **tile caching** OSM côté front.
- **Caps** sur les calculs de targets (`limit_per_task` , `hard_limit_total`).
- **Pagination** standardisée (listes, nearby, bbox, radius ; page/limit ≤ 200).
- **Indices** adaptés (cf. 5.4) pour requêtes géo + filtres combinés.

5.8 Déploiement & configuration

- **Docker** : images séparées frontend/backend ; compose pour dev.
- **Variables d’environnement** : DB (URI/credentials), secrets JWT, SMTP, fournisseurs OpenData.
- **CI/CD** : GitHub Actions — lint/tests → build → déploiement (Railway/équivalent).
- **Fichiers** : uploads GPX (répertoire dédié, nettoyage planifié recommandé).

5.9 Évolutions d’architecture envisagées

- **Mode offline** : IndexedDB + stratégie de sync.
- **Multi-challenges map** : vue combinée (optimisation multi-objectifs).
- **Cache applicatif** côté API (targets/progress récents).
- **Observabilité** : métriques Prometheus, traces OpenTelemetry.

5.10 Glossaire rapide

- **UC / UserChallenge** : instance d’un challenge pour un utilisateur.
- **Task** : condition exprimée via **AST** (AND/OR/NOT + règles feuille).
- **Target** : cache candidate satisfaisant ≥1 tâche.
- **Progress snapshot** : état horodaté (global + par tâche).